BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

MODELING AND OPTIMIZATION OF HIGH-PERFORMANCE MANY-CORE SYSTEMS FOR ENERGY-EFFICIENT AND RELIABLE COMPUTING

by

JIE MENG

B.S., University of Science and Technology of China, 2004
M.A.Sc., McMaster University, 2008

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2013

UMI Number: 3575304

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

UMI
Dissertation Publishing

UMI 3575304
Published by ProQuest LLC 2013. Copyright in the Dissertation held by the Author.
Microform Edition © ProQuest LLC.
All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code.

ProQuest

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

© 2013 by
Jie Meng
All rights reserved.

Approved by

First Reader

Ayse K. Coskun, PhD
Assistant Professor of Electrical and Computer Engineering

Second Reader

Martin Herbordt, PhD
Associate Professor of Electrical and Computer Engineering

Third Reader

Ajay Joshi, PhD
Assistant Professor of Electrical and Computer Engineering

Fourth Reader
Allyn E Hubbard

Allyn E. Hubbard, PhD
Professor of Electrical and Computer Engineering
Professor of Biomedical Engineering

Fifth Reader

Arun Rodrigues, PhD
Researcher at Sandia National Laboratories

If you cry because the sun has gone out of your life, your tears will prevent you from seeing the stars.

Rabindranath Tagore

Acknowledgments

I would like to express my gratitude to my advisor, Professor Ayse K. Coskun, for her inspirational guidance, invaluable support, and constant encouragement throughout my graduate school career. She has been not only a supportive advisor but also a role model for me.

I would like to thank Professor Martin Herbordt for his guidance and valuable feedback on my research projects as well as his helpful career advice and suggestions. I also thank Professor Ajay Joshi for his advice and comments that led to the publication of my first paper at Boston University.

I would like to give special thanks to Professor Allyn E. Hubbard and Professor Ari Trachtenberg for their guidance when I was trying to find my research direction. I would also like to thank my doctoral committee, Professor Martin Herbordt, Professor Ajay Joshi, Professor Allyn E. Hubbard, and Dr. Arun Rodrigues for their feedback and contributions.

I would like to express my sincerest appreciation to Dr. Arun Rodrigues and Dr. Mingyu Hsieh for their advice and support during my internship at Sandia National Laboratories. I would also like to thank Mr. Xin Ma and Mr. Jim Ignowski for the summer internship opportunity at Intel Corporation, Hudson, MA.

I would like to thank my fellow lab mates, co-authors, and friends at Boston University for their friendship and for their encouragement during the sleepless nights of working together before paper deadlines. I am also grateful to our collaborators and co-authors Dr. Mohamed M. Sabry, Arvind Sridhar, and Professor David Atienza at École Polytechnique Fédérale de Lausanne (EPFL) for their productive collaboration and the stimulating discussions.

Finally, I would like to express my deepest gratitude to my family for their encouragement, understanding, and unconditional support.

The research that forms the basis of this dissertation has been partially funded by NSF CAREER grant CNS-1149703, Sandia National Laboratories, and Design Automation Conference A. Richard Newton Graduate Scholarship.

The contents of Chapter 3 and 4 are in part reprints of the material from the papers, *Jie Meng and Ayse Coskun, "Analysis and Runtime Management of 3D Systems with Stacked DRAM for Boosting Energy Efficiency"*, in Proceedings of Design Automation and Test in Europe Conference (DATE), 2012, and *Jie Meng, Katsutoshi Kawakami, and Ayse Coskun, "Optimizing Energy Efficiency of 3D Multicore Systems with Stacked DRAM under Power and Thermal Constraints"*, in Proceedings of Design Automation Conference (DAC), 2012.

The content of Section 4.3 is in part a reprint of the material from the papers, *Ayse K. Coskun, Jie Meng, David Atienza, and Mohamed M. Sabry, "Attaining Single-Chip, High-Performance Computing through 3D Systems with Active Cooling"*, in IEEE Micro, Special Issue on Big Chips, August, 2011, and *Mohamed M. Sabry, Arvind Sridhar, Jie Meng, Ayse K. Coskun, and David Atienza , "GreenCool: An Energy-Efficient Liquid Cooling Design Technique for 3D MPSoCs via Channel Width Modulation"*, in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, April, 2013.

The contents of Chapter 5 and 6 are in part reprints of the material from the papers, *Jie Meng, Fulya Kaplan, Mingyu Hsieh, and Ayse K. Coskun, "Topology-Aware Reliability Optimization for Multiprocessor Systems"*, in Proceedings of International Conference on VLSI and System-on-Chip (VLSI-SoC), 2012, and *Fulya Kaplan, Jie Meng, and Ayse K. Coskun, "Optimizing Communication and Cooling Costs in HPC Data Centers via Intelligent Job Allocation"*, in Proceedings of International Green Computing Conference (IGCC), 2013.

MODELING AND OPTIMIZATION OF HIGH-PERFORMANCE MANY-CORE SYSTEMS FOR ENERGY-EFFICIENT AND RELIABLE COMPUTING

(Order No.)

JIE MENG

Boston University, College of Engineering, 2013

Major Professor: Ayse K. Coskun, PhD, Assistant Professor of Electrical and Computer Engineering

ABSTRACT

Many-core systems, ranging from small-scale many-core processors to large-scale high performance computing (HPC) data centers, have become the main trend in computing system design owing to their potential to deliver higher throughput per watt. However, power densities and temperatures increase following the growth in the performance capacity, and bring major challenges in energy efficiency, cooling costs, and reliability. These challenges require a joint assessment of performance, power, and temperature tradeoffs as well as the design of runtime optimization techniques that monitor and manage the interplay among them. This thesis proposes novel modeling and runtime management techniques that evaluate and optimize the performance, energy, and reliability of many-core systems.

We first address the energy and thermal challenges in 3D-stacked many-core processors. 3D processors with stacked DRAM have the potential to dramatically improve performance owing to lower memory access latency and higher bandwidth.

However, the performance increase may cause 3D systems to exceed the power budgets or create thermal hot spots. In order to provide an accurate analysis and enable the design of efficient management policies, this thesis introduces a simulation framework to jointly analyze performance, power, and temperature for 3D systems. We then propose a runtime optimization policy that maximizes the system performance by characterizing the application behavior and predicting the operating points that satisfy the power and thermal constraints. Our policy reduces the energy-delay product (EDP) by up to 61.9% compared to existing strategies.

Performance, cooling energy, and reliability are also critical aspects in HPC data centers. In addition to causing reliability degradation, high temperatures increase the required cooling energy. Communication cost, on the other hand, has a significant impact on system performance in HPC data centers. This thesis proposes a topology-aware technique that maximizes system reliability by selecting between workload clustering and balancing. Our policy improves the system reliability by up to 123.3% compared to existing temperature balancing approaches. We also introduce a job allocation methodology to simultaneously optimize the communication cost and the cooling energy in a data center. Our policy reduces the cooling cost by 40% compared to cooling-aware and performance-aware policies, while achieving comparable performance to performance-aware policy.

Contents

1 **Introduction** **1**

1.1 Problem Statement 3

1.2 Thesis Contributions 6

2 **Background and Related Work** **10**

2.1 Background 10

2.2 Modeling and Management of 3D Many-core Systems 11

2.3 Energy and Reliability Management in Servers and Data Centers . . . 14

2.4 Distinguishing Aspects from Prior Work 16

3 **Modeling of 3D Many-core Systems** **19**

3.1 Overview 19

3.2 Methodology for Modeling 3D Many-core Systems 22

3.2.1 Modeling Target 3D Systems with DRAM Stacking 22

3.2.2 Modeling 3D On-chip DRAM Accesses 25

3.2.3 Performance Simulation of 3D Many-core Systems 30

3.2.4 Modeling the Power Consumption of 3D Many-core Systems . 31

3.2.5 Modeling the Temperature of 3D Many-core Systems 33

3.3 Performance, Energy, and Temperature Evaluation of 3D Many-core Processors 35

3.3.1 Performance Evaluation of 3D Many-core Systems 35

3.3.2 Power Evaluation of 3D Many-core Systems 38

3.3.3 Temperature Analysis of 3D Many-core Systems 40

3.4 Summary . . . 40

4 Runtime Management of 3D Many-core Systems 42

4.1 Overview . . . 42
4.2 Runtime Management for 3D Many-core Systems . . . 44
4.3 Managing 3D Many-core Systems with Liquid Cooling . . . 55
4.4 Summary . . . 59

5 Modeling of Many-core Systems in Data Centers 61

5.1 Overview . . . 61
5.2 Performance and Cooling Energy Modeling in HPC Data Centers . . . 62
  5.2.1 Workload and Performance Model . . . 63
  5.2.2 Cooling Energy Model . . . 65
  5.2.3 Temperature Model . . . 72
5.3 Reliability Modeling for Many-core Systems in HPC Data Centers . . . 73
  5.3.1 Wear-out Failure Mechanisms . . . 73
  5.3.2 Lognormal Distributions for Lifetime Reliability . . . 74
  5.3.3 System Reliability Modeling . . . 77
  5.3.4 Topology and System Reliability Analysis . . . 77
5.4 Summary . . . 81

6 Runtime Management of Many-core Systems in Data Centers 83

6.1 Overview . . . 83
6.2 Topology-Aware Reliability Optimization . . . 85
  6.2.1 Experimental Methodology . . . 88
  6.2.2 Evaluation Results . . . 91
6.3 Joint Performance and Cooling Cost Optimization for Data Centers . . . 96
  6.3.1 Performance-aware Job Allocation . . . 96
  6.3.2 Cooling-aware Job Allocation Policy . . . 97

6.3.3 Joint Optimization Policy for Data Center Job Allocation . . 99
6.3.4 Experimental Results 102
6.4 Summary 110

7 Conclusion and Future Research Directions 112
7.1 Conclusion 112
7.1.1 A Simulation Framework and Runtime Optimization for Boosting Energy Efficiency in 3D Many-core Processors 112
7.1.2 Optimizing the Reliability, Performance, and Cooling Cost of Many-core Systems in HPC Data Centers via Workload Allocation 114
7.2 Future Research Directions 115
7.2.1 3D Stacked Systems 115
7.2.2 HPC Data Centers 118

References 125

Curriculum Vitae 135

List of Tables

3.1 3D system core architecture parameters. 23

3.2 Dimensions of the blocks in the target 3D systems. 25

3.3 DRAM access latency. 29

3.4 Thermal simulation configuration in HotSpot. 34

4.1 Regression coefficients for a target 3D system with $85^oC/175$W constraints for all the V-F settings. 47

4.2 Results of the target 3D system with static settings. 49

4.3 Results of the target 3D system with our runtime optimization policy. 49

6.1 Intel Clovertown core architecture parameters. 89

6.2 Workload characteristics. 90

6.3 Dimensions of the target system. 91

6.4 Simulation results for the single-row job allocation. 103

List of Figures

1·1 The report from U.S. environmental protection agency to the Congress on server and data center energy efficiency shows that the national energy usage of the servers and data centers in 2006 is more than doubled compared to the electricity consumed in 2000 (U.S. Environmental Protection Agency, 2007). 1

1·2 The distributions of the energy consumption in HPC data centers (Rajic, 2009). 2

1·3 An illustration of a 3D many-core processor with stacked DRAM. TSVs are used to connect the on-chip DRAM layer with the logic layer. . . 4

2·1 (a) 3D test vehicle and (b) TSV fabricated by EPFL (Atienza, 2010). 11

3·1 An illustration of our simulation framework for jointly analyzing performance, power, and temperature tradeoffs of 3D stacked systems. . 21

3·2 An illustration of a generic 3D 16-core processor with 2-layer on-chip DRAM stacking. 22

3·3 The layout for the logic layer of target 3D system. 24

3·4 An illustration of the 3D system with DRAM stacking that has (a) `single-bus regular memory access` and (b) 4-way `parallel memory access`. 26

3·5 Memory request queuing delay in different memory access schemes. Average access rates of 0.0035, 0.012, and 0.025 are obtained by simulating single-bus, 4-way parallel, and 8-way parallel access schemes, respectively. 28
3·6 Average memory accesses per 10ms on different DRAM ranks on 3D system with stacked DRAM. 29
3·7 Percentage of IPS improvement for 3D systems with DRAM stacking over 2D baselines. 36
3·8 IPS temporal behavior analysis of streamcluster running on 3D systems with DRAM stacking versus running on 2D baseline systems. 37
3·9 IPS temporal behavior analysis for 2D-baseline versus 3D-DRAM systems for fluidanimate. 37
3·10 Average core power for the 3D `high-performance` system with DRAM stacking and the 2D baseline. 38
3·11 Average core power for the 3D `low-power` system with DRAM stacking and the 2D baseline. 39
3·12 Peak chip temperatures for the 2D-baseline and the 3D stacked DRAM systems. 39

4·1 IPS for PARSEC and NAS benchmarks running on 2D baseline and the 3D system with parallel access. 44
4·2 Peak chip temperature on the 3D system with parallel access running at different V-F settings. 45
4·3 Total chip power on the 3D system with parallel access running at different V-F settings. 46
4·4 The flowchart of our runtime optimization policy. 47

4·5 Performance improvement on 3D system with parallel access compared to 3D system with regular access. 48
4·6 3D system using our runtime management policy in comparison to running all benchmarks at the static V-F setting of 0.8GHz/0.98V. . 50
4·7 Temperature trace of *ua* on the 3D system running at 1.4GHz/1.02V and the V-F setting selected by our runtime management policy. . . . 51
4·8 3D system with runtime management policy in comparison to temperature-triggered DVFS policy. 51
4·9 Prediction accuracy of our runtime management policy versus memory access (MA) driven DVFS. 52
4·10 The DRAM layer layout for the `high-performance` 3D system with on-chip DRAM. 54
4·11 DRAM bank temperature and access rate for `streamcluster` in 3D `high-performance` system with 128-Byte memory bus. 55
4·12 An illustration of 3D many-core systems with two logic layers stacking and off-chip DRAM. 56
4·13 Peak and average temperatures for 64-core 2D system and 3D system with two logic layers, including the results with no thermal management (No DTM), with temperature-aware load balancing (TALB), and with TALB combined with DVFS (TALB+DVFS). 57
4·14 Peak and average temperature between liquid-cooled 3D systems and the 2D air-cooled baseline, including the temperatures results with no thermal management (No DTM), with temperature-aware load balancing (TALB), and with TALB combined with DVFS (TALB+DVFS), as well as fuzzy controller combined with TALB (Fuzzy+TALB). . . . 59

5·1 Layout of the target data center. 62

5·2 Communication pattern and distance measure. 64
5·3 Cross-interference coefficient matrix for our system. 67
5·4 Exit and recirculation coefficients for our system. 69
5·5 Layout of the Intel Clovertown System (Teng et al., 2009). 78
5·6 System reliability for different series-parallel scenarios with T_H=80°C and per-core MTTF of 5 years at 60°C. 79
5·7 System reliability improvement of clustering over balancing $((R_{clustered} - R_{balanced})/R_{balanced})$ for various T_H, ΔT (T_H-T_L). 80
5·8 System reliability for a *processor-level parallel* system with initial per-core MTTF values of 3, 5 and 7 years. 81

6·1 A flow chart for illustrating the *GCLB* reliability optimization policy for *processor-level parallel* systems. 85
6·2 An illustration of the clustering and balancing job allocations on the target system under 75% utilization. P represents power consumption, and $P1 > P2 > ... > P6$. 86
6·3 System reliability with *GCLB* and thermal balancing allocation policies for the target system under 75% utilization. 92
6·4 System reliability with *GCLB* and thermal balancing allocation policies for the target system under 50% utilization. 92
6·5 System reliability for *GCLB* optimization policy compared to thermal balancing for systems with 75% utilization, considering a thermal constraint of 75^oC. 93
6·6 Temporal workload utilization for the target system. 94
6·7 System reliability of *GCLB* compared to random job allocation for dynamically changing workload utilization. 94

6·8 Exploration of 16-core system reliability with *GCLB* and thermal balancing allocation policies under 75% utilization. 95
6·9 Joint optimization algorithm. 101
6·10 Allocation scheme for the three policies. 103
6·11 Job sizes and percentage of active nodes for multiple-row allocation. . 104
6·12 Cooling power for multiple-row allocation. 105
6·13 Individual job communication costs for multiple-row allocation. . . . 105
6·14 Percentage of the active nodes and the cooling power traces for dynamic allocation. 107
6·15 Histogram of the communication cost for the dynamic allocation experiment. 109

7·1 IPC of SPEC benchmarks for increasing L2 cache size. The IPC values are normalized with respect to using a 256KB L2 cache. 117
7·2 EDP and EDAP of the 3D system with cache resource pooling and its 3D baseline with 1MB static caches, normalized to its 2MB baseline. 118
7·3 Percentage reduction in job communication cost using RCB-based task mapping for the dynamic allocation scenario. 123

Chapter 1

Introduction

Energy efficiency is an increasingly important concern in computing system design. The rapid growth of data-intensive computing has led to larger demands for computing facilities and higher amounts of electricity to power them. As shown in Figure 1·1, the energy used by data center servers and their supporting cooling infrastructures has doubled between 2000 and 2006, and this trend is expected to continue (U.S. Environmental Protection Agency, 2007). In fact, the cooling subsystems are responsible for close to half of the computing energy expenses in today's high-performance

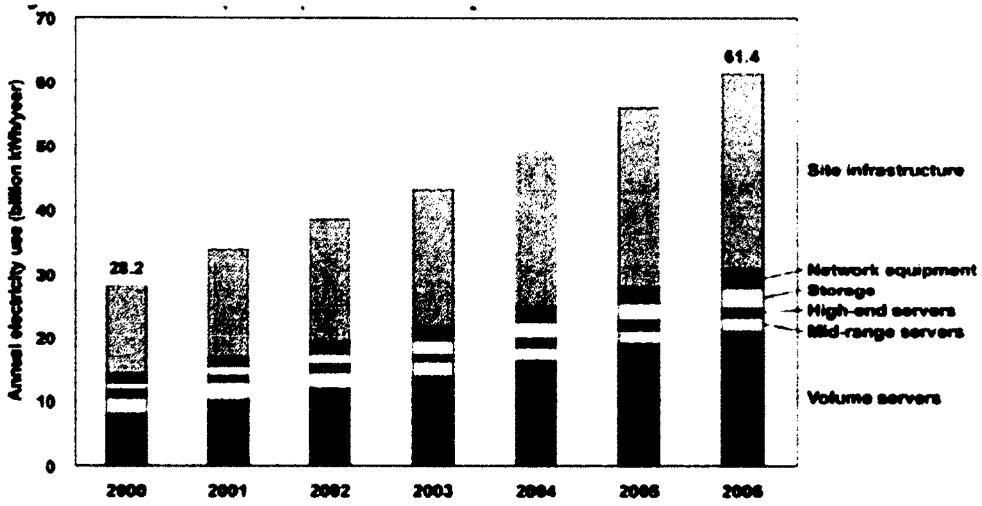


Figure 1·1: The report from U.S. environmental protection agency to the Congress on server and data center energy efficiency shows that the national energy usage of the servers and data centers in 2006 is more than doubled compared to the electricity consumed in 2000 (U.S. Environmental Protection Agency, 2007).

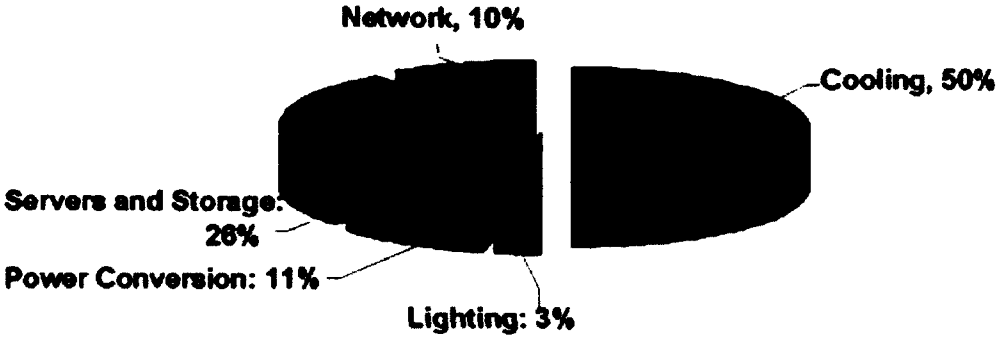


Figure 1·2: The distributions of the energy consumption in HPC data centers (Rajic, 2009).

computing (HPC) clusters and data centers, as presented in Figure 1·2 (Rajic, 2009; Brown and Reams, 2010). The increased energy consumption in data centers also has negative implications on system reliability, complexity, and scalability (Stavros Harizopoulos, 2009; Coskun et al., 2009b). Therefore, in computing system design area, it is important to develop advanced design techniques for energy-efficient computing.

Many-core systems have become the main trend in computing system design owning to their potential of providing higher energy efficiency in comparison to single-core computing systems (Kongetira et al., 2005). Today's many-core systems appear in a number of computing domains ranging from small-scale many-core processors to large-scale HPC data centers. The workloads in these domains involve a large variety of applications, such as scientific computing, modeling, and financial applications. These applications differ in their performance characteristics, such as instructions per cycle (IPC), memory access trends, and communication intensities. Therefore, the workload characteristics of many-core systems are expected to considerably vary within or across applications during the system's lifetime. For future many-core systems that are expected to run such dynamically changing workloads, novel modeling and management approaches are required in order to achieve significant energy efficiency improvements.

This thesis focuses on developing the modeling and runtime management techniques that evaluate and optimize the energy efficiency and reliability of many-core systems. Our goal is to find solutions that recognize the dynamically changing workload characteristics and understand the complex interplay among performance, energy, and temperature for both single-chip 3D many-core processors and for HPC data centers that consist of thousands of processors.

1.1 Problem Statement

Today, performance, energy, temperature, and reliability have become the main challenges in computing system design. In many-core systems, performance is the first-order constraint. Although the performance of computing systems has increased tremendously in the last decade, the demand for higher performance is still there and will not disappear in the near future. Following the higher performance demand in many-core systems, the computing power increases and causes higher on-chip power densities. The increase in power densities results in higher on-chip temperatures and large thermal variations, and creates thermal hot spots. The elevated peak temperatures and thermal variations accelerate the failure mechanisms, degrade system reliability, and also cause higher cooling cost (Stavros Harizopoulos, 2009; JEDEC, 2006; Coskun et al., 2009b).

In order to address these challenges, this thesis focuses on two important domains in many-core systems that are expected to dominate the future computing system design trend: one domain is the many-core single-chip processor and the other is the HPC data center that includes thousands of processors.

For many-core single chip processors, the performance of conventional 2D processors is limited by the large latency between last-level caches and main memory. 3D stacked design, where multiple chips are vertically connected, has emerged as a promising solution to overcome this performance bottleneck. Figure 1·3 provides an example of a 3D many-core processor with a stacked DRAM layer. TSVs are used to connect the on-chip DRAM layer with the logic layer in the 3D processor. Such 3D stacked architecture enables significant improvement in system energy efficiency because of the high-bandwidth connections between the memory and logic layers provided by TSVs. At the same time, 3D design improves per-chip transistor density without requiring aggressive technology scaling, enhances manufacturing yield by vertically stacking smaller chips in comparison to building large single-layer chips, and enables heterogeneous integration of different technologies, such as logic layers, DRAM layers, and analog/RF layers (Black et al., 2006; Loh, 2008).

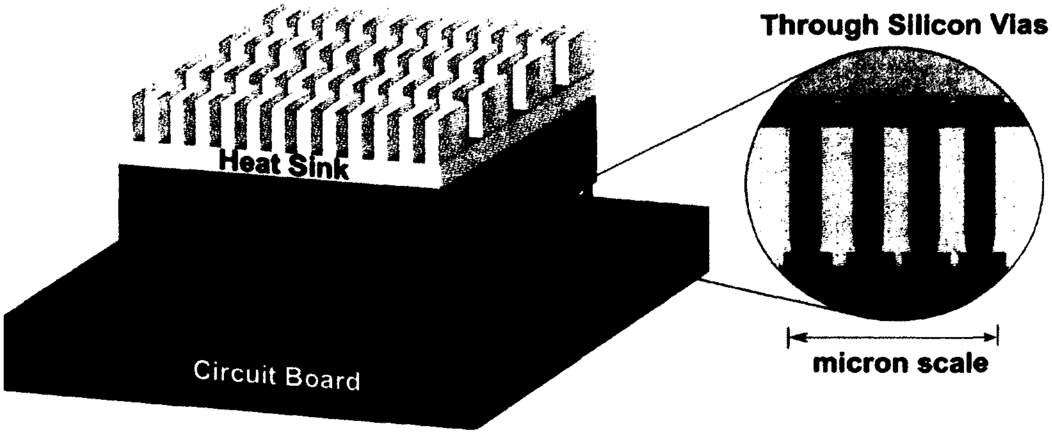


Figure 1·3: An illustration of a 3D many-core processor with stacked DRAM. TSVs are used to connect the on-chip DRAM layer with the logic layer.

However, using 3D stacked systems to achieve the energy efficiency goal brings new challenges in architecture design, manufacturing, testing, runtime operations, and system reliability. Thermal challenges are among the major concerns in building energy-efficient and reliable 3D many-core systems (Liu et al., 2005; Loi et al., 2006; Coskun et al., 2010). Existing temperature management methods for 3D systems include thermally-aware floorplanning, temperature-aware job allocation, and dynamic voltage-frequency scaling (DVFS) (Puttaswamy and Loh, 2007; Cong et al., 2007; Zhu et al., 2008). However, the energy and thermal management approaches for 3D systems have been mostly disjoint from detailed performance and power evaluation. In addition, performance evaluation for 3D systems has mainly focused on a small number of cores (e.g., single-core, quad-core) running single-threaded workloads (Liu et al., 2005; Loi et al., 2006; Loh, 2008). The comprehensive design, evaluation, and runtime management methodologies with a thorough consideration of performance, energy, and temperature tradeoffs in 3D many-core systems are not available.

The energy efficiency and reliability challenges also exist in many-core systems in the HPC data centers. As the number of cores and power density per processor increase, temperature and reliability are becoming significant concerns in data centers as well. High temperatures jeopardize the reliability of the chips and significantly impact performance. In modern processors, temperature and reliability challenges are addressed by management techniques such as clock-gating and DVFS (Hanson et al., 2007; Kang et al., 2010; Coskun et al., 2009a). Temperature-aware workload management approaches have been proposed for both single-core (Hanson et al., 2007; Kumar et al., 2006) and many-core processors (Teodorescu and Torrellas, 2008a; Winter and Albonesi, 2008; Donald and Martonosi, 2006; Coskun et al., 2009c). Among temperature-aware workload management policies, temperature balancing has been shown to be effective at the processor level (Coskun et al., 2009c). The main idea behind thermally-aware workload allocation is to exploit temperature variations resulting from executing jobs with different CPU usage profiles. "Hot" jobs, such as computation-intensive algorithms, cause the chip to run at a higher temperature compared to "cool" jobs. Through intelligent scheduling of such hot and cool jobs, we can reduce thermal hot spots and variations. However, for large-scale many-core systems with multiple chips or multiple servers, where some failures can be tolerated by the inherent redundancy of the system, the reliability impact of thermal balancing has not been studied.

In HPC data centers, high temperatures also result in a large amount of cooling energy consumption. It has been reported that nearly half of the energy in the computing clusters today is consumed by the cooling infrastructure (Rajic, 2009; Brown and Reams, 2010). It is possible to reduce the cooling cost by allowing the data center temperatures to rise; however, component reliability constraints impose thermal thresholds as failure rates are exponentially dependent on the processor temperatures (JEDEC, 2006). One approach to address the cooling energy challenge of HPC data centers is to perform cooling-aware job allocation (Moore et al., 2005; Tang et al., 2008; Pakbaznia and Pedram, 2009).

Another critical aspect in data center management is performance. In HPC clusters, highly parallel scientific, financial, or other applications run on multiple nodes for long durations in the range of minutes, hours or days. The threads of these applications communicate with each other through communication infrastructures such as the message passing interface (MPI). The running time of a communication-intensive application is highly dependent on the location of the individual computing units that are communicating with each other. The communication cost of communication-intensive applications has a significant impact on system performance in HPC data centers (Leung et al., 2002). However, existing job allocation algorithms for HPC data centers address cooling efficiency and performance separately. How to jointly optimize the performance and cooling energy tradeoffs through job allocation in HPC data centers is an open question.

1.2 Thesis Contributions

This thesis contributes to solving the energy and temperature challenges in 3D many-core processors and many-core systems in HPC data centers from both modeling and management aspects.

Our research addresses the performance and temperature bottlenecks of 3D many-core systems by firstly providing a methodology for constructing a comprehensive evaluation framework with detailed modeling of performance, power and temperature. Although thermal modeling (Coskun et al., 2010) and performance (or delay) evaluation approaches exist, they are largely disjoint and typically include coarse-grained assumptions about one another. Our research aims at integrating detailed performance simulation with power and thermal evaluation models in order to enable realistic evaluation of real-world multi-threaded applications running on 3D many-core systems. To the best of our knowledge, our work is the first to jointly analyze performance, power, and temperature of both DRAM and processor layers of 3D many-core processors through architecture-level evaluations.

Utilizing the detailed analysis enabled by our simulation framework, we are able to design and evaluate runtime management and optimization policies for improving the energy efficiency and reliability of 3D many-core systems. In order to exploit the performance potential of 3D processors with DRAM stacking while maintaining the peak power and temperature constraints, we propose a runtime optimization policy that dynamically monitors workload behavior and selects among low-power and *turbo* (high-performance) operating modes in an application-aware manner. Leveraging the detailed modeling and analysis of on-chip DRAM layers, we also introduce a memory management policy that targets applications with spatial variations in DRAM accesses, and performs temperature-aware mapping of virtual memory accesses to physical DRAM banks.

For many-core servers in HPC data centers, reliability has become a serious concern as HPC moves towards exascale. In this thesis, we use a detailed temperature-dependent reliability modeling approach to demonstrate that for systems with multiple chips, *clustering* jobs with higher power consumption may result in higher system reliability compared to aggressively *balancing* the temperature. Following an analysis of the tradeoffs between load *balancing* and *clustering*, we propose a novel policy that optimizes system reliability by choosing between *clustering* and *balancing* at runtime according to the system topology.

At the data center level, an important distinguishing aspect compared to processor or server-level modeling and optimization is the need to consider the data center cooling cost. Following the observation that existing HPC job allocation algorithms address cooling and communication delay optimizations separately, in this thesis, we propose a joint optimization policy that reduces both cooling power and communication latency in an HPC data center.

The specific contributions of this thesis are as follows:

- A simulation framework for 3D systems with on-chip DRAM. Our work is the first to jointly analyze performance, power, and thermal characteristics at the architecture level for both DRAM and processor layers.

- Runtime optimization and management of 3D systems with DRAM stacking. We propose a novel runtime optimization policy that maximizes the system performance by characterizing the application behavior and predicting the operating points that satisfy the power and thermal constraints. Our experiments demonstrate that our policy achieves an EDP reduction of up to 61.9% for a 16-core 3D processor with stacked DRAM compared to a 3D system managed by a temperature-triggered DVFS policy.

- Reliability analysis of multi-chip many-core systems. Using a reliability modeling approach to accurately model temperature-induced wear-out failure mechanisms under various system reliability configurations (i.e., topologies), we quantify the tradeoffs between *clustering* higher power jobs and *thermal balancing* at various operating temperatures.
- Design of a job allocation policy that is aware of the reliability topology to optimize the system reliability. We design light-weight predictors to estimate application power and chip peak temperature during allocation. Our policy adapts to workload changes while respecting the thermal constraints. Experimental results show that our policy improves the system reliability by up to 123.3% compared to existing temperature balancing approaches.
- A job allocation technique that jointly optimizes the communication cost of HPC applications and the cooling energy in a data center. We design an optimization algorithm that selects the cooling-efficient locations while allocating jobs and, at the same time, minimizes the distances among the communicating nodes. Our policy reduces the cooling power by 40% on average compared to cooling-aware and performance-aware policies, while achieving comparable performance to performance-aware policy.

The rest of the thesis starts with a discussion of the background and related work in Chapter 2. Chapter 3 introduces the methodology for constructing a comprehensive simulation framework for jointly investigating the tradeoffs among the performance, power, and temperature of 3D systems. Chapter 4 discusses our research on developing the optimization and management strategies for 3D stacked systems with on-chip DRAM using the integrated simulation framework. Chapter 5 provides the performance, thermal and reliability models for data centers to evaluate the communication cost, cooling energy and reliability. Chapter 6 introduces our runtime reliability optimization for multi-chip servers and our joint optimization of cooling cost and communication cost of many-core systems in HPC data centers. Chapter 7 summarizes the thesis and also discusses our future work directions and open research problems.

Chapter 2

Background and Related Work

2.1 Background

Reducing energy consumption of computing systems is a challenging problem today. Energy spent on computing has considerably grown in the last decade. It is reported that the energy used by data centers and their supporting cooling infrastructures has doubled between 2000 and 2006 (U.S. Environmental Protection Agency, 2007). The computing energy consumption today surpasses 3% of total US electricity use and increases by 15% every year (Brown and Reams, 2010; Koomey, 2008). In addition, the side effects of high energy use have important global environmental consequences such as the emission of greenhouse gases, resulting in global warming. High energy consumption also has implications for system reliability and scalability. The increased power densities result in elevated on-chip temperatures and large thermal variations, both of which degrade system reliability and increase system design complexity (Coskun et al., 2009b; Srinivasan et al., 2004b).

In the last decade, we have witnessed significant developments in computing hardware design for chip-level energy and thermal management. State-of-the-art techniques typically focus on turning off or slowing down under-utilized resources (e.g., (Hanson et al., 2007; Kang et al., 2010)). A number of techniques have been introduced to predict the idle time slots of cores and other resources to minimize the performance overhead of going in and out of low-power operating modes (Benini et al., 2000; Donald and Martonosi, 2006). Dynamic Voltage and Frequency Scaling

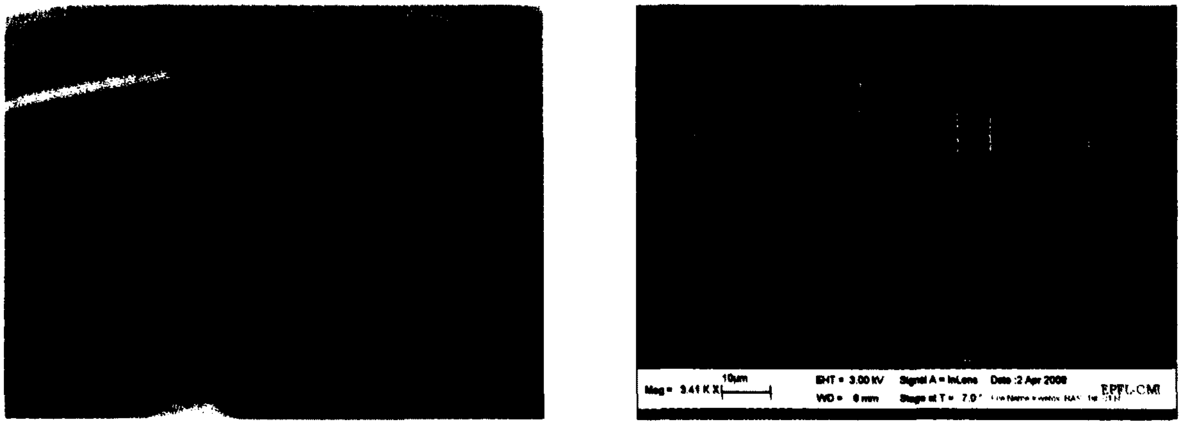
(DVFS) is another commonly used technique (Skadron et al., 2003), and has been adopted in recent many-core chip design (Howard et al., 2010). Recent research has also proposed runtime job scheduling and dynamic power management approaches, such as variation-aware application scheduling and system-level power optimization policies (Teodorescu and Torrellas, 2008b; Isci et al., 2006b), to improve energy efficiency. In addition, system-level approaches, such as temperature-aware scheduling (Coskun et al., 2008; Coskun et al., 2009b) or energy-aware consolidation in virtualized environments (Dhiman et al., 2010), are able to improve energy efficiency considerably.

As future systems are expected to run more performance demanding workloads, novel design approaches are required in order to achieve significant energy efficiency improvements. In this thesis, we focus on developing novel energy- and temperature-aware runtime management and optimization techniques, which dynamically recognize the hardware-software characteristics and understand the complex interplay among performance, energy, and temperature.

2.2 Modeling and Management of 3D Many-core Systems

3D stacking has emerged as an attractive design technique to improve manufacturing yield, transistor density per chip footprint, and performance (Black et al., 2006). The initial work on 3D integration includes the concept of through silicon via (TSV) based chip stacking and integration technology (Koyanagi et al., 1998; Topaloglu, 2011). 3D integration technology can usually be classified as monolithic or stacking-based. Monolithic 3D integration builds multiple active device layers on a single wafer, while 3D stacking approach involves manufacturing of each layer separately using conventional fabrication techniques. These layers are later stacked using solder bumps. Thus, 3D stacking is more practical and becomes the focus in most of the recent 3D integration research (Golshani et al., 2010; Black et al., 2006; Liu et al., 2005). 3D stacking process could be categorized as wafer-to-wafer, die-to-wafer, or die-to-die stacking. Wafer-to-wafer stacking maximizes the throughput and minimizes the manufacturing cost, while die-to-wafer or die-to-die stacking is the only option when die sizes are not matched. In 3D stacking, multiple layers are assembled using bonding technologies, such as wire, micro-bump, or TSV based bonding. Comparing to wire or micro-bump bonding, TSV based 3D integration has the potential to offer the greatest vertical interconnect density, and therefore is the most promising vertical integration technology (Ferri et al., 2008; Khan et al., 2011; Dong et al., 2010). Figure 2·1 shows the magnified images of a five-layer 3D stacked chip, which is wire-bonded on the side (without TSVs), and TSV fabricated by EPFL (Atienza, 2010).

One of the prominent advantages of 3D stacking is the ability to integrate heterogeneous technologies within the same chip, such as stacking memory layers with the processors. Designing 3D systems with on-chip DRAM is a promising solution to improve memory bandwidth and reduce memory access latency (Black et al., 2006; Loh, 2008). Reducing the memory access overhead is especially beneficial for many-



(a) (b)

Figure 2·1: (a) 3D test vehicle and (b) TSV fabricated by EPFL (Atienza, 2010).

core systems, where long off-chip memory latency has been a gating performance bottleneck. However, power densities and temperatures also increase following the performance improvement. In fact, high temperatures already bring major challenges because of their adverse effects on cooling costs and reliability (Puttaswamy and Loh, 2007; Coskun et al., 2010; Srinivasan et al., 2004b).

Prior work on the modeling of 3D systems with memory stacking mostly considers performance, power, and thermal evaluations separately, focusing on the systems with a small number of cores or single-threaded workloads. For example, Liu et al. report that a single-core processor with 3D memory stacking increases system performance by 126%; however their work does not consider the power or thermal impact (Liu et al., 2005). Loh explores 3D-stacked memory architectures for 4-core processors (Loh, 2008) with a thermal analysis using HotSpot (Skadron et al., 2003). Their thermal simulations use estimated power values that are not tied with detailed architecture-level performance analysis. Sun et al. study the architecture-level design of 3D stacked L2 cache, without extending the power and thermal analysis for 3D stacked memory (Sun et al., 2009). Wu et al. provide the power density analysis and power delivery consideration in a formulation of 3D processor cost model to estimate the impact of power delivery on manufacturing cost (Wu et al., 2010). However, they do not evaluate the power consumption of the memory components on the 3D chips.

The recent research on 3D system energy and thermal management includes design-time optimization methods and runtime management polices based on task scheduling and DVFS techniques. For design-time optimization methods, Cong et al. propose transformation techniques for 3D IC placement (Cong et al., 2007). Hung et al. present a thermally-aware floorplanner for 3D architectures (Hung et al., 2006). Healy et al. propose a microarchitectural floorplanning algorithm for 3D ICs using linear programming and simulated annealing (Healy et al., 2007). Their static optimization methods are implemented at design stage, and do not address dynamic changes in workload profiles.

Dynamic power management on traditional multi-core (2D) systems has been well studied, and a number of such techniques can be extended to 3D systems as well. Isci et al. present a runtime phase prediction methodology to control DVFS based on frequency of memory operations (Isci et al., 2006a). Cochran et al. propose a scalable method for determining the optimal V-F settings under power constraints (Cochran et al., 2011). Recently proposed dynamic energy and temperature management methods for 3D systems include runtime workload scheduling, dynamic voltage-frequency scaling (DVFS), and temperature-aware job allocation. Zhu et al. propose a runtime thermal management approach using task migration and DVFS (Zhu et al., 2008). Zhou et al. introduce an OS-level scheduling algorithm for optimizing 3D system temperature using dynamic workload scheduling (Zhou et al., 2008). These methods that explicitly target 3D systems, however, do not perform a detailed performance analysis of the applications. Also, detailed performance analysis and thermal optimization for 3D systems have been mostly disjoint so far. For example, thermal management policies focusing on 3D systems provide performance estimates based on worst-case scenarios, without providing an architecture-level evaluation (Coskun et al., 2010).

2.3 Energy and Reliability Management in Servers and Data Centers

A number of approaches on reliability management focus on microarchitectural optimization (Srinivasan et al., 2004a; Biswas et al., 2011). Recent work has also introduced reliability management techniques specifically targeting many-core systems. Hanumaiah et al. optimize the reliability of a many-core processor running tasks with hard deadline constraints by solving a quasi-convex optimization problem (Hanumaiah and Vrudhula, 2011). Wang et al. maximize the lifetime of many-core systems while maintaining a given aggregate processor speed by applying sequential quadratic programming (Wang and Chen, 2010). Coskun et al. propose a simulation framework to evaluate the impact of management policies on processor lifetime and demonstrate the benefits of temperature balancing (Coskun et al., 2009c). Bose et al. integrate the modeling of wear-out failure mechanisms into a power-performance simulator to project failure rates and consequent system lifetime (Bose et al., 2010).

Several reliability management techniques consider both the wear-out mechanisms and the system topology. Huang et al. (Huang et al., 2009) use the Weibull distribution to model aging effects. *RAMP* uses Monte Carlo simulations and lognormal distributions to compute reliability, and a simple MIN-MAX approach to model series-parallel topologies (Srinivasan et al., 2005). Reliability of a computer system with series-parallel components can also be computed using probabilistic models that takes the inherent redundancy of the system into consideration (Coskun et al., 2006).

Recent research has also introduced temperature-aware job allocation policies. Moore et al. develop a temperature-aware workload placement algorithm through establishing a prioritized list of servers for saving energy in data centers (Moore et al., 2005). Coskun et al. design adaptive scheduling policies that leverage thermal sensor readings for reducing temporal and spatial temperature variations on multi-core processors (Coskun et al., 2008). Wang et al. propose a thermally-aware job scheduling algorithm for data centers to allocate workloads based on their task-temperature profiles (Wang et al., 2009). However, these policies do not consider the impact of system topology on system reliability during job allocation.

Performance has been the main goal of job allocation techniques in data centers and supercomputers. Performance-aware job allocation algorithms typically focus on minimizing the average number of communication hops between processors on which a job is running. Bhattacharya et al. propose a heuristic for job allocation in a mesh-connected parallel processor (Bhattacharya and Tsai, 1994). They use a look-ahead mechanism that looks into the queue of waiting jobs and selects free processors from the sub-meshes in a mesh-connected data center to allocate the jobs. Mache et al. present the MC allocation strategy for mesh-connected parallel computers. Their method yields compact allocations by containing the jobs in the smallest rectangular area possible (Mache et al., 1997).

Bender et al. propose an MC1x1 processor-allocation algorithm, in which the first sub-mesh is a 1X1 shell and subsequent sub-meshes grow in square shapes until finding enough available nodes to allocate the upcoming job (Bender et al., 2008). However, existing performance-aware job allocation strategies solely target the performance and communication costs without considering the potential impact of job allocation on the power, temperature, or the cooling costs.

As thermal management and reducing the cooling costs are among the dominant concerns for today's data centers, a number of thermal modeling and management techniques at data center level have been proposed recently. Jungsoo et al. use a linear formula that computes server temperatures as a function of ambient room temperature, thermal resistance between die and air, and server power (Kim et al., 2012). However, their model does not consider the effect of recirculation on temperature. Moore et al. carry out computational fluid dynamics (CFD) simulations to conduct thermal evaluation (Moore et al., 2005). However, CFD simulation is expensive and cannot be used for real-time data center thermal management. Heath et al. introduce a data center temperature emulation suite called Mercury that emulates temperatures based on the data center layout, hardware, and component utilizations (Heath et al., 2006). Despite its efficiency advantages, Mercury has not been validated for large data center systems. Tang et al. propose a linear model to compute data center temperatures and cooling energy costs, and solve an optimization problem for minimizing the peak node inlet temperature (MPIT) through job assignment (Tang et al., 2008). They use both genetic algorithms and sequential quadratic programming to solve the problem. However, their main focus is enterprise/transactional workloads with independent tasks on different data center nodes, so their model does not include the communication latency during allocation.

2.4 Distinguishing Aspects from Prior Work

Our work improves upon the state-of-the-art for the modeling and management of 3D many-core processors and HPC data centers in the following aspects:

- Introduces a widely applicable and generalizable methodology for accurately and jointly analyzing the performance, energy, and temperature characteristics of 3D many-core systems, while most prior research in 3D area targets a specific architecture or only one of these three aspects.

- Addresses the unique challenges for parallel applications representing future computing workloads running on many-core systems, instead of focusing solely on conventional single-threaded applications. With such parallel programs that push existing processor designs to their limit, our work is able to drive the design and analysis of the new generation computing systems.

- Delivers a set of energy and thermal management policies that are aware of the workload properties and the 3D architectural features governing the system performance. Such temperature-aware policies enable us to push the performance bounds of 3D systems dramatically compared to current chips while maintaining reliable and low-energy operation.

- Proposes a workload allocation policy to optimize the system reliability for multi-chip servers. Most temperature-aware job allocation methods make use of *temperature balancing.* Following our analysis that shows *clustering* may provide better reliability than *balancing* depending on the system reliability topology, we propose a job allocation method that selects between workload *balancing* and *clustering* depending on the system topology to optimize reliability for multi-chip many-core systems.

- Designs a job allocation policy that optimizes both the application performance (in terms of the communication cost) and the cooling energy cost of HPC data centers under reliability constraints. Prior work has addressed performance, reliability, and cooling cost optimizations as separate problems. Our policy confines the communicating nodes of a job in close proximity, but it also selects the most cooling-efficient locations possible.

Chapter 3

Modeling of 3D Many-core Systems

3.1 Overview

This chapter presents our research on constructing a comprehensive simulation framework to address the complex interplay between performance, energy, and temperature of 3D systems. The goal is to achieve an accurate and thoroughgoing exploration of both the merits and challenges of 3D stacked systems. Our research focuses on 3D systems with DRAM stacking, because stacking the main memory on the chip reduces the off-chip memory access delays, and thus, has the potential for significantly increasing the system performance and energy efficiency.

3D many-core processors bring us both merits and challenges. On one hand, 3D systems offer promising performance improvement owing to the opportunities of heterogeneous integration, building of large many-core chips with high yield, and shorter global wire lengths. On the other hand, 3D systems exacerbate the already existing thermal challenges because of the higher thermal resistivities for the layers away from the heat sink and higher power densities per chip footprint brought by the increased performance. Thermal hot spots and large temporal and spatial temperature variations adversely affect system energy efficiency and reliability. In 3D systems with on-chip DRAM, the power and temperature of the DRAM layers also substantially increase because of the high memory access rate and the heat transfer from the logic layer, while high DRAM temperatures severely affect memory reliability and system performance (Ghosh and Lee, 2007; Liu et al., 2011).

Prior work usually conducts disjointed simulations for the performance, power, and temperature of 3D many-core systems or uses coarse-grained estimations based on analytical models (Loi et al., 2006; Loh, 2008). The existing energy and thermal management policies for 3D systems have been mostly derived indirectly from detailed performance and power evaluations. For example, recently published management policies for 3D systems provide worst-case performance estimates without providing an architecture-level performance simulation (Coskun et al., 2010). A similar problem exists in the previously proposed techniques on optimizing 3D DRAM organization, which do not provide detailed DRAM power and thermal evaluations connected with detailed performance simulations of the 3D many-core systems (Loh, 2008; Ghosh and Lee, 2007).

Our research on constructing the simulation framework is the first to jointly analyze performance, power, and temperature tradeoffs for both DRAM and processor layers in the 3D stacked systems. It is an essential step for conducting an accurate investigation of 3D system energy and temperature characteristics, for optimizing the energy efficiency and reliability of future 3D many-core systems, and for providing better understanding of the benefits and limitations of 3D memory stacking.

As illustrated in Figure 3·1, our simulation framework consists of the modeling of target systems, performance simulation, power modeling, and temperature modeling. We first model the logic layer and DRAM layer of our target 3D systems, including abstracting the memory access and bus latencies. The system configuration parameters and floorplans are used as inputs for performance simulation, power model, and temperature model of the simulation framework. Then, we run performance simulations on an architecture-level full-system simulator, such as M5 (Binkert et al., 2006), to collect detailed performance statistics. In the M5 simulator, we model 3D systems with on-chip DRAM by configuring the main memory access latency and bus width

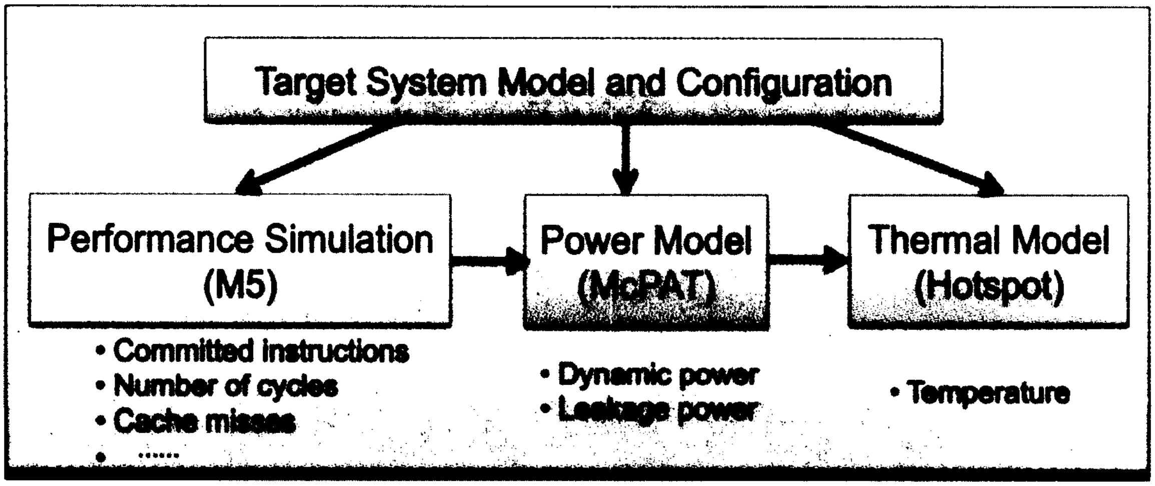


Figure 3·1: An illustration of our simulation framework for jointly analyzing performance, power, and temperature tradeoffs of 3D stacked systems.

to mimic the high data transfer bandwidth provided by the TSVs. The performance results are fed into a power model, such as McPAT (Li et al., 2009), for estimating the core power. The McPAT results are calibrated to match the published or measured power of target architectures for improving their accuracy. We also utilize a cache power model, such as CACTI (Thoziyoor et al., 2008), and the DRAM power calculator from MICRON. The power traces as then used as inputs in the thermal model, such as HotSpot (Skadron et al., 2003), to simulate the temperatures of both the logic and DRAM layers of 3D systems.

In this chapter, we introduce the methodology of modeling the target 3D many-core systems, performance simulation, as well as the power and thermal models. We present the evaluation results on the performance, power, and temperature for both `high-performance` and `low-power` 3D systems running parallel workloads by utilizing our integrated simulation framework.

3.2 Methodology for Modeling 3D Many-core Systems

This section presents the modeling of 3D systems with on-chip DRAM, performance simulation infrastructure, power model, and thermal model that are utilized in our research for constructing the simulation framework as introduced in Section 3.1. Our modeling methodology considers performance, power, and temperature simulations jointly, enabling a more accurate evaluation in comparison to the modeling methods introduced in prior work (Coskun et al., 2010).

3.2.1 Modeling Target 3D Systems with DRAM Stacking

Our research targets 3D many-core systems with stacked on-chip DRAM, as they provide high speed and wide bandwidth for accessing main memory by utilizing the vertical TSVs. Figure 3·2 provides an illustration of a 16-core 3D system with DRAM stacking. In this 3D system, the processing cores and caches are on one layer and a 2-layer 3D DRAM is stacked below the logic layer. TSVs are used for vertically connecting the core and DRAM layers. We model our 3D stacked architectures with two types of cores: a `high-performance` core and a `low-power` core.

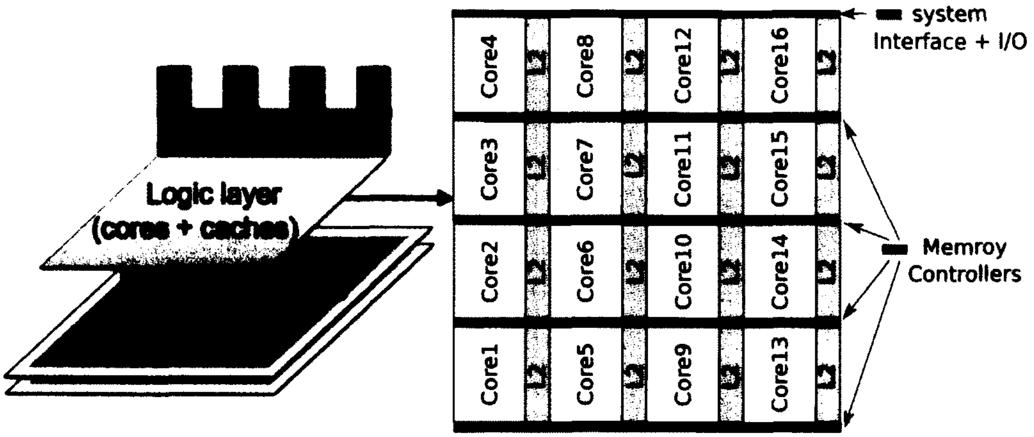


Figure 3·2: An illustration of a generic 3D 16-core processor with 2-layer on-chip DRAM stacking.

Table 3.1: 3D system core architecture parameters.

Parameter	`High-performance`	`Low-power`
CPU Clock	2.1GHz	1.0 GHz
Issue	out-of-order	out-of-order
Decode Width	3-way	2-way
Reorder Buffer	84 entries	40 entries
BTB size	2048 entries	512 entries
RAS size	24 entries	16 entries
Integer/FP ALU	3/3	2/1
Load Queue	32 entries	16 entries
Store Queue	32 entries	12 entries
L1 ICache	64KB@2ns	16KB@2ns
L1 DCache	2-way	2-way
	64B-block	64B-block
L2 Cache	512KB@6ns	512KB@5ns
	16-way	4-way
	64B-block	64B-block

The architecture for the `low-power` core is similar to the architecture of the cores used in the Intel single-chip cloud computer (SCC) (Howard et al., 2010). The `high-performance` system includes more aggressive core architectures, which are modeled based on the AMD Family 10h microarchitecture of the cores in the AMD Magny Cours processor. We simulate both the 2D baselines (single-layer, off-chip memory) and 3D systems with on-chip DRAM for the two target architectures. The architectural parameters for the cores and the caches are listed in Table 3.1.

For each processor, we use the same architectural configuration for the 2D baseline and the 3D systems (i.e., the only difference is in the latency and bandwidth to the DRAM). Each core on the 16-core processors has multiple-issue and out-of-order execution. We assume both processors are manufactured at 45nm and have a supply

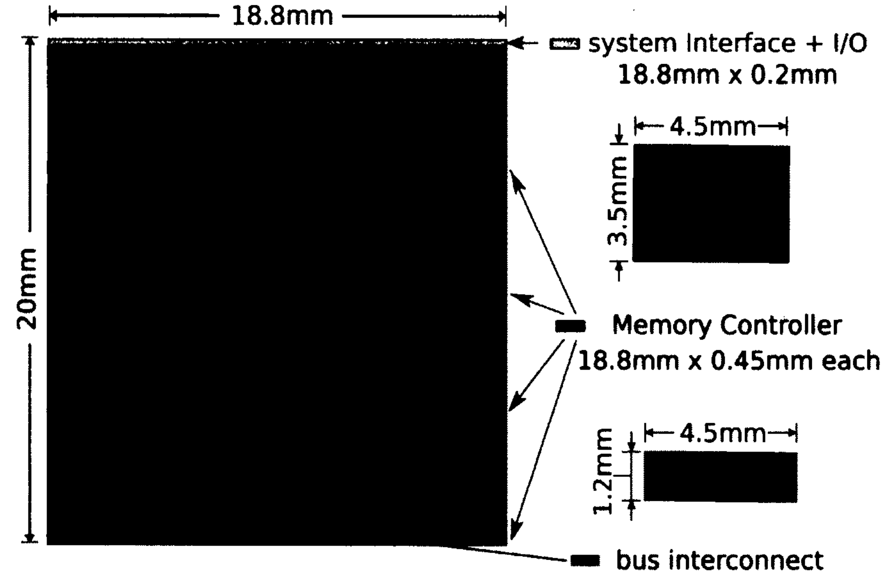


Figure 3·3: The layout for the logic layer of target 3D system.

voltage of 1.14V at the highest available frequency setting. The `high-performance` core has a larger number of integer and floating point arithmetic logic units as well as larger L1 level instruction and data caches in comparison to the `low-power` core.

Figure 3·3 presents the layout of the logic layer of the `high-performance` 16-core 3D system with stacked DRAM. Each core has private 16 KB L1 instruction and data caches, and a private L2 cache. As shown in Figure 3·3, all the L2 caches are located on the same layer as the cores and connected by a shared bus. MESI cache coherence protocol is used for maintaining the consistency among the caches. The 2D baseline and the 3D systems both have on-chip memory controllers.

The dimensions for the components of the 16-core processors are listed in Table 6.3. The `low-power` system has a total die area of $128.7mm^2$ and operates at 1 GHz, while the `high-performance` system has a total die area of $376mm^2$ and operates at 2.1GHz. We assume face-to-back, wafer-to-wafer bonding for building the 3D systems, as wafer-to-wafer bonding allows for reliably manufacturing larger 3D systems approaching sizes of $20mm \times 20mm$ with the current technology.

Table 3.2: Dimensions of the blocks in the target 3D systems.

(all values in mm except TSVs)	`High-perf.`		`Low-power`	
	Length	Width	Length	Width
Chip	20	18.8	11.7	11
Core	4.5	3.5	2.4	1.625
L2 Cache	4.5	1.2	2.4	1.3
Memory Controller	18.8	0.45	11.7	0.308
DRAM	20	18.8	11.5*	9*
TSVs	diameter $10\mu m$		pitch $20\mu m$	

* *This system includes 2 DRAM layers, while the high-performance system has a single DRAM layer of the same memory capacity.*

3.2.2 Modeling 3D On-chip DRAM Accesses

3D systems with on-chip DRAM provide high speed and wide bandwidth for accessing the main memory by utilizing the vertical TSVs, while the accesses to the off-chip main memory in traditional 2D design are limited by slow off-chip buses.

In order to simulate the data transfer between the logic layer and the on-chip DRAM layer on the 3D many-core systems, we consider **`single-bus regular memory access`** and **`parallel memory access`**, both with a fast memory bus at 2GHz. As illustrated in Figure 3-4, in **`single-bus regular memory access`**, all accesses go through a single bus between the memory controller and DRAM. On the other hand, the **`parallel memory access`** scenario allows the four on-chip memory controllers to access the four DRAM ranks at the same time. In order to implement the **`parallel memory access`** on the 3D processor, we deploy 512 TSVs on each memory controller. These TSVs provide a 64-Byte bus width for each memory controller. In our experiments, we consider TSVs with a diameter of $10\mu m$ and a center-to-center pitch of $20\mu m$. Thus the total TSV area only takes up less than 0.2% of the chip

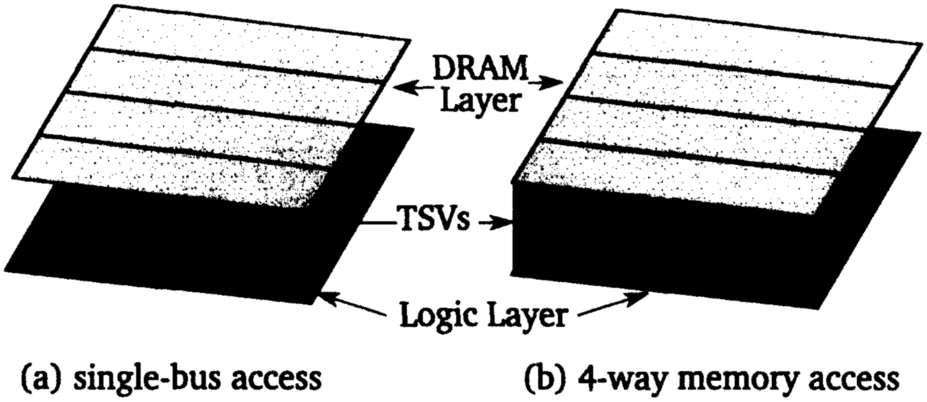


Figure 3·4: An illustration of the 3D system with DRAM stacking that has (a) `single-bus regular memory access` and (b) 4-way `parallel memory access`.

area overhead. The small overhead of TSVs also allows us to implement an 8-way `parallel memory access` scenario with eight on-chip memory controllers accessing eight DRAM ranks at the same time.

In order to quantify the performance improvements of our target 3D systems versus their 2D baselines, we need to have an accurate model of the memory access latency in both cases. We model the memory access latency by examining the different components that contribute to the latency. For many-core systems, there are three main components of the memory access latency from the last-level caches to main memory: the propagation delay between last-level caches to the memory controller (LLC-to-controller delay), the data request time spent at the memory controller (memory controller processing latency), and the data retrieval time spent at the DRAM.

To model the LLC-to-memory controller delay, we assume that all the private L2 caches are connected to the memory controllers through a shared bus. Figure 3·3 illustrates the physical layout of the logic layer, including the shared bus. We assume that the global bus interconnect is routed around the chip in a serpentine fashion. For modeling the bus interconnect, we use energy-optimized repeater-inserted pipelined channels to reduce the global wire delay (Meng et al., 2011). The wire propagation delay is linear with respect to the wire length, owing to the repeaters that are inserted to partition the wire into smaller segments. Each pipeline stage is designed using predictive technology model for 45nm and has a propagation delay of 183ps per mm (Jin et al., 2008). We estimate the average distance from an L2 cache to a memory controller block as 9.4mm based on the layout. Thus, the round trip LLC-to-memory controller latency is 4ns (rounded up).

The memory controller processing latency is strongly governed by the memory request queuing delay (Awasthi et al., 2010). Modern memory controllers typically consist of a memory request queue that buffers the pending requests waiting to get scheduled, and a scheduler that selects the next request to be serviced (Ipek et al., 2008). The memory controller processing latency is dominated by the time spent by a memory request in the request queue waiting to get scheduled. We apply queuing theory to model the memory controller queuing delay, where the memory request queue is modeled as a M/D/N queuing system. In the M/D/N queuing formula, the queuing delay depends on two parameters: arrival rate and service rate. Arrivals are determined by an exponential process, service times are deterministic, and N is the number of memory controllers in the 3D system.

We use the average memory access rate across all the benchmarks as the arrival rate of the memory request queue. We estimate the service rate by considering the DRAM access time (t_{RAS} and t_{RP}) and the parallel memory access in the 3D many-core system. For the target system, we use the row active time $t_{RAS} = 36ns$ and row precharge time $t_{RP} = 15ns$ as reported by MICRON's DDR3 SDRAM. Thus, we model the memory request queue service rate for the 3D many-core system with `single-bus access`, where all accesses go through a single bus between the memory controller and DRAM, as 0.02 per cycle. As parallel access allows memory request

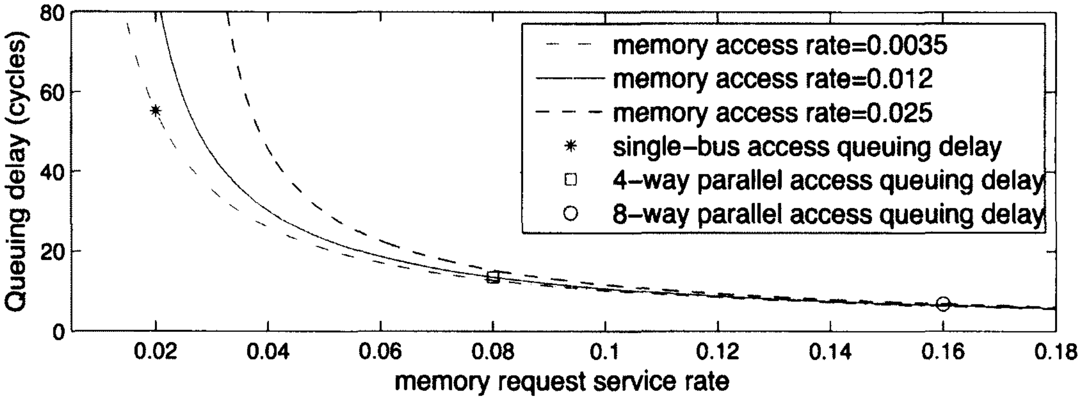


Figure 3·5: Memory request queuing delay in different memory access schemes. Average access rates of 0.0035, 0.012, and 0.025 are obtained by simulating single-bus, 4-way parallel, and 8-way parallel access schemes, respectively.

access multiple DRAM banks at the same time, we assume that the service rate is four times and eight times of the service rate for the `single bus access` for the 3D many-core system with `4-way` and `8-way parallel memory access`, respectively. Figure 3·5 presents the queuing delay of the memory request in the memory controller request queue under different memory access schemes. In Figure 3·5, different curves represent the queuing delay with average access rates of 0.0035, 0.012, and 0.025 that are obtained by simulating `single-bus`, `4-way parallel`, and `8-way parallel access` schemes, respectively. Once the memory controller queuing delay is obtained, we use it to configure the memory access latency in the performance simulator for evaluating the performance of 3D many-core systems with DRAM stacking.

DRAM access latency consists of address decoding time, column and row active time, and data transfer time. Stacking DRAM layers on top of the logic layer makes the data transfer much faster between DRAM and cores. We use the same DRAM parameters for the off-chip DRAM in the 2D baseline and for the DRAM layer in 3D system, which is consistent with the assumptions used in earlier studies (Loh, 2008; Loi et al., 2006). We consider a 1GB DRAM consisting of 4 ranks, each of which

Table 3.3: DRAM access latency.

	2D-baseline design	**3D system with single-bus**
memory controller	4ns LLC-to-controller delay, 48ns MC processing time	4ns LLC-to-controller delay, 24ns MC processing time
main memory	off-chip DRAM $t_{RAS} = 36ns$, $t_{RP} = 15ns$	on-chip DRAM $t_{RAS} = 36ns$, $t_{RP} = 15ns$
total delay	103ns	79ns
memory bus	off-chip bus, 200MHz 8-Byte bus width	on-chip bus, 2GHz 64-Byte bus width

has 4 banks (a total number of 16 DRAM banks). We use the MICRON's row active and row precharge time as discussed above. Table 3.3 summarizes the memory access times for the 2D system and 3D system with `single-bus access`.

From our simulation results for the NAS and PARSEC benchmarks as shown in Figure 3·6, we observe the main memory accesses are evenly distributed between the four ranks. Thus, we assume the memory access latency with `parallel access` is one fourth of the latency with `single-bus regular access`. Note that this is a

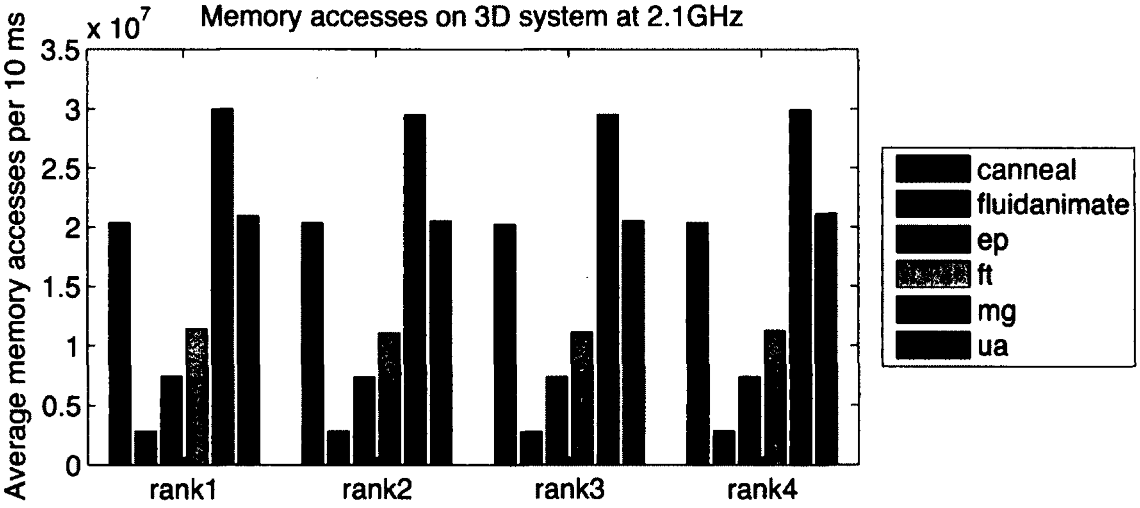


Figure 3·6: Average memory accesses per 10ms on different DRAM ranks on 3D system with stacked DRAM.

conservative assumption as the simultaneous accesses also enable faster processing at the memory controller because of fewer pending requests in the request queues.

3.2.3 Performance Simulation of 3D Many-core Systems

We use the M5 full-system simulator (Binkert et al., 2006) to build the performance simulation infrastructure. We simulate our target system with the Alpha instruction set architecture (ISA) as it is the most stable ISA currently supported in M5. The full-system mode in M5 models a DEC Tsunami system to boot an unmodified Linux 2.6 operating system. We select parallel applications from the PARSEC benchmark suite (Bienia, 2011) and the NAS Parallel Benchmark (NPB) suite (Bailey et al., 1994) as our workloads, both of which represent future multi-threaded workloads and have been widely used in parallel system studies.

M5 models a split-transaction bus that is configurable in both latency and bandwidth. The bus arbitration follows first-come-first-serve logic, and uses round-robin scheduling for bus accesses. We model the 3D system with on-chip DRAM in M5 by configuring the main memory access latency and the bus width between L2 caches and main memory. In this way, based on the methodology provided in Section 3.2.1 and Section 3.2.2, the simulator mimics the high data transfer bandwidth provided by the TSVs. Table 6.1 and Table 6.3 summarize the architecture characteristics, memory access delay, and bus configurations.

We run PARSEC benchmarks in M5 with sim-large input sets and NAS with class B problem sets. For each NAS benchmark, we use a warm-up period of 1 billion instructions to get past the initialization phase. For each PARSEC benchmark, the start of the region-of-interest (ROI, i.e., the parallel phase) is pre-defined in the PARSEC hooks libraries. We fast-forward the M5 simulation to the ROI and execute the instructions in the ROI with the detailed out-of-order CPUs for all the benchmarks. We collect performance statistics from M5 simulations periodically and use them as inputs for our power model.

We implement thread-binding in M5 for the PARSEC and NAS benchmarks to control thread allocation. A thread is bound on a specific core during a time interval and does not move among cores. The default thread-binding policy for is in-order assignment, which means thread i is bounded to core i $(1 \leq i \leq 16)$.

In the 3D system performance simulations, we execute each benchmark in the PARSEC and NAS benchmark suites with the detailed out-of-order CPUs for 1 second, and collect the performance statistics at every 10ms. In order to collect the access statistics for the 3D stacked DRAM, we distinguish between the memory accesses to each DRAM bank by observing the least significant bits for the physical memory addresses. In this way, we track the number of memory accesses to each DRAM bank at every interval.

For evaluating the many-core system throughput, we use instructions retired per second (IPS) as our metric. This metrics is used when comparing the throughput of the 3D systems with on-chip DRAM against their 2D baselines as well as comparing the performance of the `high-performance` system and `low-power` system that are running under different operating frequencies.

3.2.4 Modeling the Power Consumption of 3D Many-core Systems

We use McPAT 0.7 (Li et al., 2009) to estimate the runtime dynamic power of the cores in our target system. McPAT computes the core power consumption by taking the system configuration parameters and M5 performance statistics as inputs. We simulate the dynamic core power for our target 3D systems using McPAT 45nm technology. To improve accuracy for runtime power computations, we calibrate the McPAT runtime dynamic power values for the cores to match the published or measured dynamic core power of the target core architectures.

In order to calibrate the McPAT runtime dynamic core power, we firstly derive the average dynamic core power values from power simulation across the benchmark suite. Then, we compute the calibration factor, R, to translate the McPAT raw data to the target power scale. After that, we use R to scale each benchmark's dynamic core power consumption. A similar calibration approach has been introduced in prior work (Kumar et al., 2003).

Our power model can also estimate the power of systems manufactured using other process technologies. For example, let us assume our target system is manufactured at 22nm and operated at 1GHz, while using the core architecture based on the cores used on Intel SCC (Howard et al., 2010). Since the 48-core Intel SCC processor is designed using 45nm technology, we first need to scale the reported Intel core power to 22nm technology.

The switching power dissipated by a CMOS device is proportional to $C \cdot f \cdot V_{dd}^2$, where C is the load capacitance, f is the operating frequency, and V_{dd} is the supply voltage. We assume that there is negligible change in capacitance. While the V_{dd} dependency of the processor leakage power is exponential, we estimate it as a second order polynomial of V_{dd} around its nominal value since the V_{dd} variation is only around 20% of default setting (Su et al., 2003).

As both our target system and the Intel chip operate at 1GHz, we estimate the processor power of the equivalent 22nm core using Equation (3.1), where the supply voltage for 22nm processor is assumed as 0.9V, and reported average core power and supply voltage for Intel SCC for the 45nm technology are 1.83W and 1.14V, respectively.

$$Power_{22nm} = Power_{45nm} \cdot (\frac{V_{dd_{22nm}}}{V_{dd_{45nm}}})^2. \tag{3.1}$$

L2 cache power is calculated using CACTI 5.3 (Thoziyoor et al., 2008). After we collect the L2 cache read and write access rates from performance simulation results in M5, we use them to scale the read and write power values obtained from CACTI. For the on-chip memory controllers in both of the 3D systems, we estimate the memory controller power consumption as 5.9W based on the memory controller power reported for the Intel SCC (Howard et al., 2010). The system interface and I/O power as well as the on-chip bus power are negligible with respect to the total chip power (Howard et al., 2010).

The DRAM power in the 3D system is calculated using MICRON's DRAM power calculator, which takes the memory read and write access rates as inputs to compute the power for DRAM. We obtain detailed DRAM power traces for each of the DRAM banks sampled every 10ms interval, corresponding to the performance traces collected from M5.

3.2.5 Modeling the Temperature of 3D Many-core Systems

3D systems exacerbate the existing thermal problems in 2D systems because of the higher thermal resistivity of the layers that are away from the heat sink. An accurate thermal model is necessary for evaluating the thermal behavior along with the energy efficiency of our target 3D systems.

We use HotSpot 5.0 (Skadron et al., 2003) for the thermal simulations. We run simulations for both the 2D and 3D systems using the default chip package in HotSpot to represent efficient packages in high-end systems. Calibrated power traces are used as the inputs for the thermal model. The 3D `low-power` system has one logic layer and two DRAM layers, where each DRAM layer having 8 bank components. The 3D `high-performance` system consists of one logic layer and one DRAM layer with 16 bank components.

All simulations use the HotSpot grid model for higher accuracy and are initialized with the steady-state temperatures. The parameters in HotSpot simulations for 2D and 3D architectures are listed in Table 3.4.

In order to model the thermal effect of the TSVs in 3D stacked systems, we extend the default HotSpot by utilizing the methodology for modeling the interlayer material heterogeneity introduced in prior work (Coskun et al., 2010).

Our HotSpot extension allows the user to model the heterogeneity in the layer by modifying the resistivity and capacitance for any unit on the chip. To calculate the thermal resistivity of the blocks with TSVs, in our temperature model, we assume that the TSVs are evenly spread throughout the memory controller. As we know the dimensions of a single Copper TSV, we can calculate the area the TSVs cover in the memory controller block ($Area_{TSV}$) as well as the area of the memory controller block without TSVs. The joint parallel resistivity of Copper and thermal interface material (TIM) can be calculated as follows:

Table 3.4: Thermal simulation configuration in HotSpot.

Thermal Parameters	
Chip thickness	0.1mm
Silicon thermal conductivity	100 W/mK
Silicon specific heat	1750 kJ/m^{3}K
Sampling interval	0.01s
Spreader thickness	1mm
Spreader thermal conductivity	400 W/mK
DRAM thickness	0.05mm
DRAM thermal conductivity	100 W/mK
Interface material thickness	0.02mm
Interface material conductivity	4 W/mK
Heat sink thickness	6.9mm
Heat sink convection resistance	0.1K/W

$$R_{Joint} = \frac{Area}{\frac{Area - Area_{TSV}}{R_{TIM}} + \frac{Area_{TSV}}{R_{Copper}}} \tag{3.2}$$

where $Area$ is the area of a memory controller block where TSVs are located at, $Area_{TSV}$ is the area of the memory controller block with TSVs, R_{TIM} is the thermal resistivity of TIM, and R_{Copper} is the thermal resistivity of Copper. Thus, we get the thermal resistivity for the memory controller block with TSVs as $0.156mK/W$, which is lower than the original TIM resistivity of $0.25mK/W$. We also model the TSVs going through the DRAM layer, and compute the joint thermal resistivity of silicon and Copper as $0.0098mK/W$. We then specify these thermal resistivity values in the floorplan file in HotSpot for temperature computations.

3.3 Performance, Energy, and Temperature Evaluation of 3D Many-core Processors

In this section, we present the evaluation results on the performance, power, and temperature for both of the 16-core `high-performance` and `low-power` systems running parallel workloads. We quantify the benefits of 3D DRAM stacking compared to the equivalent 2D baseline systems.

3.3.1 Performance Evaluation of 3D Many-core Systems

This subsection presents the performance results for 3D systems with on-chip DRAM. Figure 3·7 compares the performance of the 3D systems with on-chip DRAM against the 2D baselines. We use instructions retired per second (IPS) as our performance metric. By using 3D DRAM stacking, we achieve an average IPS improvement of 109.7% for the `high-performance` system and 52.6% for the `low-power` system across the 9 PARSEC benchmarks, compared to the 2D systems with off-chip memory. The

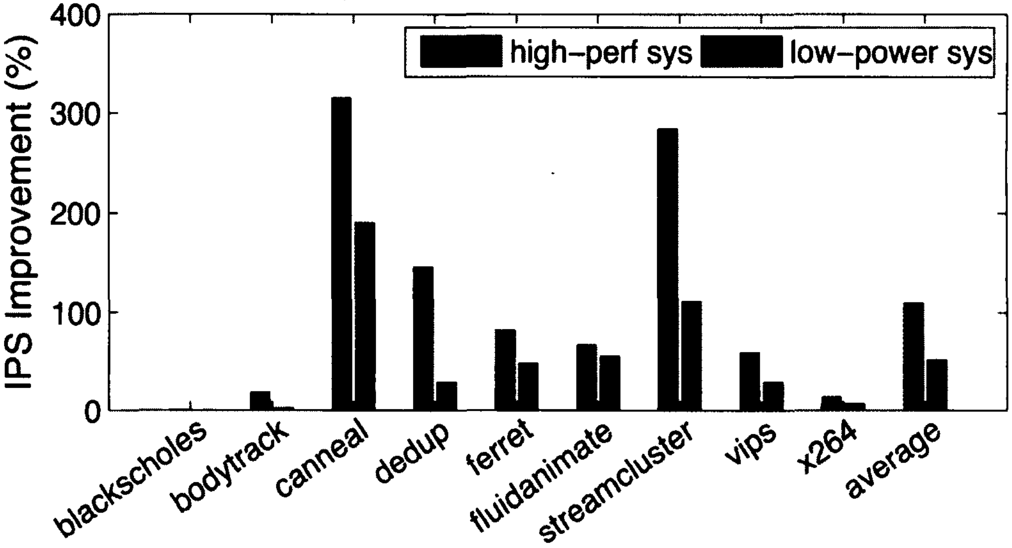


Figure 3·7: Percentage of IPS improvement for 3D systems with DRAM stacking over 2D baselines.

`high-performance` system has larger IPS improvements than the `low-power` system because of its more advanced core architecture.

In both of the `high-performance` and `low-power` systems, streamcluster and canneal achieve higher IPS improvements (over 100%) compared to all the other benchmarks, as these two benchmarks are highly memory-bound and therefore benefit more significantly from the reduction in memory access latency. On the other hand, the CPU-bound benchmarks, such as blackscholes and x264, have limited performance improvement. These results indicate that 3D systems with on-chip DRAM have dramatically high performance improvement for memory-bound benchmarks with high memory access rate.

We select two PARSEC benchmarks, fluidanimate and streamcluster, to demonstrate the temporal performance trends. In Figure 3·8, we observe that for both 2D and 3D architectures the IPS of streamcluster is stable during simulation time, while the IPS of fluidanimate changes periodically as shown in Figure 3·9. These trends are the same in both `high-performance` and `low-power` systems. Also, streamclus-

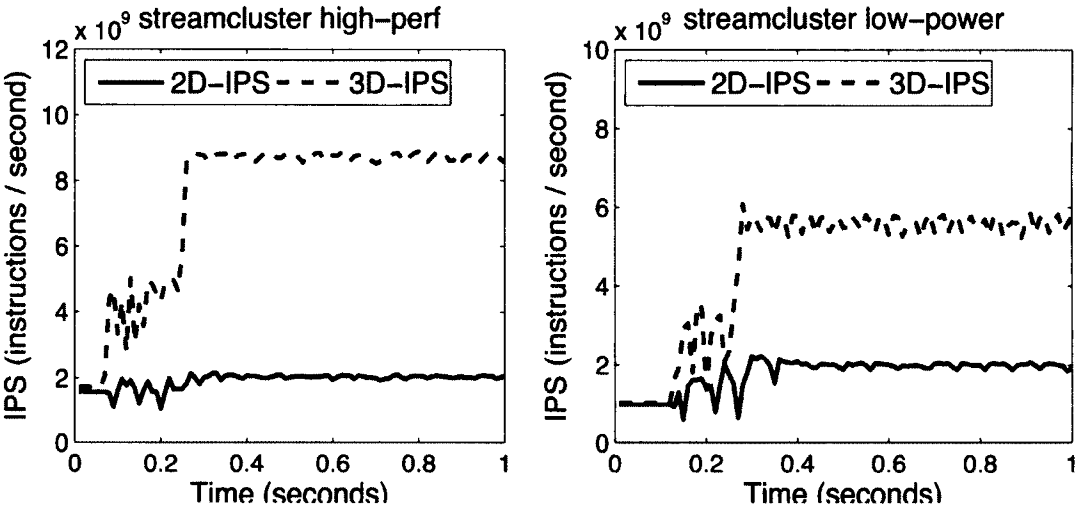


Figure 3·8: IPS temporal behavior analysis of streamcluster running on 3D systems with DRAM stacking versus running on 2D baseline systems.

ter improves its IPS by 284% in `high-performance` system, while fluidanimate has 67.3% higher IPS in comparison to the 2D baseline. This is because streamcluster has a significantly higher number of main memory accesses than fluidanimate.

The significant performance improvement for benchmarks such as streamclus-

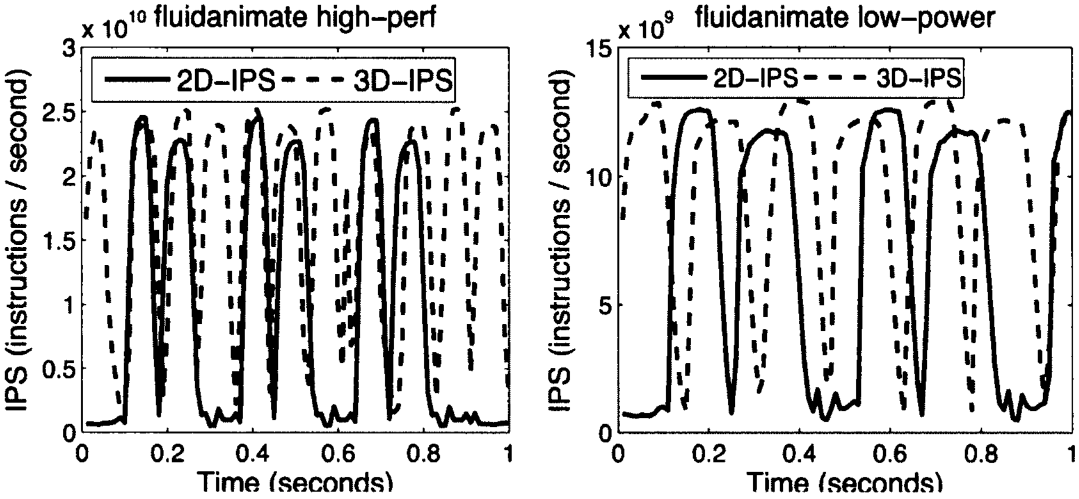


Figure 3·9: IPS temporal behavior analysis for 2D-baseline versus 3D-DRAM systems for fluidanimate.

ter suggests considerable increases in core power. In addition, temporal changes of IPS for some benchmarks, such as fluidanimate, demonstrate that using average power/temperature or coarse-grained performance estimates in the analysis of 3D systems cannot capture the runtime trends accurately. Dynamically changing performance patterns, resulting in higher power and temperatures, can only be observed by detailed architectural evaluation and periodic sampling of runtime events, which are integrated in our simulation approach.

3.3.2 Power Evaluation of 3D Many-core Systems

We present the power evaluation results for 3D systems with DRAM stacking. Figures 3·10 and 3·11 demonstrate the core power increase for the 3D `high-performance` and `low-power` systems, respectively, compared to the 2D baselines.

From the evaluation results, we observe that power consumption per core increases by 29.98% and 6.9% on average for the 3D `high-performance` and `low-power` systems, respectively, across the benchmark set. Among all the benchmarks, canneal has the highest increase in core power, as it has the largest performance improve-

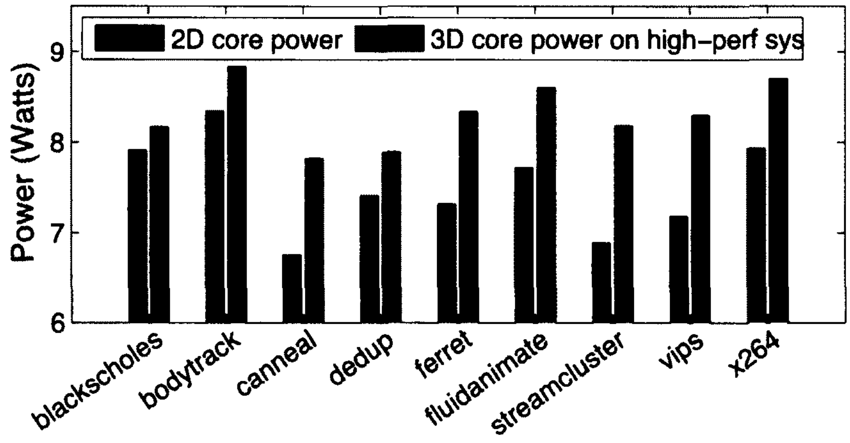


Figure 3·10: Average core power for the 3D `high-performance` system with DRAM stacking and the 2D baseline.

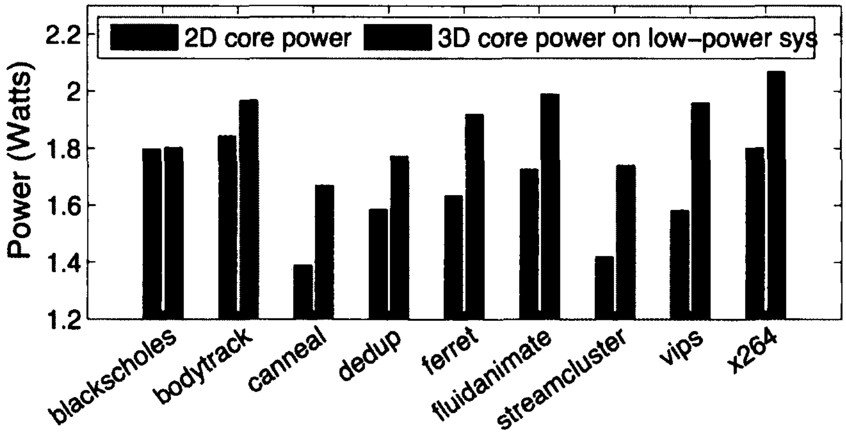


Figure 3·11: Average core power for the 3D `low-power` system with DRAM stacking and the 2D baseline.

ment. The core power of fluidanimate also increases considerably, as it is already at a high power range and the IPS of fluidanimate has additional 67.3% increase in 3D `high-performance` system.

Our results demonstrate an average energy delay product (EDP) improvement of 51.3% for the `high-performance` system and 37.9% for the `low-power` system compared to their equivalent 2D baselines. `canneal` running on `high-performance` system has 88.5% EDP reduction, which is the largest energy efficiency improvement across all the benchmarks. On the other hand, the substantial increase in core power motivates detailed thermal analysis of both systems.

3.3.3 Temperature Analysis of 3D Many-core Systems

We illustrate the thermal behavior for 3D systems in Figure 3·12 for four benchmarks from the PARSEC benchmark sets (canneal, ferret, streamcluster and vips). The peak chip temperatures on the 3D `high-performance` and `low-power` systems and the 2D baselines are shown in the figure. The maximum peak temperature increase is 18.1^oC for running streamcluster in `high-performance` system and 5.8^oC in `low-power` system. We notice that, in comparison to ferret and vips, streamcluster

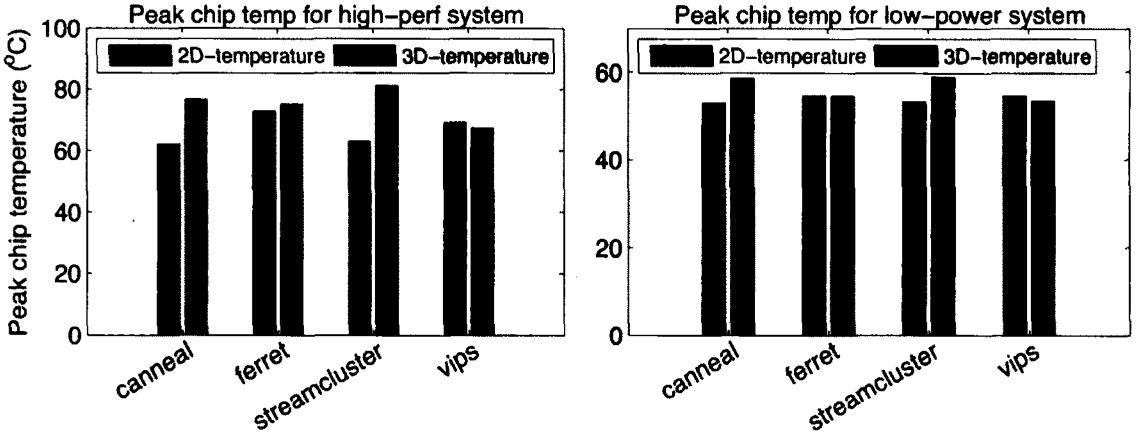


Figure 3·12: Peak chip temperatures for the 2D-baseline and the 3D stacked DRAM systems.

has lower core power while having higher peak chip temperature. This is because that streamcluster has the highest DRAM access rate across all the benchmarks. The high DRAM access rate results in high temperature on the stacked DRAM layer.

We observe that some of the benchmarks running on our 3D systems (e.g., vips) obtain a peak temperature decrease. This is a result of the relatively low memory access rates of vips. Low frequency of memory accesses results in low DRAM power, which already has lower power density compared to the logic layer. The lower power DRAM layer shares the heat of the hotter cores, decreasing the adjacent logic layer temperature for benchmarks with low frequency of memory accesses. These results highlight that it is important to explore the application-aware management and optimization policies to improve the energy efficiency of 3D many-core processors while maintaining the power and temperature constraints.

3.4 Summary

3D integration enables stacking DRAM layers on processor cores within the same chip. On-chip memory has the potential to dramatically improve performance due to lower memory access latency and higher bandwidth. Higher core performance increases power density, requiring a thorough evaluation of the tradeoffs between performance and temperature. However, detailed performance analysis and thermal optimization for 3D processors have been mostly disjoint so far.

In this chapter, we have presented a comprehensive simulation framework for 3D many-core processors. Our simulation framework is able to capture the performance, energy, and temperature of 3D processors running dynamically changing workload, while most current simulation frameworks could only provide the average results. To the best of our knowledge, our work is the first to jointly analyze performance, power, and thermal characteristics for both DRAM and processor layers on 3D many-core processors.

Utilizing this simulation framework, we have evaluated the performance, power, and temperature characteristics of two 16-core 3D processors running parallel benchmark suites. Our results show an average of 109.7% IPS improvement in the 3D processors, while the average per-core power increases by 29.98% and peak temperature increases by 18.1^oC, in comparison to the equivalent 2D processors.

The simulation results demonstrate that 3D processors with DRAM stacking provide significant performance improvement, while brings power and temperature challenges at the same time. These results motivate us to explore runtime management policies for achieving high performance under power and temperature constraints. In the next chapter, we discuss runtime management and optimization methods for 3D many-core processors.

Chapter 4

Runtime Management of 3D Many-core Systems

4.1 Overview

This chapter introduces our research on investigating and developing energy- and temperature-aware management policies for improving energy efficiency and reliability of 3D stacked architecture, with a special focus on the systems with DRAM stacking. Our research consists of investigating existing efficient thermal management techniques and developing novel energy- and thermal-aware optimization policies for 3D many-core processors.

In Chapter 3, we have presented a simulation framework that provides a joint assessment of performance, energy, and temperature tradeoffs in 3D systems with stacked DRAM. Through the evaluation results, we have observed that the workload dynamics change during the lifetime of a system. Thus, it is imperative to have runtime optimization techniques that monitor and actively manage the interplay among performance, power, and temperature of 3D systems.

A number of static management techniques have been proposed for 3D systems to reduce peak chip temperature and optimize system reliability (Cong et al., 2007; Hung et al., 2006; Healy et al., 2007). However, they cannot be adapted to the performance and power variations within and across parallel workloads. In fact, there are dramatic variations with respect to system utilization in today's many-core computing systems, which requires runtime management and optimization approaches. Dynamic management strategies that are proposed in prior work, such as temperature-aware scheduling and DVFS, are effective methods for controlling temperatures on many-core processors (Coskun et al., 2009a). However, their power and temperature results are disjoint from performance simulations, which makes their evaluation results less convincible.

Utilizing the detailed evaluation results from our integrated simulation framework, we are able to analyze the existing dynamic energy and thermal management policies for 3D many-core systems. Leveraging the analysis results, we develop new techniques that are aware of the runtime variations of workloads and system architecture-level configurations. In our work, we focus on energy and thermal management for parallel workloads running on many-core systems, as thread interactions impact performance more in parallel workloads than in single-threaded applications.

In this chapter, we introduce our management policies to optimize the energy efficiency of 3D many-core systems with on-chip DRAM stacking and present the evaluation results. We propose a runtime optimization policy that dynamically monitors workload behavior and selects operating points for adapting to varying application phases. Our policy selects among *low-power* and *high-performance* (or *"turbo"*) execution modes from the available voltage-frequency (V-F) settings by utilizing predictions from a regression-based model. Experimental results demonstrate that our runtime optimization policy achieves an EDP reduction of up to 61.9% compared to a 3D system managed by a temperature-triggered DVFS policy. We also introduce a memory management policy that targets applications with spatial variations in DRAM accesses and performs temperature-aware mapping of memory accesses to DRAM banks. In the end of this chapter, we discuss managing 3D many-core systems with liquid cooling.

4.2 Runtime Management for 3D Many-core Systems

Our runtime optimization policy is motivated by the observations of running PARSEC and NAS benchmarks on our simulation framework under different V-F settings. Figure 4·1 displays the performance results of the 2D baseline and the target 3D system with stacked DRAM. Figures 4·2 and 4·3 present the temperature and power results of the target 3D system in comparison to the 2D baseline system, respectively.

From Figure 4·1, we notice that the average IPS of the 3D system running at 0.8GHz is sufficiently high to match the performance of the 2D baseline for most of the benchmarks. We also observe that applications dramatically differ in their performance behavior. For the memory-intensive benchmarks, such as *streamcluster* and *mg*, the high memory access rates result in significant performance improvements when running on the 3D system with stacked DRAM in comparison to 2D baseline.

However, from Figures 4·2 and 4·3, we can see that the peak temperature also considerably increases with the performance improvements. Thus, we run such memory-

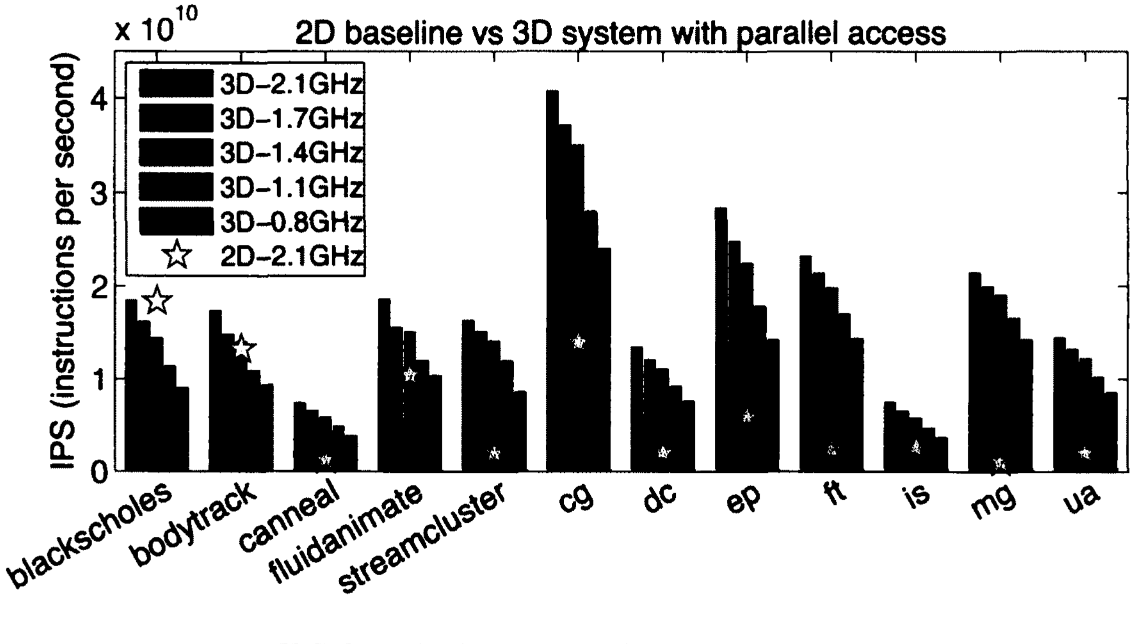


Figure 4·1: IPS for PARSEC and NAS benchmarks running on 2D baseline and the 3D system with parallel access.

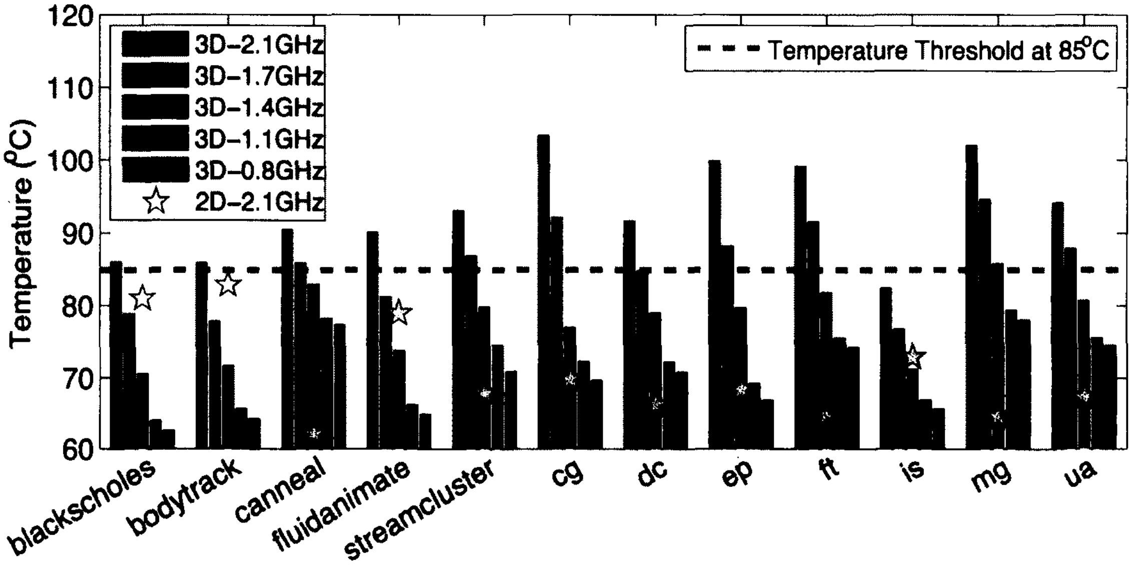


Figure 4·2: Peak chip temperature on the 3D system with parallel access running at different V-F settings.

intensive benchmarks at the *low-power* mode by exploiting the *performance slack*. Figure 4·1 shows that, even at *low-power* mode, the memory-intensive benchmarks running on the 3D system still have significant performance improvements in comparison to running on 2D baseline. For CPU-intensive workloads, on the other hand, the low memory access rates result in a cooler DRAM layer that shares the temperature of the hotter core layer. For benchmarks such as *blackscholes*, we switch to the *turbo* mode with higher V-F settings for boosting the performance by taking advantage of the *temperature slack*.

The goal of our runtime optimization policy is to select operating points maximizing performance while maintaining the power and temperature constrains for both logic and DRAM layers. In order to achieve this goal, we formulate our optimization method as in Equation (4.1). In Equation (4.1), (F, V) is the set of available V-F settings. The objective of our optimization method is to maximize throughput (IPS) under power and thermal constraints. P_{cap} is the power budget of the target system, and T_{thld} is the peak temperature threshold to ensure reliable operation. As shown

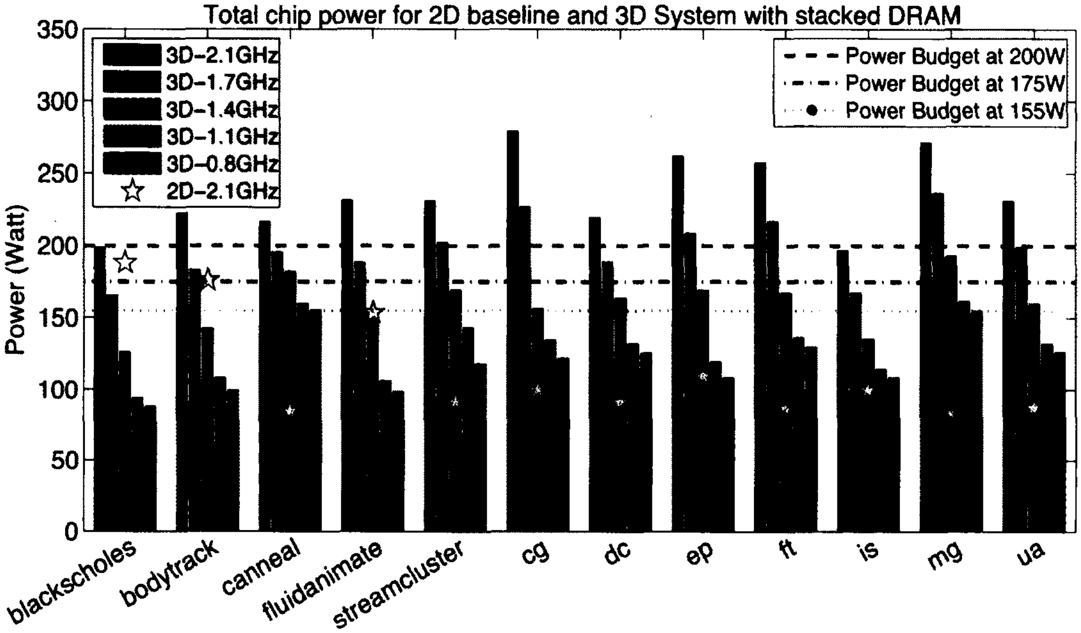


Figure 4·3: Total chip power on the 3D system with parallel access running at different V-F settings.

in Figure 4·2, we set T_{thld} at 85^oC. Figure 4·3 shows three P_{cap} settings. Our policy satisfies T_{thld} and P_{cap} at the same time. For example, at a loose P_{cap} of 200W, T_{thld} at 85^oC dominates the optimization decisions. A more strict P_{cap} at 175W or 155W requires taking peak power into account. Peak power management is an increasingly important feature owing to power supply limitations and potential energy cost reduction opportunities at large computer clusters.

$$\begin{aligned} &\underset{(f,v)\in(F,V)}{\text{maximize}} && IPS(f,v) \\ &\text{subject to} && power(f,v) \leq P_{cap},\ temperature(f,v) \leq T_{thld}. \end{aligned} \tag{4.1}$$

Figure 4·4 illustrates the flow of our runtime optimization policy. We start running the application with the lowest V-F setting to ensure reliable operation, and collect the performance statistics at regular intervals of 100 million instructions. Based

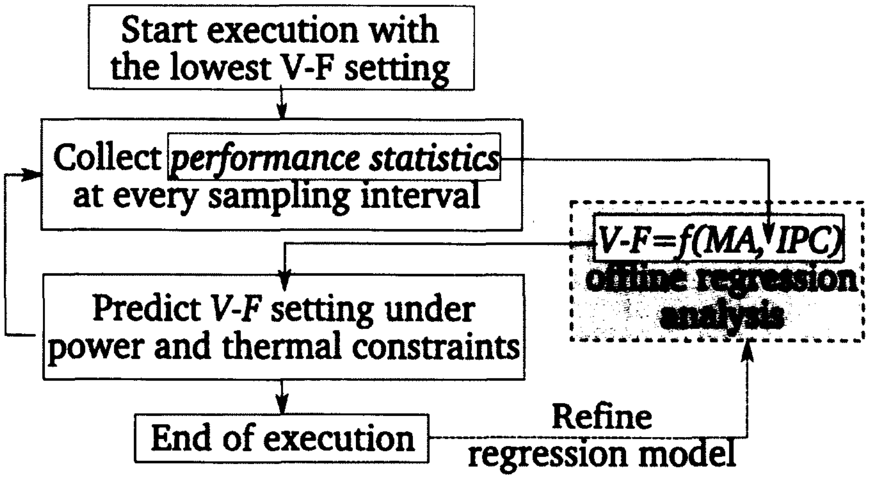

Start execution with the lowest V-F setting
Collect performance statistics at every sampling interval
V-F=f(MA, IPC)
offline regression analysis
Predict V-F setting under power and thermal constraints
End of execution
Refine regression model

Figure 4·4: The flowchart of our runtime optimization policy.

on a model we construct offline, we predict the highest V-F setting satisfying the constraints using the performance statistics as inputs. We continue running the application with the predicted V-F setting. This process is repeated at every interval.

We choose instructions per cycle (IPC) and memory access per instruction (MA) to construct a regression-based model for selecting the V-F settings. This is because IPC is a good indicator of the power of the logic layer and MA is a good indicator of the power of the DRAM layer. Power densities on both layers affect chip peak temperature on the 3D system. Our V-F prediction model is in the form of $V-F = c_0 + c_1 \cdot MA + c_2 \cdot IPC + c_3 \cdot MA * IPC$.

Table 4.1: Regression coefficients for a target 3D system with $85^oC/175$W constraints for all the V-F settings.

V-F setting	c_0	c_1	c_2	c_3
2.1GHz/1.1V	3.68	-147.95	-0.059	0.19
1.7GHz/1.06V	3.74	-141.77	-0.071	0.23
1.4GHz/1.02V	3.76	-145.71	-0.075	0.36
1.1GHz/1.0V	3.80	-147.08	-0.087	0.41
0.8GHz/0.98V	3.87	-152.01	-0.072	0.58

We train the regression model with power and performance statistics from simulations across all benchmarks. Note that we need to use different coefficients in the model depending on the current V-F setting, as MA and IPC vary with the V-F setting. As an example of the V-F prediction for a 3D system with $85^oC/175$W constraints, we list the coefficients of the regression-based model for all the V-F settings in Table 4.1. The regression model provides accurate prediction as shown in Figure 4·9, and can be refined at runtime if needed. The overhead of the runtime prediction is negligible, since computing a simple equation at every interval has very low computational cost.

We evaluate our runtime optimization policy on 3D systems with parallel access, and compare our optimization policy against using static V-F settings, a temperature-triggered DVFS policy, and a DVFS policy guided by memory accesses.

The performance improvement of the 3D system with parallel on-chip DRAM access running at 2.1GHz and 0.8GHz is demonstrated in Figure 4·5. We show that

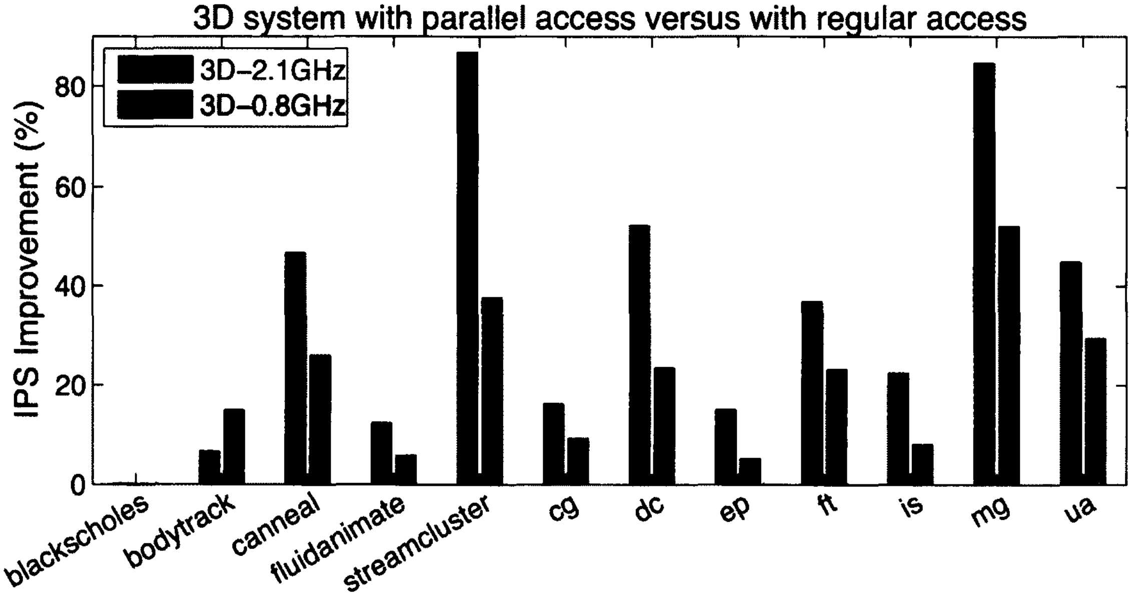


Figure 4·5: Performance improvement on 3D system with parallel access compared to 3D system with regular access.

Table 4.2: Results of the target 3D system with static settings.

Policy	**Static V/F settings (GHz/V)**				
	0.8/0.98	**1.1/1.0**	**1.4/1.02**	**1.7/1.06**	**2.1/1.1**
Peak P (W)	154.72	161.53	193.37	236.79	279.25
Peak T (^oC)	78.10	79.46	85.85	94.65	103.39
EDP*⋆ ($J{\cdot}s$)	246.42	167.63	135.18	132.19	119.82
IPnS⋆**	10.63	12.86	15.73	16.93	18.93

* *EDP per 10billion instructions*

** IPnS stands for instructions per nanosecond

⋆ Average across all benchmarks

enabling parallel access to the 3D DRAM layer improves IPS by up to 86.9% compared to using regular access. `streamcluster` and `mg` show higher IPS improvements than the other benchmarks, since they have higher memory access rates and thus benefit more from reduced average memory access time.

We compare the performance and energy efficiency for 3D systems running our runtime optimization policy and using static V-F settings. The results are shown in Table 4.3 and Table 4.2. We notice that the peak temperatures go over the thermal

Table 4.3: Results of the target 3D system with our runtime optimization policy.

Policy	**Runtime optimization**		
	85^oC/155W	**85^oC/175W**	**85^oC/200W**
Peak P (W)	154.85	168.63	189.62
Peak T (^oC)	77.97	80.81	83.32
EDP*⋆ ($J{\cdot}s$)	185.67	145.11	130.03
IPnS⋆**	14.47	15.68	16.02

* *EDP per 10billion instructions*

** IPnS stands for instructions per nanosecond

⋆ Average across all benchmarks

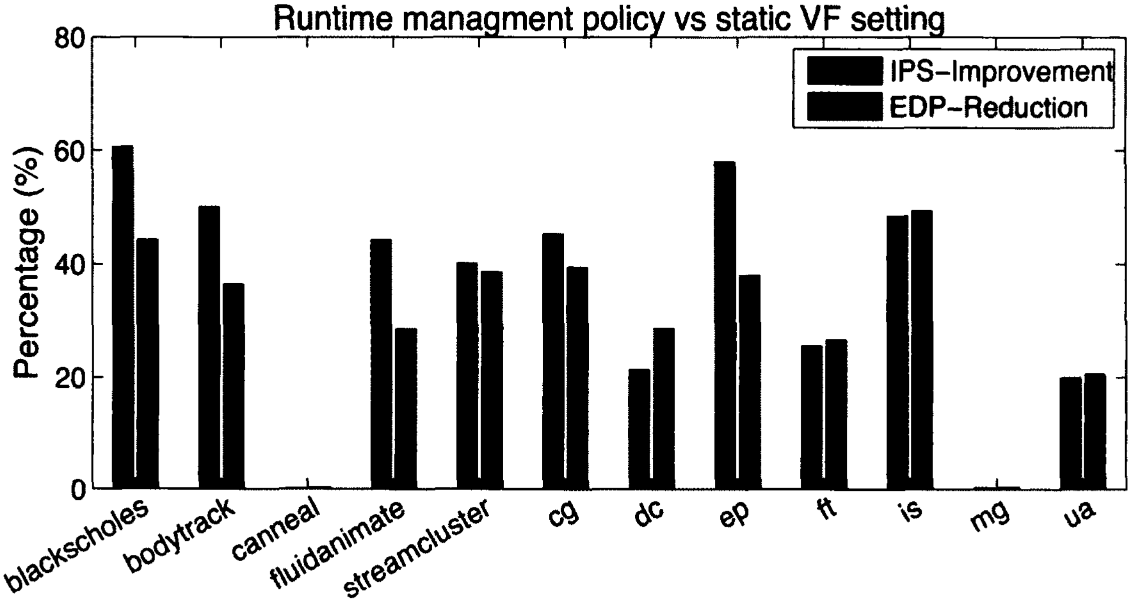


Figure 4·6: 3D system using our runtime management policy in comparison to running all benchmarks at the static V-F setting of 0.8GHz/0.98V.

constraint of 85°*C* for applications running on the 3D systems with frequency settings higher than 1.1GHz.

With a loose power constraint of 200W, we compare our policy with the static V-F setting at 1.1GHz/1.0V which maintains temperature below 85°*C* for all the benchmarks. Our policy achieves an average IPS improvement of 24.6% and EDP reduction of 22.4% across all the benchmarks. With strict constraints of 85°*C*/155W, our runtime policy improve the IPS of 3D system by 60.6% in comparison to static V-F setting at 0.8GHz/0.98V, as demonstrated in Figure 4·6.

We present the runtime V-F selection process of our optimization policy in Figure 4·7. For *ua*, 1.4GHz/1.02V is the reliable static operating point, maintaining the temperature below 85°*C*. However, the phase change of *ua* creates a *temperature slack* periodically. Our policy takes advantage of the *temperature slack* and switches to 1.7GHz during periods of low temperature.

We demonstrate the advantage of our runtime optimization policy over apply-

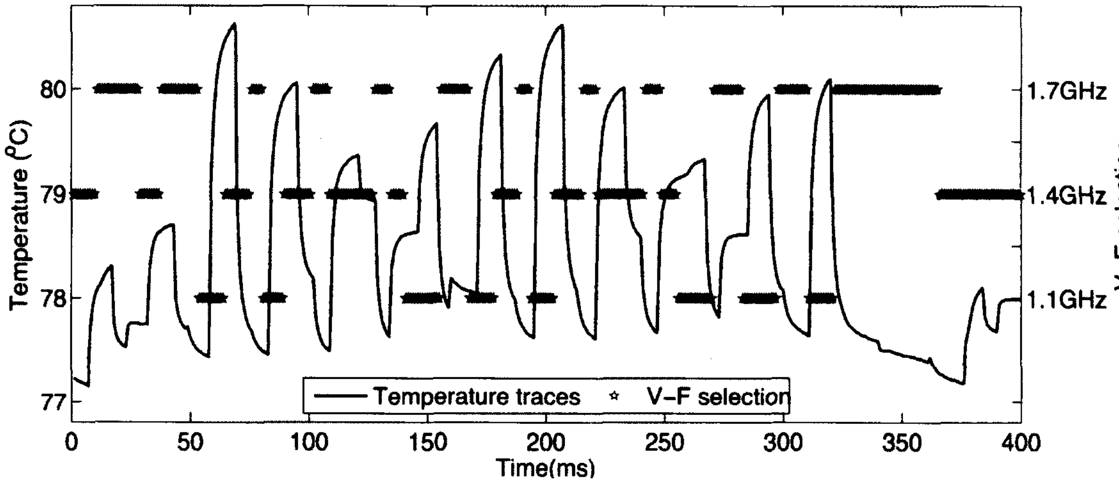


Figure 4·7: Temperature trace of *ua* on the 3D system running at 1.4GHz/1.02V and the V-F setting selected by our runtime management policy.

ing temperature-triggered DVFS in Figure 4·8. Temperature-triggered DVFS is a well-known policy for thermal management on 2D systems (Skadron et al., 2003; Coskun et al., 2009c). It tracks chip peak temperature and selects the operating point based on temperature sensor readings. For safe operation while maintaining system performance, we choose two temperature thresholds as 80^oC and 70^oC.

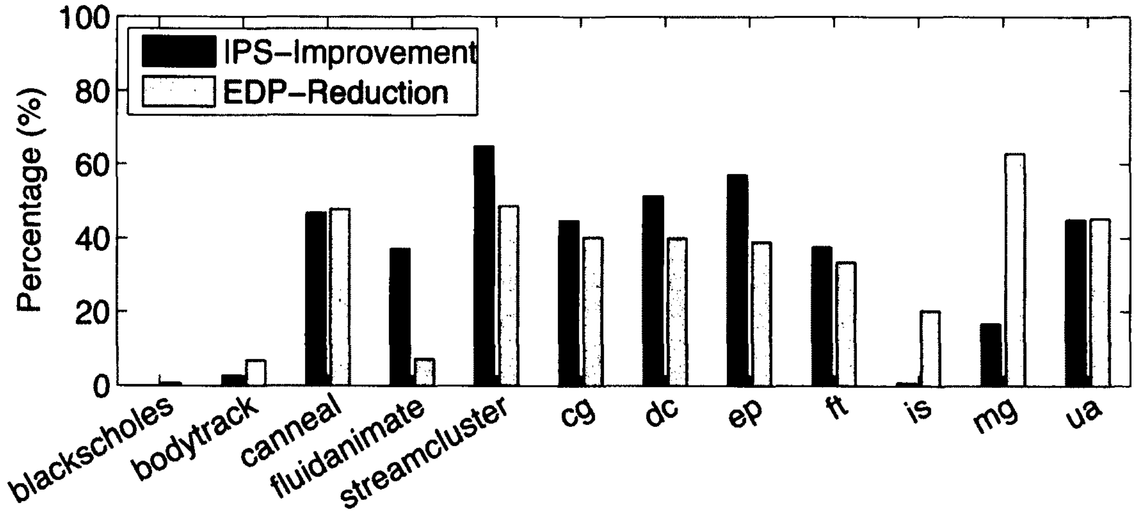


Figure 4·8: 3D system with runtime management policy in comparison to temperature-triggered DVFS policy.

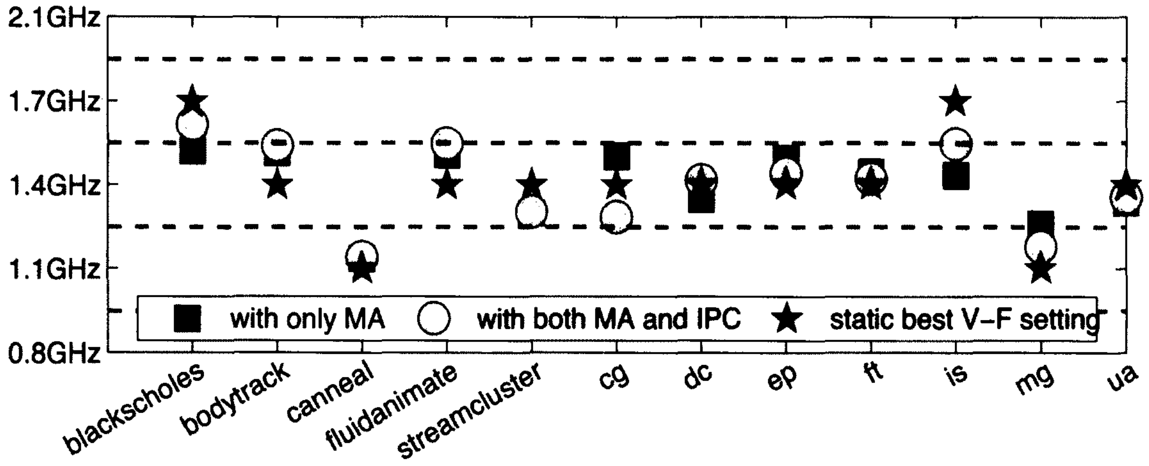


Figure 4·9: Prediction accuracy of our runtime management policy versus memory access (MA) driven DVFS.

Temperature-triggered DVFS reduces/increases the V-F setting when temperature goes above/below $80^oC/70^oC$.

Our policy improves EDP by up to 61.9% and IPS by 32.2% on average across all the benchmarks in comparison to the temperature-triggered DVFS policy. The performance of *blackscholes* and *is* does not differ between our policy and the temperature-triggered DVFS policy. This is because they have low temperature while running at 2.1GHz/1.1V. The benchmarks that have high temperatures when running on 3D systems with stacked DRAM, such as *streamcluster*, show larger performance improvement using our runtime policy. Our policy selects the highest V-F settings to operate at safe temperatures, while temperature-triggered DVFS may oscillate around the high temperature threshold.

We also compare our optimization policy against memory access driven DVFS, in which V-F selections are mainly guided by the memory access rate (Isci et al., 2006b). For implementing memory access driven DVFS, we construct a regression-based model for selecting V-F setting with only MA. We show the V-F prediction for 3D system with 85°/ 175W constraints in Figure 4·9. By only using MA, three out of twelve benchmarks end up with different V-F settings than the optimal ones; while the predictions are all accurate using both IPC and MA as in our policy. The benchmarks that are predicted incorrectly using only MA are *blackscholes*, *is*, and *mg*. *blackscholes* has low MA but high IPC, *is* has both low MA and low IPC, and *mg* has high MA and relatively higher IPC than the other memory-bound benchmarks. Our policy provides accurate prediction as we take the power and temperature constraints on both logic and DRAM layers into account on 3D systems with stacked DRAM, where both high IPC and memory access rate could result in high chip power and peak temperature.

In addition to developing the runtime optimization policy to exploit the performance potential of 3D many-core systems with DRAM stacking, we also investigate management approaches to control the temperature of DRAM layer. DRAM performance is severely affected from high temperatures due to the impact of temperature on DRAM refresh rates. In fact, prior research has shown that temperature sensitivity often becomes more critical for memory layers than for logic layers (Ghosh and Lee, 2007; Liu et al., 2011).

In order to reduce the temperature and thermal variation on both the logic layer and the DRAM layer of the 3D systems, we propose a memory address management policy. The motivation of implementing this method is base on two facts. One is that high memory access rate of a DRAM bank is generally raising up high power, and the temperature of a DRAM bank is the result of the power on both itself and its neighbors. The other is that the temperature of a DRAM bank is dependent on its location on the 3D DRAM layers, the banks that are located on the center of the DRAM layers generally have higher temperatures than the banks that are on the corners of the 3D DRAM layers. Therefore, the main idea of the memory address management policy is to map more frequently accessed memory address ranges to

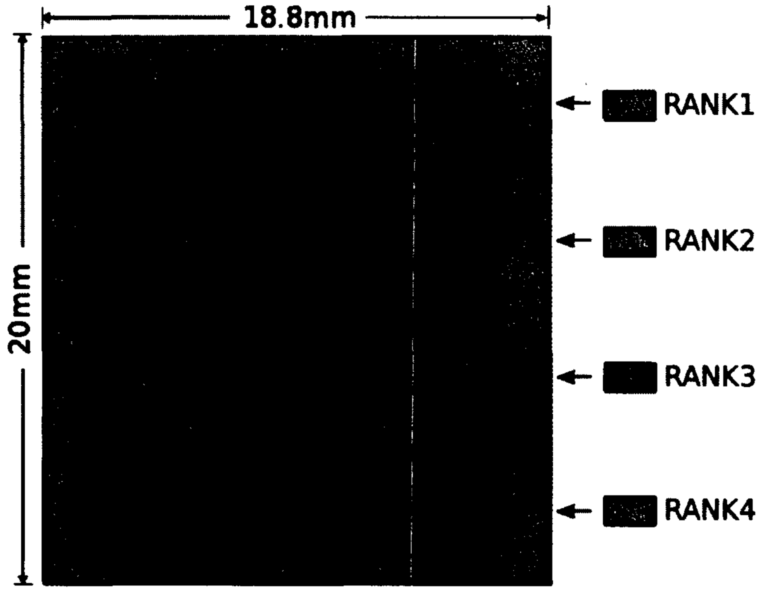


Figure 4·10: The DRAM layer layout for the `high-performance` 3D system with on-chip DRAM.

physical banks with lower temperatures.

Our policy targets memory-intensive applications with high spatial variations in their access rates across different DRAM banks. Figure 4·11 illustrates the peak temperatures and the number of accesses per cycle across the 16 DRAM banks while running `streamcluster` on the 3D `high-performance` system with 128-Byte memory bus. The location of each bank is shown in Figure 4·10. Banks 6, 7, 10, 11, which are located on the center of the DRAM layer have higher temperatures than banks 1, 4, 13, 16, which are on the corners. The variations in DRAM bank access rates indicate differences in power consumption across the DRAM banks. In Figure 4·11, the most accessed DRAM bank 9 and least accessed bank 3 have average power consumption of 5.1W and 1.9W, respectively.

Based on this analysis, our memory management policy maps more frequently accessed memory address ranges, such as the address range for bank 9 in the default mapping, to physical banks with lower temperatures (e.g., bank 1). The memory address mapping is implemented by the OS when virtual memory addresses are

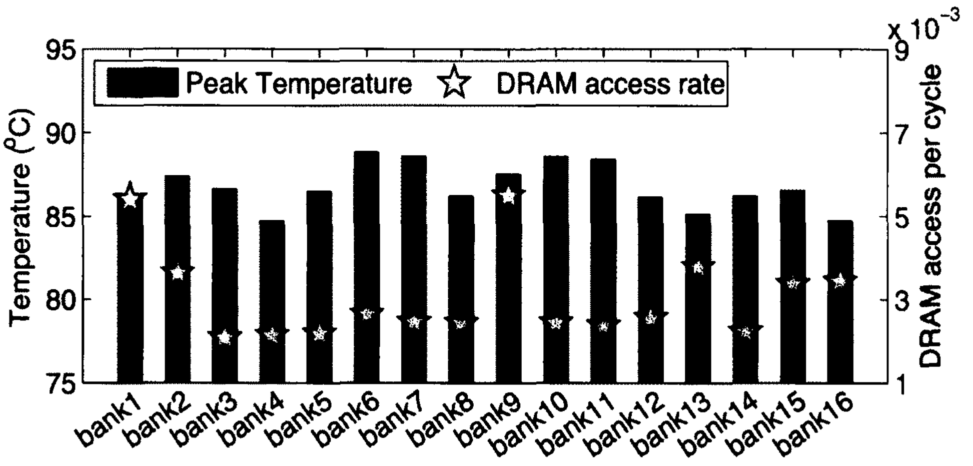


Figure 4·11: DRAM bank temperature and access rate for `streamcluster` in 3D `high-performance` system with 128-Byte memory bus.

translated into physical addresses. The specific memory mapping strategy matching the virtual memory address ranges to physical locations can be determined based on average case analysis statically. This approach has no additional cost compared to existing memory mapping mechanisms. The mapping policy can also be updated if average case workload dynamics change significantly. Simulation results show that our policy reduces DRAM peak temperature by 1.4^oC and the thermal variations by 2^oC for `streamcluster` running on the 3D `high-performance` system with 128-Byte memory bus in comparison to the worst-case allocation, where the banks receiving higher number of accesses are located in the center of the DRAM layer.

4.3 Managing 3D Many-core Systems with Liquid Cooling

Many-core systems provide a lot of hardware parallelism and potential performance increase. However, as recent chip sizes for many-core systems reach $300mm^2$ to $400mm^2$ and more, they are prone to larger process variations, lower yield, and higher on-chip wire delay and power consumption.

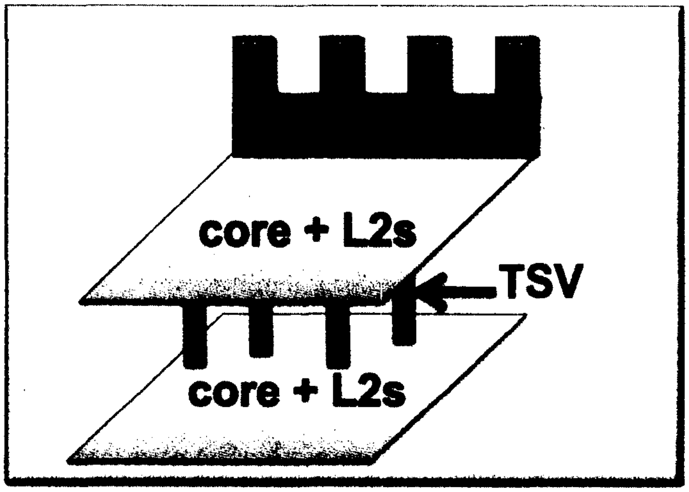


Figure 4·12: An illustration of 3D many-core systems with two logic layers stacking and off-chip DRAM.

3D many-core system with logic layer stacking is a promising solution to design large many-core chips as it improves manufacturing yield because of smaller chip area, and reduces wire length and capacitance. However, as the number of cores and number of logic layers in 3D many-core systems increase, system temperature easily goes out of feasible ranges, even by applying the thermal management policies for 3D many-core systems that we have proposed in Section 4.2. Liquid cooling has a higher efficiency of removing heat compared to conventional heat sinks, thus are introduced to address the thermal challenges in 3D many-core systems. In this section, we discuss the modeling and management of 3D many-core systems.

We use the simulation framework that introduced in Chapter 3 for the modeling of 3D many-core systems with logic stacking. We assume our target 3D many-core system as a 64-core processor that is manufactured at 45nm. As illustrated in Figure 4·12, the target 3D many-core system has two vertically stacked logic layers and off-chip DRAM. We assume that the floorplans of the two logic layers are identical, each layer has 32 core, and each core has a private L2 cache. The core and cache architectural parameters are the same as for the target 2D many-core system, which

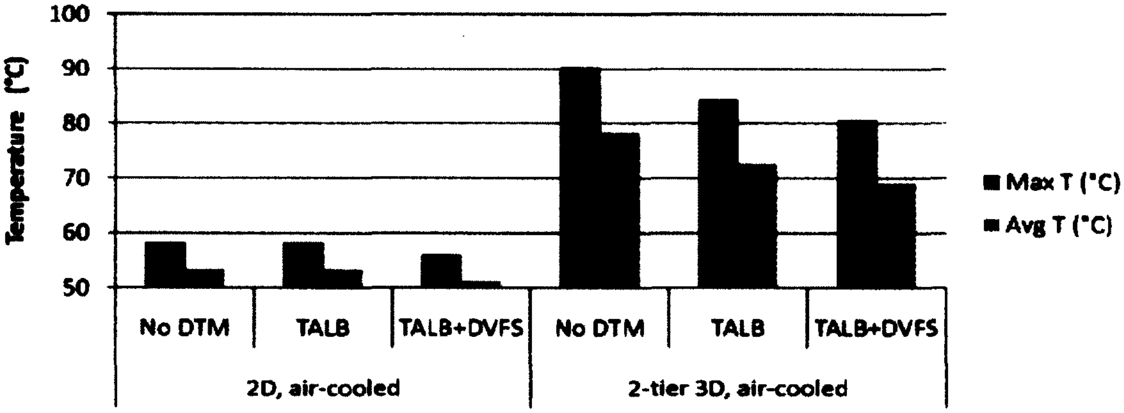


Figure 4·13: Peak and average temperatures for 64-core 2D system and 3D system with two logic layers, including the results with no thermal management (No DTM), with temperature-aware load balancing (TALB), and with TALB combined with DVFS (TALB+DVFS).

are shown in Table 3.1.

Figure 4·13 presents the peak and average temperatures for 64-core 2D system and target 3D many-core system with two logic layers. In the simulation framework, we assume negligible difference in core performance between 2D and 3D systems when they are running the same applications, because the already low cache access times are not strongly affected by vertical stacking. We notice that temperature increases significantly due to vertically stacking two logic layers. The peak temperature with no thermal management of the target 3D many-core system increases by around 30^oC in comparison to the peak temperature with no thermal management of 64-core 2D system. We compare the results of peak and average temperatures for the 2D and 3D many-core systems with no thermal management (No DTM), with temperature-aware load balancing (TALB), and with TALB combined with DVFS (TALB+DVFS), respectively. TALB allocates jobs to cores with the objective of balancing chip temperature (Coskun et al., 2010).

The comparison results show that TALB reduces the peak temperature below the critical value of 85^oC, and TALB+DVFS reduces the temperature further. This observation demonstrates the significance of thermal management for 3D many-core systems with logic layer stacking. However, when we build the same system into a 3D system with four logic layers stacking, each logic layer consisting of 16 cores, the peak and average temperatures exceed 100^oC and 90^oC even with TALB+DVFS, which makes the design of 3D many-core systems unfeasible (Coskun et al., 2011). Therefore, for high-performance 3D architectures, applying scheduling, DVFS, or other existing techniques cannot mitigate the temperature challenges effectively without hurting the system performance. We need to consider more efficient heat removing techniques, such as liquid cooling, to address the thermal challenges in 3D many-core systems.

Liquid cooling has been proposed as a promising solution to address the pressing thermal challenge of 3D many-core systems due to the logic stacking, as it has a higher efficiency of removing heat compared to conventional heat sinks and fans. A prototype 3D system with built-in microchannels has been manufactured by IBM Zurich and EPFL (Brunschwiler et al., 2009; Coskun et al., 2011). The modeling of 3D many-core system temperature with liquid cooling model already exits and is implemented in HotSpot (Skadron et al., 2003; Coskun et al., 2010). However, liquid cooled 3D many-core systems bring new challenges in cooling control and require efficient integration with chip-level thermal management techniques.

We have looked into managing the 3D many-core big chips with microchannel cooling (Coskun et al., 2011). Figure 4·14 compares maximum and average temperature between liquid-cooled 3D systems and the 2D air-cooled baseline. We observe that liquid cooling dramatically reduces temperatures for the 3D many-core systems with multiple logic layers, which makes stacking more logic layers possible. From the simulation results, we can see that the temperature are within the safe margins for both the 2-tier and 4-tier 3D systems with liquid cooling. We use Fuzzy+TALB

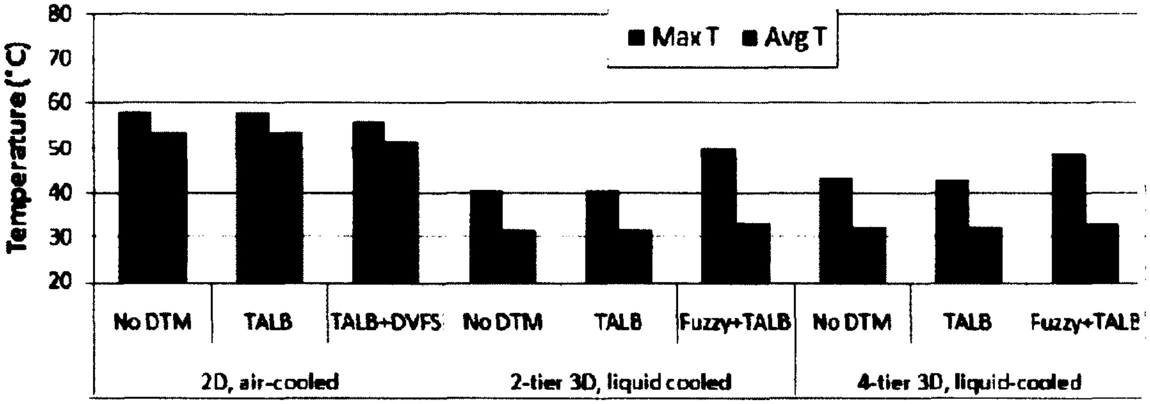


Figure 4·14: Peak and average temperature between liquid-cooled 3D systems and the 2D air-cooled baseline, including the temperatures results with no thermal management (No DTM), with temperature-aware load balancing (TALB), and with TALB combined with DVFS (TALB+DVFS), as well as fuzzy controller combined with TALB (Fuzzy+TALB).

to prevent over cooling and reduce the cooling energy by adjusting the flow rate to match the cooling need of the system (Sabry et al., 2010). We observe that TALB, Fuzzy control, and DVFS all contribute to the reliable operations on liquid-cooled 3D many-core systems by reducing the peak and average temperatures.

4.4 Summary

In this chapter, we have discussed the management and optimization policies to address the energy efficiency and thermal challenges for 3D many-core systems. We have proposed a runtime optimization policy that dynamically monitors workload behavior and selects operating points for adapting to varying application phases. Our policy selects among *low-power* and *high-performance* execution modes from available V-F settings by utilizing predictions from a regression-based model. The simulation results show that our runtime optimization policy achieves an EDP reduction of up to 61.9% compared to a 3D system managed by a temperature-triggered DVFS policy. We have also introduced a memory management policy that targets applications with spatial variations in DRAM accesses and performs temperature-aware mapping of memory accesses to DRAM banks. In order to further reduce the temperature of 3D many-core systems, we have also discussed the management of 3D systems with liquid cooling. In the following chapters, we will present the modeling and management approaches for large-scale many-core systems in HPC data centers.

Chapter 5

Modeling of Many-core Systems in Data Centers

5.1 Overview

As the number of cores and power density per processor increase, performance, cooling energy cost, and reliability are becoming critical concerns in many-core systems in HPC data centers. Different from the performance of many-core single-chip processors, the system performance of data centers running communication-intensive applications is significantly impacted by the communication cost between different processing nodes. High temperatures in data centers not only cause reliability degradation, but also increase the required cooling energy of HPC clusters. Therefore, it is important to have detailed modeling approaches to evaluate the communication cost, cooling energy, and reliability of many-core systems in HPC data centers.

In this chapter, we provide a performance model to evaluate the communication cost of HPC data centers running highly parallel workloads. We also present a thermal model to evaluate the inlet temperature and cooling energy cost of HPC data centers. In order to quantify the system-level reliability, we introduce a detailed reliability modeling approach to accurately model temperature-induced wear-out failure mechanisms under various system topologies. In the next chapter, we will propose management strategies for HPC data centers based on the evaluation results by utilizing these modeling approaches.

5.2 Performance and Cooling Energy Modeling in HPC Data Centers

In this section, we introduce the performance and cooling energy model using a small size data center with two rows of industry standard racks as an example. The layout of the target data center is shown in Figure 5·1. In this layout, the rack inlets where the cool air is supplied face the outer aisles and form cold aisles at the sides. The rack outlets, where the hot air exits, face each other and form a hot aisle in between the two rows.

In our target data center, each row is composed of 5 racks and each rack has 4 computing *nodes*. We assume that each *node* includes 10 servers and each server has 2 processors. This layout corresponds to a total of 800 processors across the two rows of the data center. The proposed data center layout has been widely used in

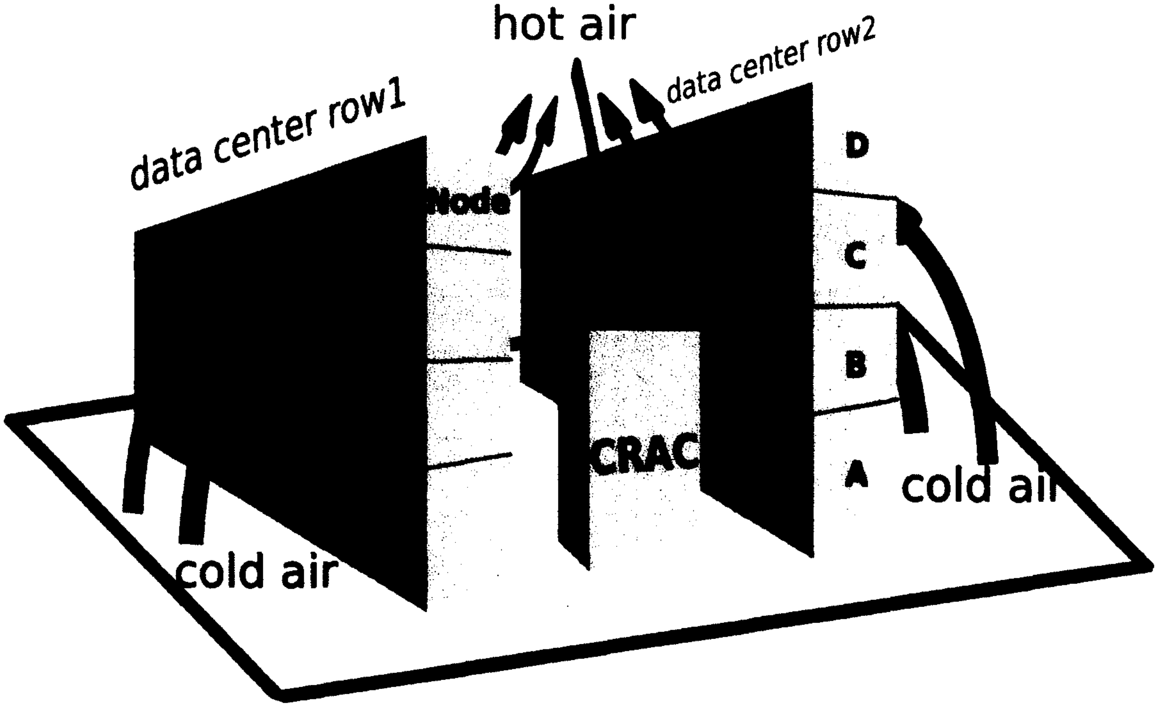


Figure 5·1: Layout of the target data center.

prior work and is representative of today's data center configurations (Sansottera and Cremonesi, 2011).

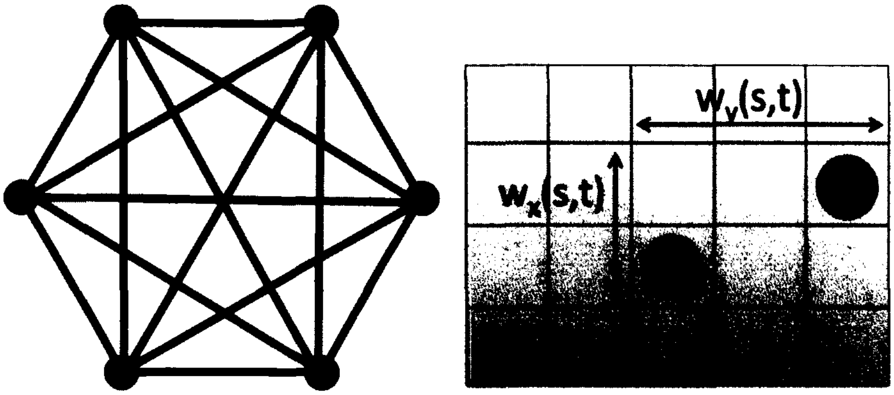
5.2.1 Workload and Performance Model

The typical workloads in HPC data centers are communication-intensive parallel applications that use high-level message passing interfaces such as MPI. For such workloads, the communication overhead inherent in the data center is one of the major performance bottlenecks (Mache et al., 1997). In order to model communication costs due to message passing, we target mesh-connected HPC data centers and supercomputing systems. Mesh-connected networks for message passing are widely used in many experimental and commercial distributed memory parallel computers, such as IBM BlueGene/L and Cplant, a commodity-based supercomputer developed at Sandia National Laboratories (Brightwell et al., 2000).

We specify our workloads as *jobs* that require a number of nodes in the data centers. The performance metric for our evaluation is the average pairwise L1 distance (Manhattan distance) across all the communicating nodes of a job running on the mesh-connected parallel system (Bender et al., 2008). We employ L1 distance as our metric as it has been demonstrated to correlate with application running time (Leung et al., 2002). We define the *communication cost* of a job as the average L1 distance across all the nodes running the job, and formulate it as in Equation (5.1).

$$CC_{job} = \frac{1}{N} \sum_{(s,t)\in(S,T)} [w_x(s,t) + w_y(s,t)] \tag{5.1}$$

where CC_{job} means the communication cost of a job. N is the job size. In this thesis, we assume $N \geq 1$ for all the jobs. We define the job size as the number of nodes a job requires. (s,t) represents the pair of source and destination nodes of a message



(a) All-to-all pattern (b) Distance metrics

Figure 5·2: Communication pattern and distance measure.

and (S,T) is the set of all the source and destination node pairs for all the messages. $w_x(s,t)$ and $w_y(s,t)$ represent the distance between s to t along the x-axis and y-axis, respectively. An illustration of the L1 distance between source and destination nodes is shown in Figure 5·2(b). The division of the summation of the L1 distances by N provides the normalization of the communication cost with respect to job size.

In this thesis, we assume *all-to-all* communication pattern for our workloads. *All-to-all* is a common communication pattern in HPC routines such as Fast-Fourier-Transform, which is part of several applications including molecular dynamics, quantum chemistry, and digital signal processing (Kumar et al., 2008). In *all-to-all* pattern, each processor communicates with all the other processors running the same job, as shown in Figure 5·2(a). In order to reflect the difference between communication cost within data center rows and between data center rows, we set the one-hop distance within a data center row as 1 and the distance between nodes of different rows as 10. The reason for the larger distance among rows is that nodes placed at different rows communicate through a larger number of switches and longer interconnects on the communication path (Belden, 2007). Thus, this effect should be included in the communication latency calculation.

We quantify the effect of job communication cost on the job running time by assuming that the application spends a certain percentage of time on communication, denoted as C% (Crovella et al., 1992). In order to calculate the job running time, we use the minimum CC_{job} that can be accomplished for a given job size as the baseline for a job's communication cost. For example, for a job of size 4, the minimum achievable communication cost is 4 using Equation (5.1). We then define the ratio of the current CC_{job} to the best case CC_{job} as the latency factor, L_f. We calculate the actual job runtime by scaling the communication portion of the runtime by L_f.

5.2.2 Cooling Energy Model

The cooling energy consumed by a data center is dependent on the layout of the data center infrastructure. In this thesis, we use a typical data center layout validated by prior work (Rad et al., 2008), as shown in Figure 5·1. In this layout, racks and perforated vent tiles are placed on a raised floor. Cold air enters the room from the floor tiles, goes into the rack inlets from the sides, and gets hotter as it moves through the racks. Hot air exits from the back of the racks into the center aisle and the exhaust air exists the room from the ceiling to be cooled again. This set-up is called hot aisle / cold aisle arrangement which avoids mixing cold supply air with exhaust air (Rad et al., 2008).

In order to evaluate the energy consumption of a data center and develop management policies, we need a fast and accurate data center thermal model. We use the model proposed and validated by Tang et al (Tang et al., 2006). Their model combines a linear, low complexity heat recirculation model with a linear power model. The proposed model is more practical than the most of other existing models as it requires a set of computational fluid dynamics (CFD) simulations only once to characterize the data center. After we obtain the measured data center specific parameters, the vector of inlet temperatures, $\boldsymbol{T_{in}}$, for all the nodes are computed using the following linear equation:

$$\boldsymbol{T_{in}} = \boldsymbol{T_{sup}} + \boldsymbol{D} \cdot \boldsymbol{P} \tag{5.2}$$

$$\boldsymbol{D} = [(\boldsymbol{K} - \boldsymbol{A}^T\boldsymbol{K})^{-1} - \boldsymbol{K}^{-1}] \tag{5.3}$$

where T_{sup} means the CRAC unit supply temperature vector. **D** is the heat distribution matrix and **P** is the node power vector. **K** is the thermodynamic constant matrix and **A** is the heat cross-interference coefficient matrix representing the recirculation phenomena. The thermodynamic constant matrix **K** is calculated as:

$$\boldsymbol{K} = diag(K_i) \tag{5.4}$$

$$K_i = \rho f_i c_p \tag{5.5}$$

where ρ=1.19 Kg/m^3 is the density of air. $0 \leq i \leq N$, where N is the total number of computing nodes in the data center. f_i=0.2454 m^3/s is the flow rate of node i. We assume the flow rate is fixed for all data center nodes, and c_p=1005 J/KgK is the specific heat of air (Tang et al., 2008).

The heat cross-interference coefficient matrix **A** represents the fraction of output heat from each node that is recirculated to the inlet of other nodes. It is an $N \times N$ matrix for a system with N nodes. In matrix **A**, each term a_{ij} represents the fraction of heat at node i recirculating back into node j. It has been shown that elements of matrix **A** mostly depend on the data center layout rather than the power consumption of the nodes or the supply temperature (Sansottera and Cremonesi, 2011). Therefore, this matrix is obtained once for a data center. The matrix **A** for the proposed data layout has been calculated through CFD simulations in prior work (Tang et al., 2006). If one has input ambient sensors mounted already, the matrix **A** can be obtained using

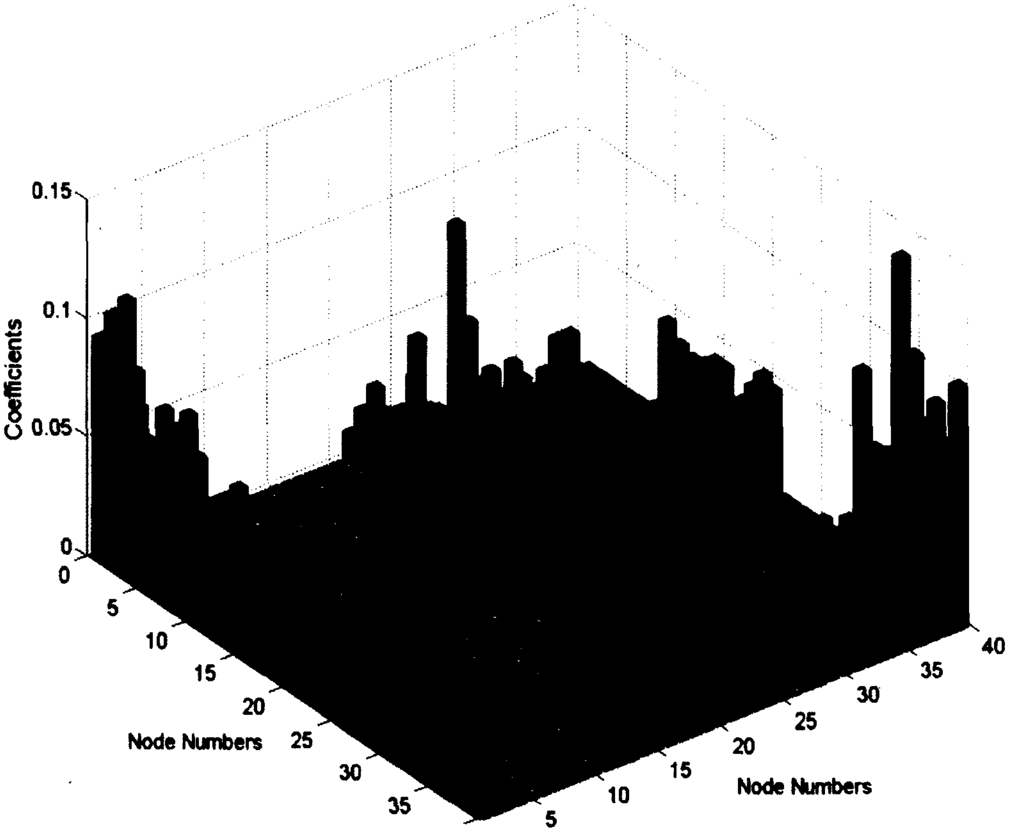


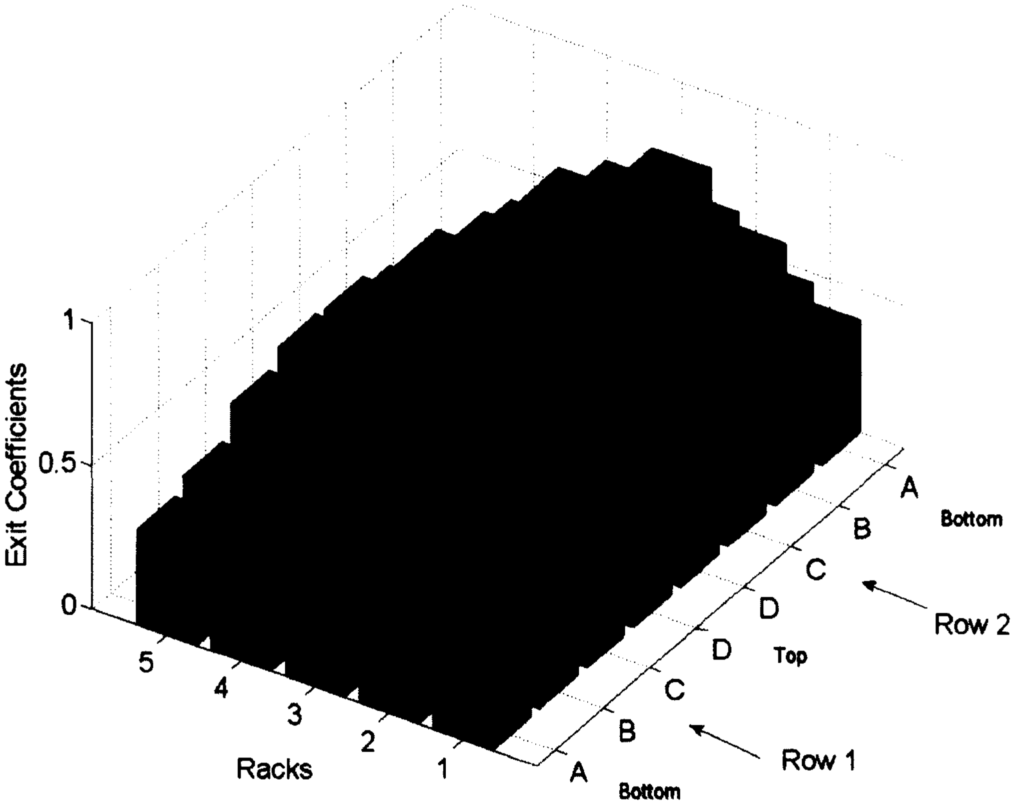
Figure 5·3: Cross-interference coefficient matrix for our system.

sensor measurements instead of CFD simulations and following the same procedure in (Tang et al., 2006).

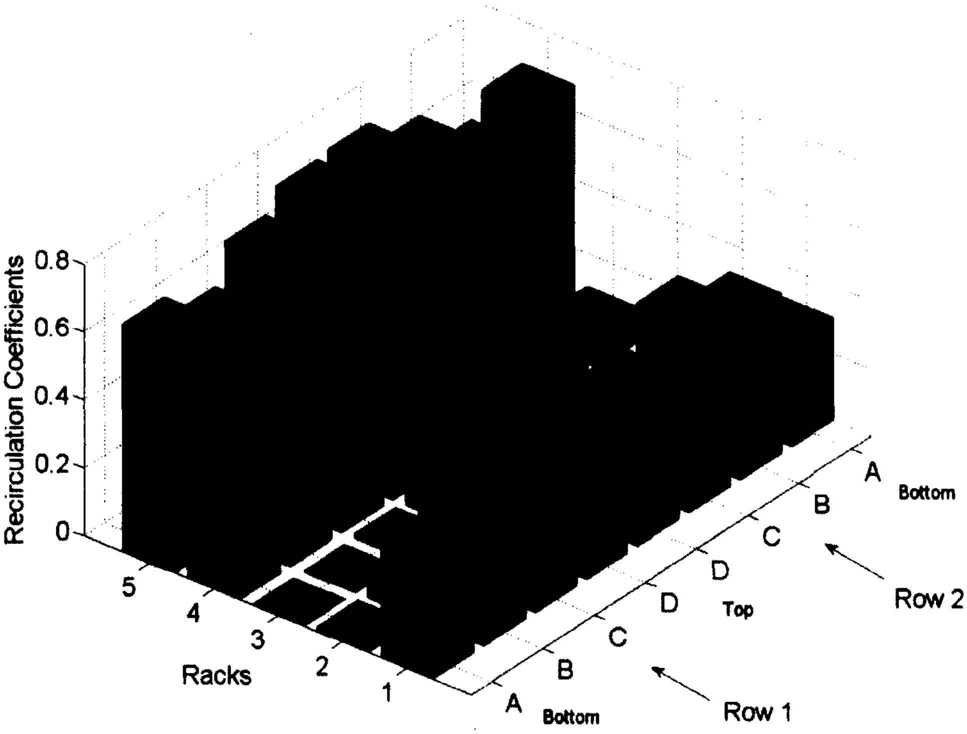
We construct the cross-interference coefficient matrix for the proposed data center using the coefficients given in (Sansottera and Cremonesi, 2011). In order to obtain the coefficients, we extract the coefficient value corresponding to each data point from the colormap plot in (Sansottera and Cremonesi, 2011). We use Matlab to implement the extraction. In Matlab, we first map RGB values to indexes, which preserve the relationship between coefficients relative to each other. We then perform calibration by scaling the matrix according to the given temperature graph in (Sansottera and Cremonesi, 2011). Figure 5·3 shows the cross-interference coefficient matrix **A** for the 40-node system in a 3-D plot. For a data center with different layout and heat flow characteristics, matrix $\mathbf{A}$ differs. The equations to calculate the inlet temperatures are independent of the data center layout. Thus, the cooling energy model applies to data centers in general.

In order to develop efficient management policies for reducing data center cooling energy consumption, we need to consider the specific thermal behavior of the given data center and take the differences in recirculation coefficient and exit coefficient of data center nodes into account. For a node j, $\sum_{i=1}^{n} a_{ij}$ is called the recirculation coefficient (RC) of node j and is a measure of the total recirculation effect of that node (Tang et al., 2006). On the other hand, for a node i, the value of $(1 - \sum_{j=1}^{n} a_{ij})$ is called the exit coefficient (EC) of node i. EC is a measure of the heat at node i's outlet returning back to the cooling system without recirculating back to other nodes. EC and RC for our system are given in Figure 5·4. As presented in Figure 5·4(a), the nodes at the bottom of the racks and at the ends of the aisles have lower EC values, which means that they contribute more to the recirculation effect. Moreover, according to Figure 5·4(b), the nodes at the top and at the ends of the aisle have higher RC values, which means that they are affected or victimized more by the recirculated heat. The asymmetry between EC and RC values of right and left end of the aisles is due to asymmetries in the heat flow within the data center.

In addition to the data center layout, the processing powers of the data center nodes and the allocation of jobs also play important roles in the cooling energy consumption in the data center. These power values are used in Equation (5.2) to calculate the inlet temperatures resulting from different allocation schemes. We perform node-level allocation in this thesis, which is a reasonable hierarchical level for HPC data centers. Once a job is assigned to a node of multiple servers, server and core level workload allocation will follow. Assume a given task of size n, which corresponds to the total number of nodes a task requires, x_i is an integer variable showing



(a) Exit Coefficients



(b) Recirculation Coefficients

Figure 5·4: Exit and recirculation coefficients for our system.

whether node i is assigned a job or not and it is either 1 or 0, respectively. Power consumption of node i can be expressed with a linear model as follows:

$$P_i = P_{idle} + x_i P_{util} \tag{5.6}$$

where P_{idle} is the node idle power and P_{util} is the power consumed by a node when running a task. We assume fixed node power when running a task is valid for HPC data centers. This is because that, even though there are fluctuations in the power, it is ignorable in comparison to the total power. We use 1000W for P_{idle} and 2500W for P_{util}. For a data center node that is processing a job, the total processing power is 3500W. These numbers are in line with the server power values given in (Sansottera and Cremonesi, 2011).

We adjust the total node power according to the actual runtime and percentage of time spent in communication. During communication intensive phase, power consumption will be lower due to the time spent waiting for messages. We assume 2 different power levels, 3500W for computation phase and 2700W during communication phase. These numbers are in line with the values in (Lively et al., 2011). We set the total power of a node as the weighted sum of the computation power and communication power. Communication level (C%) or the power levels corresponding to computation/communication phases depend on the workload and power characteristics of the system. The modeling of communication level provides us the ability to evaluate our optimization policy that is applicable to systems with different power and communication levels.

After we obtain the node inlet temperatures using Equation (5.2), we need a cooling power model to measure the power consumed by the cooling unit at various temperatures. This power depends on the efficiency of the CRAC unit. One of the most common metrics used for CRAC unit efficiency is the coefficient of performance

(CoP). CoP is defined as the ratio of the heat removed from the system to the energy spent on cooling and has the following formula:

$$CoP = \frac{P_c}{P_{AC}} \tag{5.7}$$

where P_c is the total computing power (sum of the values in **P** vector) and P_{AC} is the cooling power. CoP increases with higher CRAC supply temperature (T_{sup}). In this work, we use the CRAC unit CoP model given by (Moore et al., 2005) as follows:

$$CoP(T_{sup}) = 0.0068\,T_{sup}{}^2 + 0.0008\,T_{sup} + 0.458 \tag{5.8}$$

where T_{sup} is in Celsius. The upper limit on how much we can increase supply temperature (T_{sup}) depends on the difference between redline temperature (T_{red}), which is the highest allowed temperature at the node inlets, and maximum node inlet temperature ($T_{in,max}$). In other words, we can use this temperature slack to increase the supply temperature and operate at higher CRAC efficiency without violating the temperature constraints. A new supply temperature is found by adding this difference to T_{sup} and cooling cost is calculated as follows:

$$T_{sup'} = T_{sup} + T_{red} - T_{in,max} \tag{5.9}$$

$$P_{AC} = \frac{P_c}{CoP(T_{sup'})} \tag{5.10}$$

The proposed thermal model provides fast results as it does not require time-consuming CFD simulations and it is able to capture the effect of recirculation, which has a significant contribution in high temperatures. The accuracy of the thermal model has been verified in prior work by comparison with CFD simulation results (Tang et al., 2006).

5.2.3 Temperature Model

As described in Section 5.2.2, the inlet temperatures of data center nodes are sufficient for computing the cooling cost. In order to evaluate the reliability of many-core systems in HPC data centers, it is necessary to calculate the junction temperature of processors. We calculate the junction temperature in two steps using a linear model. The first step is to calculate the heat sink temperature as in Equation (5.11).

$$T_{HS} = T_{HS,ref} + (T_{in} - T_{in,ref}) \cdot SF \tag{5.11}$$

where T_{HS} is the heat sink temperature, $T_{HS,ref}$ is the reference heat sink temperature, T_{in} is the node inlet temperature and $T_{in,ref}$ is the reference inlet temperature.

T_{HS} corresponds to the typical heat sink temperature at reference inlet temperature $T_{in,ref}$. For example, we take $T_{in,ref}$ as the supply temperature $T_{sup} = 15°\mathrm{C}$ and $T_{HS,ref} = 45°\mathrm{C}$. This means that when the inlet temperature is 15°C, we observe 45°C on the heat sink.

SF is a scaling factor determining the effect of T_{in} deviation from $T_{in,ref}$ on the heat sink temperature. For example, $SF = 0$ corresponds to the case for which, heat sink temperature stays constant with changing inlet temperature. We take SF as 0.6 as suggested in prior work (Walsh et al., 2010).

In the second step, we calculate the server junction temperature T_j as follows:

$$T_j = T_{HS} + P \times R_{ja} \tag{5.12}$$

where T_{HS} is the heat sink temperature as described in Equation (5.11). It is also called ambient temperature for junction temperature calculation. P is the processor power. We assume server power value of 350W which includes the total power for two processors, memory, interconnects etc. In order to calculate the junction temperature of a single processor, we use 120W for processor active power. R_{ja} is the junction to ambient thermal resistance and is typically 0.1 °C/W for a high quality heat sink. In the next section, we introduce the reliability model of many-core systems in HPC data centers which takes the junction temperature of processors as inputs.

5.3 Reliability Modeling for Many-core Systems in HPC Data Centers

In this section, we introduce a detailed reliability modeling approach to accurately model temperature-induced wear-out failure mechanisms for many-core systems with various system topologies. We also present the analysis results of the reliability of a real-life multi-chip many-core system by utilizing the reliability modeling approach.

5.3.1 Wear-out Failure Mechanisms

In our reliability model, we consider three major intrinsic wear-out failure mechanisms for processors: *Electromigration (EM)*, *Time Dependent Dielectric Breakdown (TDDB)*, and *Negative Bias Temperature Instability (NBTI)*.

EM occurs in Al and Cu interconnects due to the momenta exchange between current-carrying electrons and host metal atoms. **TDDB** is a wear-out mechanism of the gate dielectric. Failure occurs when a conductive path is formed through the gate oxide to substrate due to electron tunneling current. **NBTI** has become a critical reliability concern in advanced CMOS technology. NBTI typically occurs when the PMOS transistor is negatively biased, which results in the positive charges in the gate oxide. The positive charges cause an increase in threshold voltage and can lead to the wear-out failures (Srinivasan et al., 2005; JEDEC, 2006; Alam et al., 2007). In our reliability model, we do not consider thermal cycling failure mechanisms since thermal cycles of 140^oC magnitude are required to cause damage to the silicon substrate and interconnects (Srinivasan et al., 2003). In our experiments, we observe the maximum temperature cycling amplitude as less than 40^oC. Moreover, we focus on the silicon-level wear-out failure, while the effect of thermal cycling is mostly seen in the package and die interface.

The failure rates for these three failure mechanisms can be expressed in the following general form:

$$\lambda = \lambda^0 \times e^{\frac{-E_a}{kT}} \tag{5.13}$$

where k is the Boltzmann's constant which equals to 8.62×10^5. T is the temperature, and λ^0 is a material-dependent constant.

E_a is the activation energy for the failure mechanism. For Al alloys, we have $E_{a_{EM}} = 0.7eV$ (JEDEC, 2006). For TDDB, we set the activation energy as $E_{a_{TDDB}} = 0.75eV$ (JEDEC, 2006). The activation energy for NBTI is represented as $E_{a_{NBTI}} \times 1/n$, where n is the measured time exponent. We use $E_{a_{NBTI}} = 0.15eV$ and $n = 0.25$, which give the product of $0.6eV$ (JEDEC, 2006; Alam et al., 2007).

In order to determine the constants for λ^0_{EM}, λ^0_{TDDB}, and λ^0_{NBTI}, we assume the contributions of EM, TDDB, and NBTI are similar to each other at a base temperature. We calibrate the constants in each failure rate equation to satisfy a per-core mean time to failure (MTTF of 5 years at 60^oC (Ferreira et al., 2011).

5.3.2 Lognormal Distributions for Lifetime Reliability

The reliability models in some of the prior work assume all failure mechanisms have an exponential distribution (Coskun et al., 2009c; Srinivasan et al., 2004a; Coskun et al., 2006). The exponential distribution indicates a constant failure rate throughout the processor's lifetime. However, in practice, the wear-out failure mechanisms typically have a low failure rate at the beginning of the lifetime and the rate grows with the age of the components.

Recent work has shown that lognormal distribution constitutes a more accurate model of wear-out failure mechanisms compared to exponential distribution (Srinivasan et al., 2005; Xiang et al., 2010). The lognormal distribution provides the ability to model the dependence of the failure mechanisms on time. The probability density function for lognormal distribution is given in Equation (5.14).

$$f(t) = \frac{1}{t\sigma\sqrt{2\pi}} e^{-\frac{(\ln t - \mu)^2}{2\sigma^2}} \tag{5.14}$$

where μ and σ are the mean and the standard deviation of the underlying normal distribution, respectively. Reliability at time t can be computed by integrating $f(t)$ from 0 to t. In our reliability model, we use $\sigma = 0.5$ based on experimental data from prior work (Srinivasan et al., 2005).

We calculate the reliability of each wear-out failure mechanism using lognormal distribution to obtain the reliability of a processor at a certain time. However, since there is no closed-form solution for the integration of $f(t)$, it is difficult to find an explicit solution for the failure rate or reliability.

In order to address this issue, we consider Monte Carlo simulations to calculate the processor reliability. We make use of Monte Carlo simulations to combine the effects of the individual failure mechanisms and find the reliability of a single core. By utilizing Monte Carlo simulations, we first generate a normally-distributed random number, r_{normal}, with mean 0 and standard deviation of 1. r_{normal} is obtained using two independent uniformly distributed random numbers r_1 and r_2, as shown in Equation (5.15). We then generate a scaled normally-distributed random number $r_{snormal}$ with mean μ and standard deviation of σ from the normally-distributed random number as in Equation (5.16).

$$r_{normal} = sin(2\pi r_1)\sqrt{-2\ln(r_2)} \tag{5.15}$$

$$r_{snormal} = \mu + r_{normal}\sigma \tag{5.16}$$

After the scaled normal random number is obtained, a random lognormal distribution number $r_{lognormal}$ representing a random lifetime for each failure mechanism can then be generated from the scaled normal random number as in Equation (5.17).

$$r_{lognormal} = e^{r_{snormal}} \tag{5.17}$$

The mean of the normal distribution $r_{snormal}$ (μ) and the mean of the lognormal distribution $r_{lognormal}$ (MTTF) are related to each other as in Equation (5.18).

$$\mu = \ln(MTTF) - \frac{\sigma^2}{2} \tag{5.18}$$

To compute the reliability of a processor which is composed of the lognormally distributed failure mechanisms, we generate $r_{lognormal}$ distributions (i.e., random lifetimes) for each failure mechanism. We compute $r_{lognormal}$ by calculating the MTTF values using Equation (5.13) for each failure mechanism and μ using Equation (5.18). We conduct the experiment for 10^6 iterations to generate random lifetimes for failure mechanisms. At each iteration, the lifetime of the processor is set to the minimum of the generated numbers. MTTF of the processor is then calculated by averaging the minimums across all the iterations.

In order to convert the MTTF value to reliability, we generate the cumulative distribution function (CDF) of lognormal distribution. The reliability over time t for the lognormal distribution is then determined by Equation (5.19), where $F(t)$ is the

CDF of lognormal distribution at time t.

$$R(t) = 1 - F(t) \tag{5.19}$$

5.3.3 System Reliability Modeling

The modeling of system reliability in most of the prior work only considers series system topology (Srinivasan et al., 2004a; Xiang et al., 2010). In a many-core system with series topology, the first failure on any unit on the chip causes the entire processor to fail. However, in real-life computing systems, we may have different levels of series-parallel topologies.

$$Series: \quad R_{system}(t) = \prod_{i=0}^{n} R_i(t) \tag{5.20}$$

$$Parallel: \quad R_{system}(t) = 1 - \prod_{i=0}^{n}(1 - R_i(t)) \tag{5.21}$$

In a *series* system of n components, the system fails if *any* of its components fails. On the other hand, a *parallel* system with n components fails if *all* of its components fails. Assuming failure rates are statistically independent, the overall system reliability of a many-core system containing n cores with series topology can be computed as in Equation (5.20), while the overall system reliability of a many-core system containing n cores with parallel topology can be computed as in Equation (5.21).

5.3.4 Topology and System Reliability Analysis

In order to explore the effects of system topology on reliability, we consider an eight-core system that has two processors as our target architecture. The layout of the target system is illustrated in Figure 5·5 which is based on Intel Clovertown system. Each processor in our target system has four cores in two separate sockets. The two sockets are located on two chips which are put together in a single package.

For the target system, we investigate its system reliability with four topologies: (a) All series — all 8 cores connected in series; (b) Processor-level parallel — cores in series within each processor, parallel across processors; (c) Chip-level parallel — cores in series within each chip, parallel across chips; and (d) All parallel — all 8 cores in parallel.

Among the four scenarios, the system with all-parallel topology incurs higher design cost as additional hardware is needed to detect runtime core failures and initiate the recovery process for continued execution. The OS should also be equipped to safely reconfigure the system on failure. The additional design cost would be reduced for the system with chip-level parallel topology, because the parallelism is only at the chip-level. Processor-level parallelism, as in the system with processor-level parallel topology, can be implemented in today's clusters through using sockets that allow replacement of failed processors or using multiple server nodes.

For each scenario, we evaluate the system reliability with two different workload allocation strategies: *thermal balancing* and *clustering*. In *thermal balancing* work

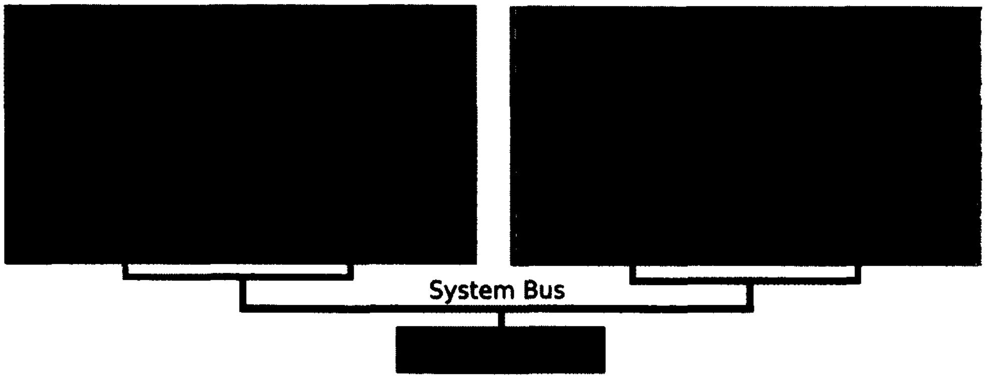
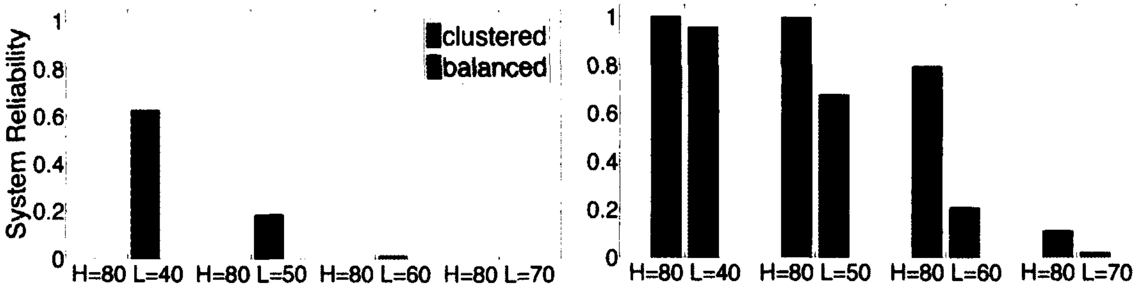


Figure 5·5: Layout of the Intel Clovertown System (Teng et al., 2009).

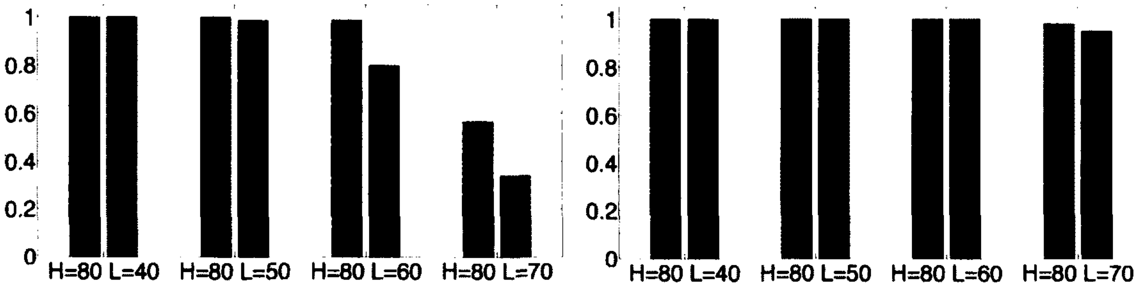
allocation, high-power loads are distributed across the chip (Coskun et al., 2009c; Mulas et al., 2008). In *clustering* workload allocation, power-hungry loads are allocated on neighboring cores. In each scenario, cores are assigned high (T_H) or low (T_L) temperatures.

We demonstrate the system reliability of clustered and balanced modes for each topology in Figure 5·6. In clustered mode, cores 0, 1, 2 and 3 have T_H and cores 4, 5, 6 and 7 have T_L. In balanced mode, cores 0, 2, 4 and 6 have T_H and the rest of the cores have T_L. However, in balanced mode, heat transfer between adjacent cores should be taken into account; thus, we assign T_B, the average of T_H and T_L, to all cores. This approximation has a few degrees error compared to detailed temperature simulations, but is sufficient to demonstrate the trends. We compute the system reliability using the reliability model described in Section 5.3. The core MTTF of 5 years at $60°C$ corresponds to a reliability value of 0.94.



(a) All series

(b) Processor-level parallel



(c) Chip-level parallel

(d) All parallel

Figure 5·6: System reliability for different series-parallel scenarios with T_H=80°C and per-core MTTF of 5 years at 60°C.

In our experiments, high temperature T_H is set as $80°C$ and low temperature T_L is swept from $40°C$ to $70°C$. Clustering degrades system reliability for *all series* scenario due to higher core temperatures. However, clustering improves reliability significantly for *processor-level parallel* system and moderately for *chip-level parallel* system. For *processor-level parallel* case, clustering provides system reliability of 0.999 and 0.995 for T_L values of 40°C and 50°C, respectively. For T_L of 60°C, it increases the system reliability from 0.2 to 0.8. Maximum increase in reliability (from 0.073 to 0.429) is seen at T_L of 65°C, which corresponds to 4.85X improvement.

We observe that, as the level of parallelism increases, system reliability for both clustered and balanced modes gets higher. Therefore, for *chip-level parallel* case, clustering is advantageous only at higher T_L values; while for *all parallel* case, it provides almost no improvement. In the rest of our analysis, we focus on *processor-level parallel* systems due to its ease of real-life implementation compared to other parallelism scenarios.

Figure 5·7 characterizes the relative reliability improvement of clustering compared to thermal balancing for the *processor-level parallel* system. We see that reliability

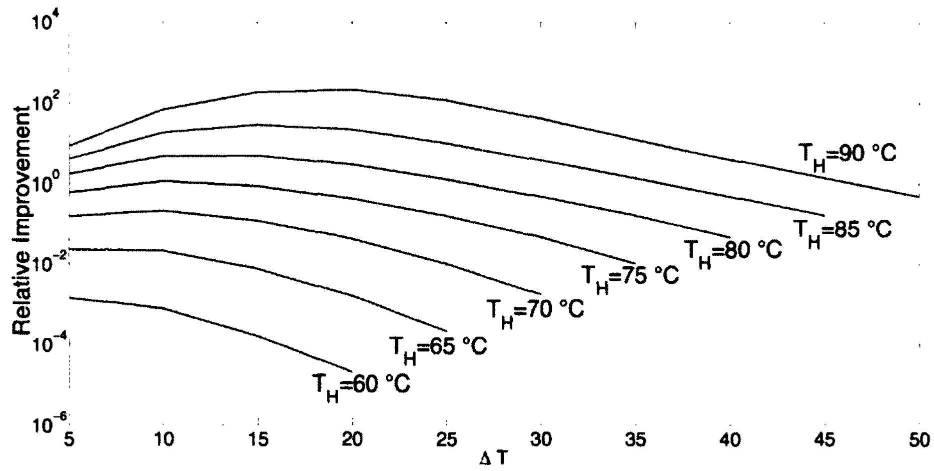


Figure 5·7: System reliability improvement of clustering over balancing $((R_{clustered} - R_{balanced})/R_{balanced})$ for various T_H, ΔT (T_H-T_L).

improvement starts becoming more noticeable for T_H values over 75°C. The relative improvement of 1 corresponds to doubling the reliability. As ΔT increases, reliability improvement first reaches a peak value and then drops. This is because for a fixed T_H value, increasing ΔT corresponds to lowering T_L, which eventually lowers T_B. Therefore, the system reliability for balanced mode increases, lowering the advantage of clustered mode.

Figure 5·8 compares the system reliability of clustering and balancing for initial per-core MTTF values of 3, 5 and 7 years at 60°*C*. As the MTTF value increases, reliability difference between clustering and balancing becomes smaller. This is expected since for example, going from MTTF of 5 years to 7 years, reliability of a core at 60°C increases from 0.9433 to 0.988. For the MTTF of 5 years, at T_H of 75°C, clustering improves reliability by 100%, while at 80°C the improvement is 4.85X. At lower MTTF values such as 3, clustering alone is not sufficient to achieve acceptable reliability levels, as the core reliability at 60°C drops to 0.712. In such cases, other reliability optimization techniques should be applied as well.

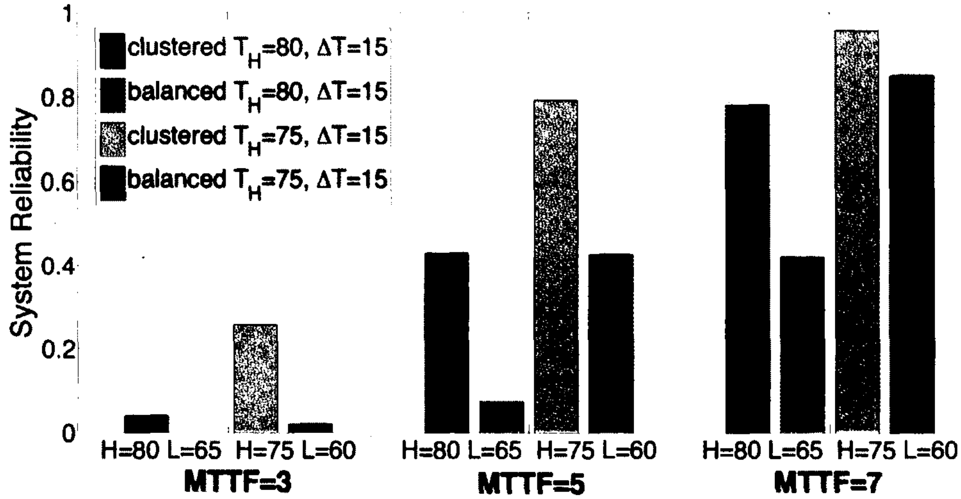


Figure 5·8: System reliability for a *processor-level parallel* system with initial per-core MTTF values of 3, 5 and 7 years.

5.4 Summary

This chapter presents the performance model, cooling energy model, and system-level reliability model for many-core systems in HPC data centers. In comparison to in many-core single-chip processors, high temperatures in data centers not only cause reliability degradation, but also increase the cooling energy consumption. On the other hand, the communication cost of applications has a significant impact on system performance.

In order to evaluate the communication cost of communication-intensive workloads running in HPC data centers, we have presented a performance model for mesh-connected parallel systems. We also have introduced a thermal model to evaluate the inlet temperature and cooling energy cost of HPC data centers.

To quantify system reliability, we have used a detailed reliability modeling approach to accurately model temperature-induced wear-out failure mechanisms under various system topologies. Utilizing the system-level reliability model, we have analyzed the reliability of a real-life multi-chip many-core system. Our analysis quantifies the tradeoffs between *clustering* higher power jobs and *thermal balancing* at various operating temperatures. We have shown that clustering can improve system reliability by up to $4.85X$ for systems with a processor-level parallel topology and 80^oC peak temperature.

In the next chapter, we propose a topology-aware reliability optimization policy that leverages the analysis from our system-level reliability model. Utilizing the evaluation results from our performance and cooling energy model, we also propose a job allocation methodology to jointly optimize the communication cost of HPC applications and the cooling energy in a data center.

Chapter 6

Runtime Management of Many-core Systems in Data Centers

6.1 Overview

As high performance computing moves towards exascale, performance, cooling cost and reliability have become serious concerns of many-core systems in HPC data centers. In addition to the modeling techniques as we discussed in Chapter 5, it is highly desirable to have dynamic management strategies that can effectively optimize performance, cooling energy, and system-level reliability of many-core systems in HPC data centers.

In the previous chapter, we have shown that *clustering* provides considerable reliability improvements in *processor-level parallel* and *chip-level parallel* systems compared to *thermal balancing*. Motivated by this analysis, we propose a topology-aware reliability optimization policy, *Globally Clustering Locally Balancing* (*GCLB*), where global refers to decisions across parallel nodes, and local refers to allocation decisions among a set of series nodes. We focus on the *processor-level parallel* scenario, as it is commonly employed in real-life multi-chip many-core systems.

Our topology-aware job allocation policy targets systems with medium to high utilization (e.g., as in high-performance computing clusters). We design low-cost predictors to estimate application power and chip peak temperature during allocation. Our policy adapts to workload changes while respecting thermal constraints. We provide an experimental validation using a large set of workload mixes representing different utilization levels and CPU usage profiles. Our policy improves the system reliability by up to 123.3% compared to temperature balancing policies. We also demonstrate the scalability of the proposed policy to larger systems.

In addition to reliability, performance and cooling costs are also critical aspects in data center management. Nearly half of the energy in the computing clusters today is consumed by the cooling infrastructure. It is possible to reduce the cooling cost by allowing the data center temperatures to rise; however, component reliability constraints impose thermal thresholds as failure rates are exponentially dependent on the processor temperatures. Data center performance is limited by highly parallel scientific, financial, or other applications that run on multiple nodes for long durations in the range of minutes, hours or days. The threads of these applications communicate with each other through communication infrastructures such as the message passing interface (MPI). The running time of a communication-intensive application is highly dependent on the location of the individual computing units that are communicating with each other.

We observe that existing algorithms for job allocation in HPC data centers address cooling efficiency and performance separately. How to optimize the performance and cooling energy tradeoffs achieved by these policies is currently an open question. In this chapter, we also propose a policy that reduces both cooling power and communication latency in an HPC data center. Experimental results demonstrate that cooling-aware policies alone do not minimize overall energy if the job allocation results in large communication overheads. Our joint optimization policy minimizes cooling cost along with the communication time, providing better performance-energy tradeoffs in HPC data centers.

6.2 Topology-Aware Reliability Optimization

The main idea of the *Globally Clustering Locally Balancing* (*GCLB*) algorithm is globally clustering high-power applications among parallel many-core processors and performing thermal balancing locally within a processor. This is because clustering across parallel nodes improves reliability; whereas for a set of series components, balancing results in higher reliability. We present a flow chart illustrating the *GCLB* optimization policy as in Figure 6·1. The *GCLB* policy periodically polls the performance counters and predicts the power consumption of each application using the performance counter data. We assign the jobs to cores according to their predicted power following the *GCLB* algorithm.

As shown in Figure 6·1, we check new job arrivals at every 10ms, which is the the typical scheduler tick in today's operating systems. We select a larger interval for GCLB, i.e., 50ms, to limit the performance impact of the policy. At 10ms intervals, we make intermediate heuristic decisions for job allocation. At 50ms intervals, the policy re-arranges the load across the processors if needed by migrating applications. Prior work has reported that cold-start overhead dominates the migration cost for SPEC

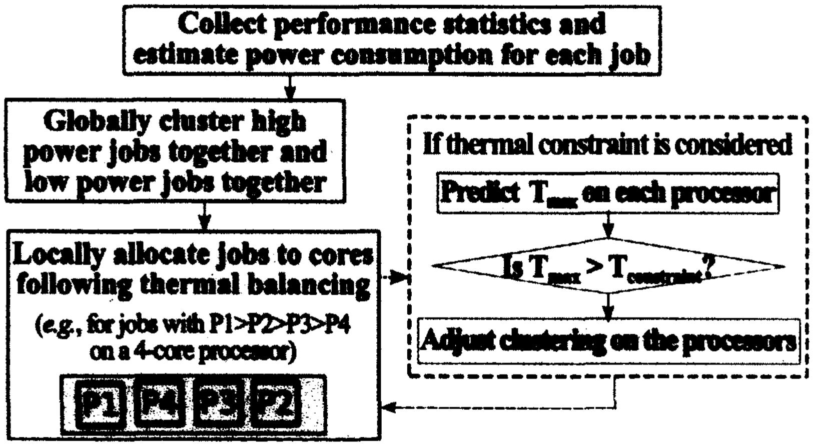

Collect performance statistics and estimate power consumption for each job
Globally cluster high power jobs together and low power jobs together
Locally allocate jobs to cores following thermal balancing
(e.g., for jobs with P1>P2>P3>P4 on a 4-core processor)
P1
P4
P3
P2
If thermal constraint is considered
Predict T_{max} on each processor
Is $T_{max} > T_{constraint}$?
Adjust clustering on the processors

Figure 6·1: A flow chart for illustrating the *GCLB* reliability optimization policy for *processor-level parallel* systems.

benchmarks, and the total migration overhead is less than 1ms (Coskun et al., 2009c). Assuming a similar overhead in our system, an interval of 50ms causes maximum 2% performance cost.

We assign newly arriving jobs to the idle cores on the system at every 10ms. In order to cluster higher power loads, we first assign new jobs to processors with a higher average power. If there is a thermal constraint, we predict the maximum processor temperature for the processor running the new job. If the maximum temperature is exceeded, we assign the new job to the processor with the next highest average power. At every 50ms, we apply the GCLB policy as follows: assuming the system has m cores, l parallel processors, and there are n jobs to be allocated (we assume $n \leq m$), we first estimate the power consumption for each job on the system. Then, we sort the power values for all the jobs. We group the sorted jobs into l groups: jobs with the highest power values are assigned to the first processor, the group with the second largest power values in the queue are assigned to the second processor, etc., until all the jobs are allocated.

After the jobs are clustered across parallel processors, within each processor, we locally balance the temperature across the series cores. The balancing method is based on thermal balancing policies in prior work (Coskun et al., 2009c), where high power jobs are assigned to expected cool locations on the chip, such as corner or side cores. Cooler jobs run in the central area, which is generally hotter. Figure 6·2 demonstrates

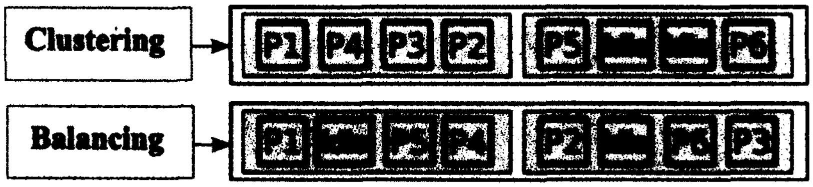


Figure 6·2: An illustration of the clustering and balancing job allocations on the target system under 75% utilization. P represents power consumption, and $P1 > P2 > ... > P6$.

the global clustering and balancing policies. Thermal balancing is applied to each processor locally.

In order to estimate the power consumption of each job, we collect performance statistics. We track instructions per cycle (IPC), number of floating point instructions, and number of integer instructions, as these metrics are strongly correlated with power consumption (Li and John, 2003). We collect the performance data using a simulator in our evaluation, while in a real system the statistics are collected through performance counters. We build a linear equation of the three performance counters using regression, and predict power consumption based on the equation. Experiments with 17 SPEC benchmarks show 4% prediction error using this method. Performance impact of power prediction is negligible, since computing a simple equation has very low computational cost.

Runtime temperature prediction techniques have been proposed in recently work (Ayoub and Rosing, 2009). In our optimization strategy, we choose a simple temperature prediction method using a linear model as we solely want to estimate the maximum temperature on a processor. For inputs to the predictor, we use power estimates for each core and absolute power differences between adjacent cores to take the heat sharing and core locations into account. We collect 100 sets of simulation results from the SPEC 2006 workloads, and validate the predictor against HotSpot simulations. Our peak temperature prediction results in maximum 8% error in comparison to HotSpot simulation results, with less than 2^oC error for most cases. For example, for processor 0 in Figure 5·5, we choose $P0$, $P1$, $P2$, $P3$, $|P0-P1|$, $|P1-P2|$, and $|P2-P3|$ as the inputs to a linear regression fit. The reason for these choices is that the peak core temperature on a many-core processor does not only depend on the power consumption of the cores, but also depends on the power differential of the adjacent cores.

GCLB algorithm can work with temperature constraints using the thermal predictor. This is important as clustering high-power workloads may result in high peak temperatures on a processor. In addition to critical thermal thresholds determined by the manufacturer, thermal constraints could be imposed by user-defined target per-core MTTF values or by cooling optimization policies.

During allocation, if the thermal constraints are not satisfied, we adjust job allocation by swapping the hottest jobs across processors and locally balance temperature after swapping. This process is repeated (a job moved once is not moved again) until the thermal constraint is met. In our algorithm, we assume we can always find a schedule that meets thermal constraints, which is a reasonable assumption for most commercial systems.

It is also possible to integrate the proposed GCLB policy with DVFS policies. Integration with DVFS can provide energy savings as well as fine tuning of the operating conditions to meet temperature or performance constraints. Hybrid policies integrating various DVFS and job allocation strategies have been designed in prior work (Coskun et al., 2009c). While cooling is mostly designed with large safety margins in commercial systems, energy-efficient cooling methods are likely to leverage temperature constraints lower than the absolute critical levels.

6.2.1 Experimental Methodology

We model the target system for evaluating the system reliability based on the core microarchitecture of Intel Clovertown. In order to evaluate the performance of our target system, we use M5 (Binkert et al., 2006) to build the performance simulation infrastructure. We use the system-call emulation mode in M5 with X86 instruction set architecture (ISA). We fast-forward each benchmark for 1 billion instructions and then execute with the detailed out-of-order CPUs for 100 million instructions to collect the detailed performance statistics.

Table 6.1: Intel Clovertown core architecture parameters.

Technology	65 nm
CPU Clock	2.66 GHz
Issue Width	4-way out-of-order
Functional Units	3/2 Int/FP ALU
	1/1 Int/FP Mult
Physical Regs	128 Int, 128 FP
RAS / ROB size	16 /96 entries
Load /Store Queue	32 / 20 entries
L1 I/DCache	32 KB, 8-way, 64B-block
L2 Cache(s)	4 MB, 16-way, 64B-block

The architectural parameters for cores and caches of our target system are listed in Table 6.1. These parameters are used for the target system configuration in our architecture level performance and power simulations.

In order to compose our workloads, we select 17 applications from the SPEC 2006 benchmark suite. Among the 17 SPEC benchmarks, 10 applications are integer (INT) benchmarks (*astar, bzip2, gcc, gobmk, h264ref, hmmer, libquantum, mcf, omnetpp, specrand_int*) and 7 applications are floating point (FP) benchmarks (*bwaves, cactusADM, dealII, GemsFDTD, lbm, namd, specrand_fp*).

We further classify these benchmarks according to their performance and memory boundedness. They are named *INT-Hmem, INT-Lmem, INT-HIPC, INT-LIPC, FP-Hmem, FP-Lmem, FP-HIPC, FP-LIPC, and Mixed*, where *Hmem* or *Lmem* means workloads with high or low memory access rates, *HIPC* or *LIPC* means workloads with high or low IPC. The workload is classified based on the instructions per nano-second (IPnS) and memory accesses per second (MemAcc) form the performance simulation. This classification is because that IPC is a common performance metric for many-core processors and MemAcc is a metric for illustrating the behavior of memory bounded

Table 6.2: Workload characteristics.

Workload	**Benchmarks**
INT-Hmem-1	mcf, mcf, bzip2, mcf
FP-Hmem-1	lbm, bwaves, lbm, lbm
INT-LIPC-1	mcf, astar, mcf, bzip2
FP-LIPC-1	lbm, bwaves, lbm, cactusADM
INT-Hmem-2	bzip2, hmmer, mcf, libquantum
FP-Hmem-2	lbm, bwaves, namd, cactusADM
INT-LIPC-2	mcf, gcc, bzips, libquantum
FP-LIPC-2	lbm, cactusADM, bwaves, lbm
Mixed_1	mcf, omnetpp, lbm, dealII
Mixed_2	gcc, gobmk, GemsFDTD, cactusADM
INT-Lmem-1	astar, specrand_int, h264ref, specrand_int
FP-Lmem-1	specrand_fp, dealII, namd, specrand_fp
INT-HIPC-1	specrand_int, omnetpp, omnetpp, h264ref
FP-HIPC-1	specrand_fp, dealII, dealII, namd
INT-Lmem-2	specrand_int, specrand_int, astar, specrand_int
FP-Lmem-2	specrand_fp, specrand_fp, dealII, specrand_fp
INT-HIPC-2	omnetpp, specrand_int, omnetpp, omnetpp
FP-HIPC-2	dealII, dealII, dealII, specrand_fp

benchmarks. Table 6.2 presents the classifications of our workloads.

We use McPAT 0.7 (Li et al., 2009) for 65nm process to obtain the runtime dynamic power of the cores. We set V_{dd} to 1.1V and operating frequency to 2.66GHz. The L2 cache (4 MB) power is calculated using Cacti 5.3 (Thoziyoor et al., 2008) as 5.06W. We calibrate the McPAT runtime dynamic core power using the published power for Intel Xeon Processor X5355. At 343K, we assume the leakage power for the cores is 35% of the total core power. We also model the temperature impact on leakage power using an exponential formula (Srinivasan et al., 2004a).

Table 6.3: Dimensions of the target system.

Dimensions	Length	Width	Area
Chip	$19.07mm$	$15mm$	$286mm^2$
Core	$4mm$	$9mm$	$36mm^2$
L2 Cache	$6mm$	$8mm$	$36mm^2$

We run HotSpot 5.0 (Skadron et al., 2003) for thermal simulations. We set the chip and package parameters using the default configuration in HotSpot to represent efficient packages in high-end systems. All thermal simulations use the HotSpot grid model for higher accuracy and are initialized with the steady-state temperatures. The chip and core areas are obtained from the published data for Intel Clovertown systems. The L2 cache area is estimated by using Cacti 5.3 (Thoziyoor et al., 2008). The detailed dimension for each component that we used in the HotSpot simulations are listed in Table 6.3.

6.2.2 Evaluation Results

We evaluate *GCLB* on the target Intel Clovertown system for three different workload utilization scenarios: high utilization, medium utilization, and low utilization, and use 75%, 50%, 25% workload utilizations as examples to represent each scenario, respectively. Figure 6·2 compares the clustering and balancing allocation policies at 75% utilization. System reliability of the clustering and balancing policies for all the workloads running on the target system with 75% workload utilization is shown in Figure 6·3. We observe that the proposed *GCLB* policy provides up to 123.3% improvement in system reliability compared to the thermal balancing policy.

Among all the workloads, the *HIPC* and *Lmem* applications have higher system reliability improvement. This is because the *HIPC* and *Lmem* applications have higher power densities causing higher temperatures. Local thermal balancing has

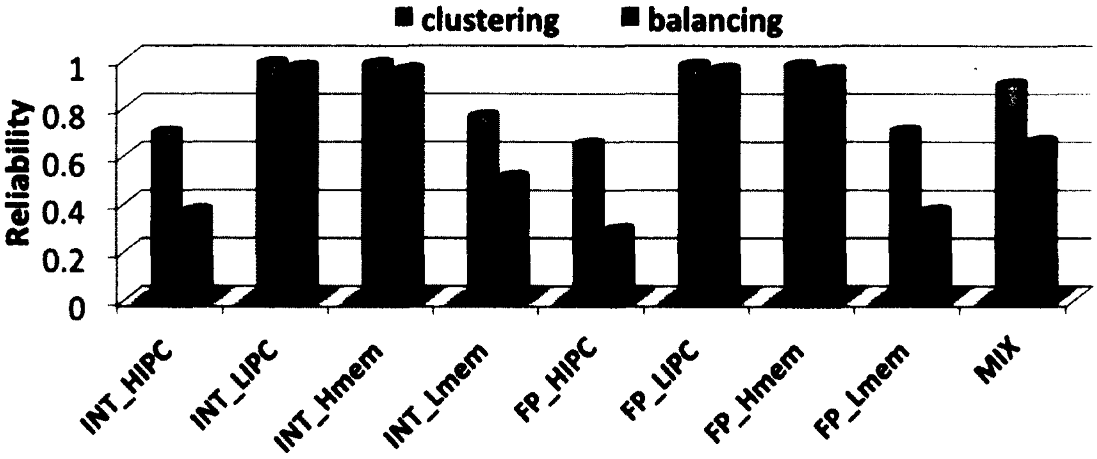


Figure 6·3: System reliability with *GCLB* and thermal balancing allocation policies for the target system under 75% utilization.

up to 27.2% reliability improvement compared to not balancing allocation within a processor. As local balancing always outperforms locally imbalanced scenarios, we do not report results for locally imbalanced cases in the rest of the results.

The system reliability for the clustering and balancing allocation policies on the target system with 50% workload utilization is presented in Figure 6·4. This medium utilization level is representative of the workload utilization in data centers. The

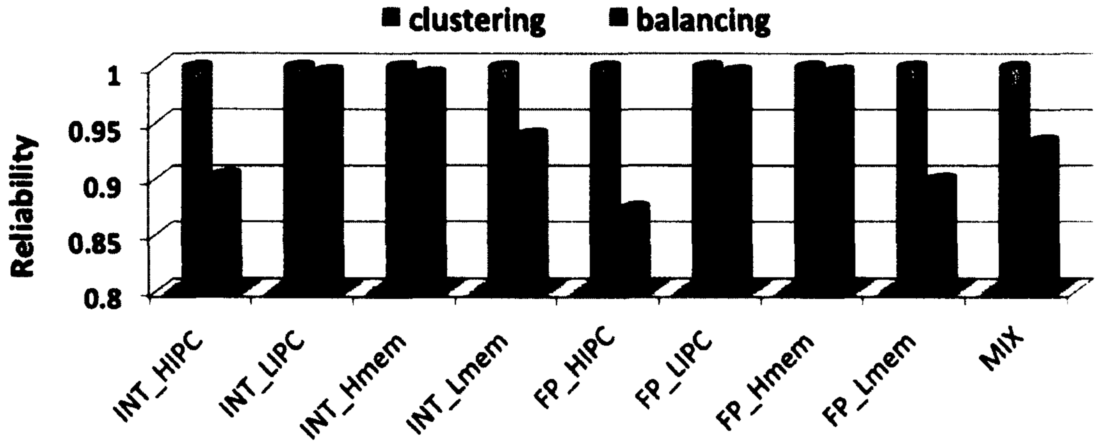


Figure 6·4: System reliability with *GCLB* and thermal balancing allocation policies for the target system under 50% utilization.

job allocations for the 50% workload utilization is similar to the illustration shown in Figure 6·2, while the $P5$ and $P6$ change to idle cores. We see that with 50% workload utilization, we achieve up to 14.3% improvement in the system reliability in comparison to thermal balancing policy. We also conduct the same analysis on the target system with 25% workload utilization. The low workload utilization scenario happens when data centers run fewer jobs (e.g., at night). In this case, clustering and balancing achieve similar reliability.

From our experimental results, we observe that when *GCLB* is applied without considering thermal constraints, peak temperature at 75% utilization is between 63.8^oC and 76.33^oC. Figure 6·5 illustrates the system reliability with *GCLB* optimization policy compared to the thermal balancing policy at 75% utilization, using a thermal constraint of 75^oC. We notice that the reliability improvement of *GCLB* decreases for some workloads, such as *FP_HIPC*. This is because GCLB moves some of the higher power jobs to lower power processors to meet the constraint, and becomes more similar to balancing.

We also explore the *GCLB* policy for dynamically changing workloads. We gen-

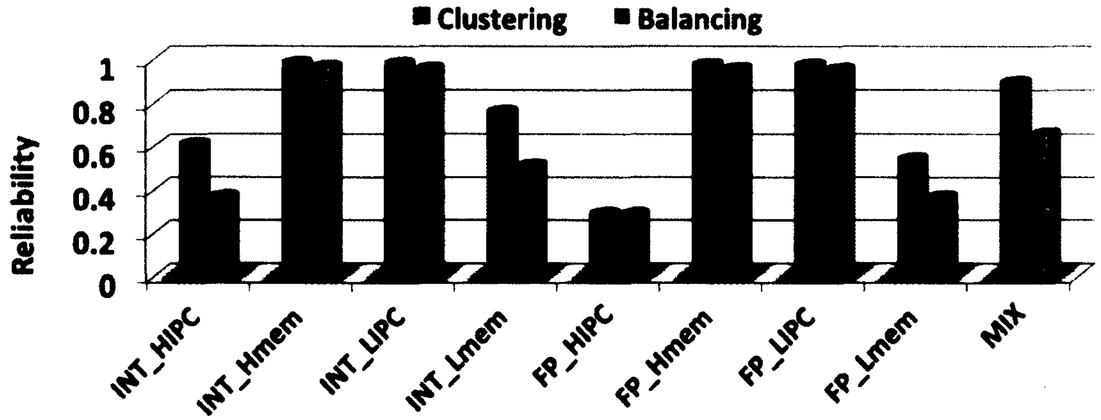


Figure 6·5: System reliability for *GCLB* optimization policy compared to thermal balancing for systems with 75% utilization, considering a thermal constraint of 75^oC.

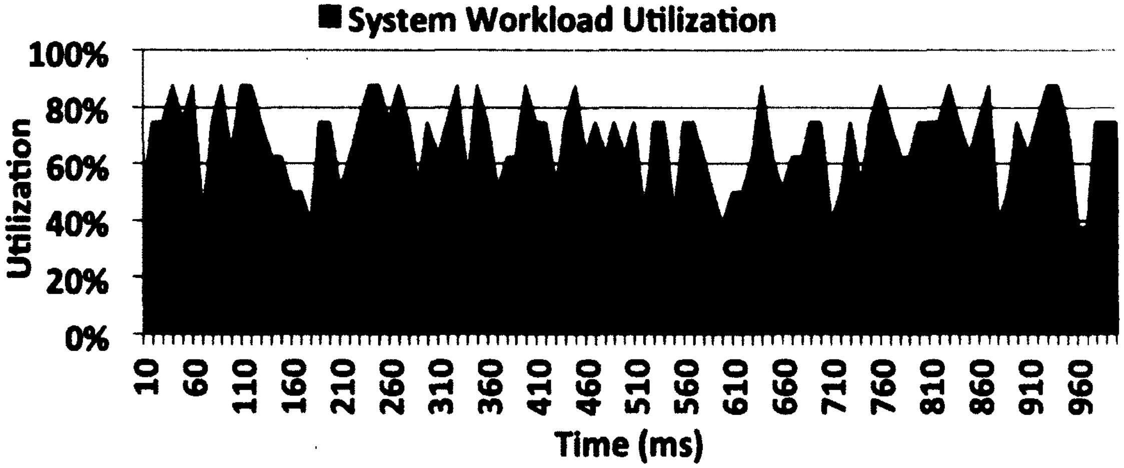


Figure 6·6: Temporal workload utilization for the target system.

erate a random workload utilization scheme which changes every 10ms with a total simulation time of one second. The temporal workload utilization for the target system is illustrated in Figure 6·6. The average workload utilization is 68%. The jobs running on the system are randomly selected among the 17 SPEC benchmarks. Figure 6·7 shows that allocating jobs according to *GCLB* policy improves reliability by 27.3% on average compared to random workload allocation. Figure 6·7 also

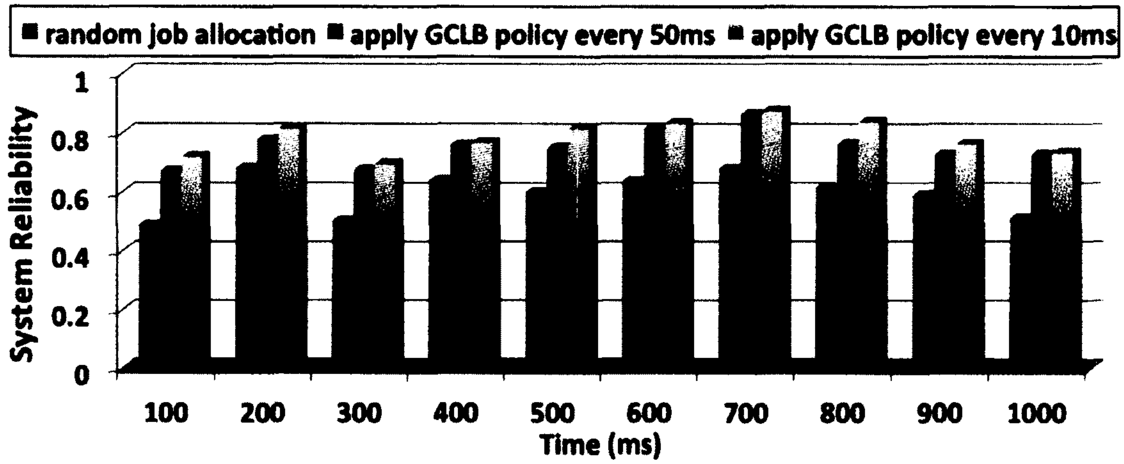


Figure 6·7: System reliability of *GCLB* compared to random job allocation for dynamically changing workload utilization.

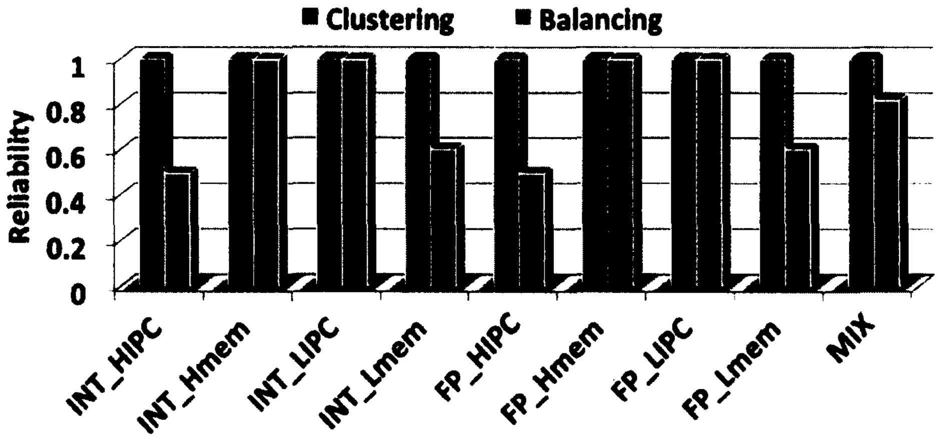


Figure 6·8: Exploration of 16-core system reliability with *GCLB* and thermal balancing allocation policies under 75% utilization.

shows that, if the *GCLB* optimization policy is applied every 10ms without considering thread migration overhead, the average system reliability improvement is 32.9%. However, as discussed in Section 6.2, migrating threads every 10ms would cost up to 10% system performance overhead. Our reliability optimization policy achieves comparable reliability improvement with less than 2% performance cost.

In order to evaluate the scalability of the *GCLB* optimization policy, we extend our analysis to a 16-core system with 4 parallel processors and 4 cores (in series) on each processor. System reliability for the 16-core system running *GCLB* compared to thermal balancing is presented in Figure 6·8. We observe that GCLB policy provides system reliability of close to 1 for all the benchmarks, and improves reliability by up to 101.7% in comparison to thermal balancing. This is because scaling to a higher number of processors provides increased parallelism and higher degree of freedom for more efficient task scheduling. For example, for the 16-core system with 75% utilization, using "clustering" assigns all the "idle" cores in one processor, which increases system reliability.

6.3 Joint Performance and Cooling Cost Optimization for Data Centers

In this section, we introduce a job allocation methodology to jointly optimize the communication cost of HPC applications and the cooling energy in a data center. We first formulate and solve the cooling energy optimization and communication cost optimization problems individually. For cooling cost minimization, we use the Minimize Peak Inlet Temperature (MPIT) algorithm (Tang et al., 2008); for communication cost minimization, we deploy the MC1X1 algorithm (Bender et al., 2008). We then propose a job allocation algorithm, which takes both cooling efficiency and communication latency into consideration. We also discuss how reliability constraints can be included in the job allocation optimization.

6.3.1 Performance-aware Job Allocation

The objective of performance-aware (i.e., communication cost-aware) job allocation is to assign a job to a set of available nodes on a target system such that the average number of communication hops between the nodes is minimized. The target system in this thesis is a mesh-connected HPC cluster, as discussed in Section 5.2. We formulate the performance-aware job allocation problem in Equation (6.1).

$$\begin{aligned} \underset{X_{job}}{\text{minimize}} \quad & CC_{job}(X_{job}) \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i = n \qquad x_i \in \{0,1\} \end{aligned} \tag{6.1}$$

where N=40 is the number of total nodes within the data center and n is the total number of nodes required by a job. X_{job} is a vector described as $X_{job} = \{x_1, x_2, ..., x_N\}$, where x_i $(i = 1, \ldots, N)$ represents whether node i is assigned the *current job* or not. It shows the selected nodes to allocate the *current job*, so n of its elements are 1 and the rest is 0. CC_{job} represents the communication cost of a job running on the target system as introduced in Section 5.2.1. Based on Equation (5.1), $CC_{job}(X_{job})$ can be formulated as:

$$CC_{job}(X_{job}) = \frac{1}{n} \sum_{(x_i, x_j) \in X_{job}} (x_i \cdot x_j)[w_x(x_i, x_j) + w_y(x_i, x_j)] \tag{6.2}$$

where n is the number of nodes a job requires and $(x_i,\ x_j)$ $(i, j = 1, \ldots, m)$ stands for a pair of source and destination nodes that a message is passing through.

We use the MC1X1 algorithm (Bender et al., 2008) to minimize the communication cost, as it aims at minimizing the pairwise L1 distance across the communication nodes and provides acceptable results for *all-to-all* communication pattern. It is also easily adaptable to the systems that do not require user information about the request processors in a particular shape, such as the Cplant system at Sandia National Laboratory (Leung et al., 2002).

The MC1X1 allocation algorithm tries to confine the allocated jobs into the smallest possible area. A rectangular-shaped area, in which all the assigned nodes are ideally confined, is called a *shell*. The node located at the center of the shell is called the shell center. For an incoming job, MC1X1 traverses the data center layout and finds shells of different centers and sizes among the available (idle) nodes. During this traversal, MC1X1 records a score for each node, where the score is the size of the smallest possible shell centered at that node. The decision of which node to select as a shell center depends on its score. A lower score indicates a smaller shell area, leading to a more compact allocation with lower communication cost.

6.3.2 Cooling-aware Job Allocation Policy

The optimization of cooling energy cost is achieved when the maximum inlet temperature $\{T_{in}\}$ in the data center is minimized (Tang et al., 2008). Therefore, a cooling-aware allocation policy assigns jobs to nodes so that the resulting $\max\{T_{in}\}$ will be minimum. We use the algorithm named Minimize Peak Inlet Temperature (MPIT) algorithm that is proposed in prior work (Tang et al., 2008). We formulate the optimization problem of allocating a job to an idle data center with minimal cooling energy as follows:

$$\begin{aligned} \underset{X_{dcenter}}{\text{minimize}} \quad & max\{T_{in}(X_{dcenter})\} \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i = n_{dcenter} \quad x_i \in \{0,1\} \end{aligned} \tag{6.3}$$

where $X_{dcenter}$ is a vector described as $X_{dcenter} = \{x_1, x_2, ..., x_N\}$, where x_i $(i = 1, \ldots, N)$ represents whether node i is assigned any job or not. Vector $X_{dcenter}$ shows all of the busy nodes in the data center corresponding to *currently* and *previously* allocated jobs. $n_{dcenter}$ is the sum of the sizes of all jobs running on the data center. Rest of the parameters are defined the same as in Equation (6.1). T_{in} represents the inlet temperature of a system which is defined in Equation (6.4).

$$T_{in}(X_{dcenter}) = T_{sup} + D \cdot P_{idle} + D \cdot X_{dcenter} \cdot P_{util} \tag{6.4}$$

where T_{sup} is the CRAC unit supply temperature, D is heat distribution matrix. P_{idle} and P_{util} are the idle and dynamic power for the nodes. Note that, in order to allocate a second job to a busy data center, we use additional constraints to represent the currently busy nodes. For example, if nodes 1, 2 and 3 are busy at the time of allocation, we add the constraints x_1=1, x_2=1, x_3=1 to solve the problem.

As described in Section 5.2.2, cooling cost is highly dependent on the CRAC supply temperature T_{sup}. If we can increase T_{sup} as much as possible without causing the nodes to exceed the redline temperature, we can save power. Therefore, the maximum allowed T_{sup} increase is limited by the maximum inlet temperature $\max\{T_{in}\}$.

We implement the optimization problem in Matlab. The *fminimax* function in Matlab returns a real number solution x_{real}. We use the discretization algorithm suggested in (Tang et al., 2008) to convert it to the nearest integer solution x_{int} which obeys the constraints. This algorithm was shown in (Tang et al., 2008) to give the highest power savings among various other approaches. x_{real} is the optimum solution to the defined linear programming problem and x_{int} is an integer solution close to the optimum. For various allocations, we compare the $\max\{T_{in}\}$ of both real and integer solutions and they are the same to the second decimal point.

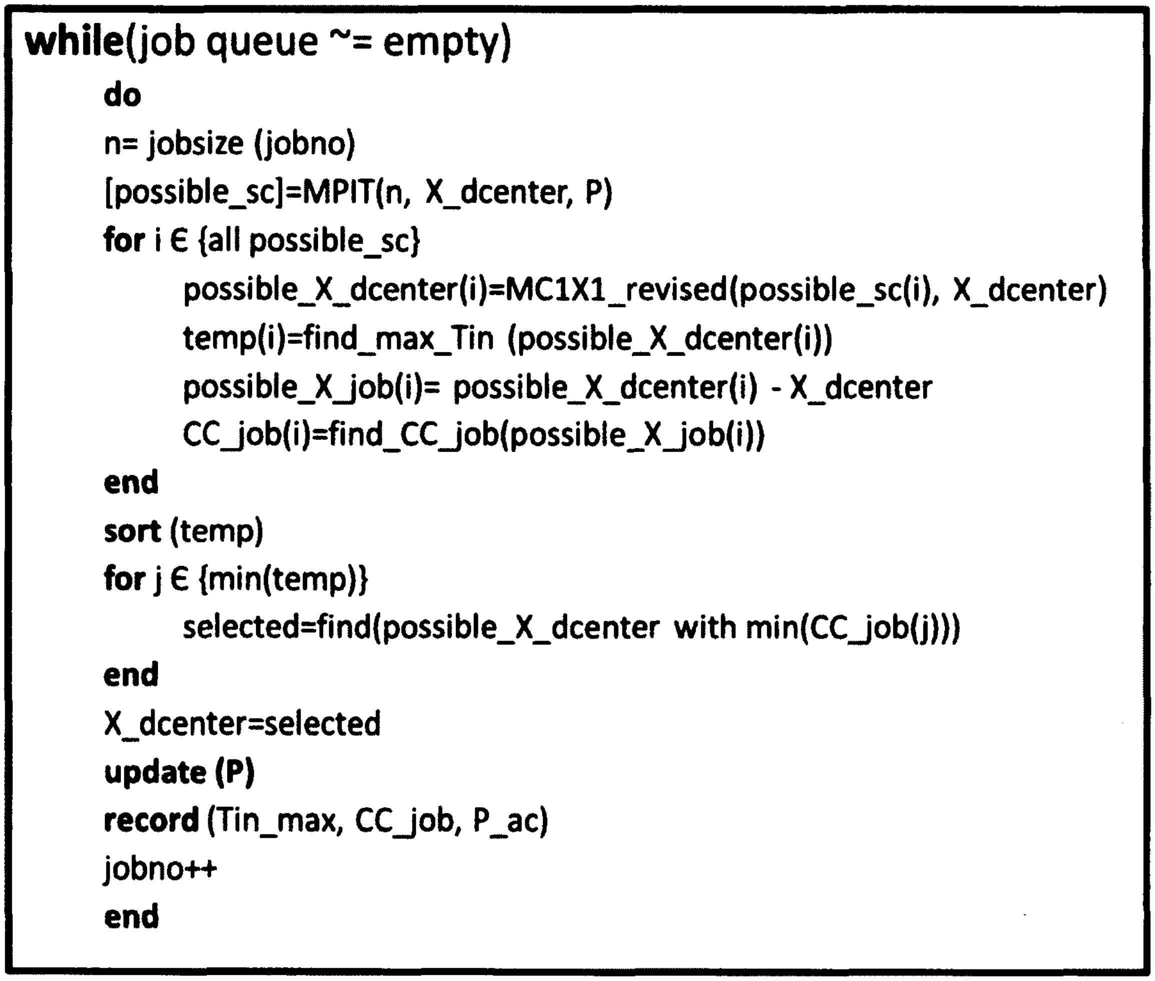
6.3.3 Joint Optimization Policy for Data Center Job Allocation

Cooling-aware and performance-aware policies optimize cooling power and communication latency independently, which means that the resulting allocations may not be successful when both objectives are considered simultaneously. Cooling-aware job allocation is mostly affected by the layout of the data center as the recirculation effect changes depending on the location of the active nodes. In most cases, cooling-aware policy allocates jobs to the nodes located far from each other. For example, for a job of size 4, cooling-efficient allocation distributes the job equally among the data center rows in order to minimize the peak inlet temperature. This causes very high communication latency for cooling-aware policy. On the other hand, performance-aware MC1X1 policy confines the nodes of each allocated job into the smallest possible *shell*. It follows a regular pattern to allocate the jobs in the data center and arbitrarily breaks ties. It does not care about whether an allocation results in high temperature as long as the allocated nodes are within the smallest shell possible, potentially causing inefficient cooling.

In order to jointly optimize the cooling energy cost and communication cost of applications running in an HPC data center, we design a heuristic algorithm combining both cooling-aware and performance-aware policies. Our algorithm first considers cooling-aware job allocation solution, and then uses the resulting nodes as candidates for shell centers to apply the performance-aware job allocation policy. Then, we break the ties of possible performance-aware job allocations by selecting the allocation with minimal peak inlet temperature.

Our algorithm first checks which nodes the cooling-aware policy would allocate the job to when a job arrives. These nodes are called as *possible shell centers.* Then, we feed the locations of these *possible shell centers* to the MC1X1 algorithm to minimize communication cost. We modify the MC1X1 algorithm to make it open a shell centered at a given input node (possible shell center) accordingly. In MC1X1, opening a shell centered at a node refers to finding the smallest square-shaped area to include all nodes of a job. Starting from the smallest shell (1 square unit), the number of available nodes in the shell are checked. If the size of the job is larger than the available nodes, shell is expanded.

Our algorithm examines whether there are multiple allocation options within the shell area when the available node count is met. For example, assume that we have a shell with 9 nodes, 3 of whom are busy, and we will assign a job of size 4 to the rest. In this case, we choose the 4 nodes with minimum communication cost possible. The resulting selection is the possible allocation corresponding to that *possible shell center.* For each *possible shell center*, revised MC1X1 algorithm gives an allocation vector, *possible_X_dcenter.* Among those vectors, we select the most cooling efficient one (i.e., resulting in smallest peak inlet temperature). For example, assume that for a job i of size 3, cooling-aware policy assigns the job to nodes 1, 4, and 5. We open shells centered at those nodes and select the one with the smallest inlet temperature.



```
while(job queue ~= empty)
    do
    n= jobsize (jobno)
    [possible_sc]=MPIT(n, X_dcenter, P)
    for i ∈ {all possible_sc}
        possible_X_dcenter(i)=MC1X1_revised(possible_sc(i), X_dcenter)
        temp(i)=find_max_Tin (possible_X_dcenter(i))
        possible_X_job(i)= possible_X_dcenter(i) - X_dcenter
        CC_job(i)=find_CC_job(possible_X_job(i))
    end
    sort (temp)
    for j ∈ {min(temp)}
        selected=find(possible_X_dcenter with min(CC_job(j)))
    end
    X_dcenter=selected
    update (P)
    record (Tin_max, CC_job, P_ac)
    jobno++
    end
```

Figure 6·9: Joint optimization algorithm.

For the cases where two or more allocations result in the same inlet temperature but different communication costs, we find and choose the job allocation that results in the smallest communication cost.

The flow of the joint optimization algorithm is illustrated in Figure 6·9. MPIT and MC1X1_revised are the cooling-aware and revised performance-aware algorithms, respectively. *X_dcenter* and *P* are the vectors holding the current busy nodes information and the power values. *Possible_sc* is the possible shell center and *CC_job* stands for the job communication cost. *Possible_X_dcenter* is the vector of busy nodes that will result from the possible allocation. *Possible_X_job* vector shows which nodes will be assigned to the job. Note that the joint policy is scalable to larger data centers. The only parameters to change for a different data center are the cross-interference coefficient matrix and the power values for the nodes.

We also consider our policy with reliability constraint. If the user or the administrator wants to add a minimum MTTF constraint to the joint policy, we check what the resulting MTTF value for each processor would be before every allocation decision. To compute these MTTF estimates, we first compute the resulting inlet temperatures for that allocation using Equation (5.2). Next we compute junction temperatures as described in Section 5.2.3. Finally, we compute processor MTTF as explained in Section 5.3. If the current allocation is expected to result in an MTTF value lower than the given threshold for any processor, we stall the allocation and wait for some of the existing jobs to finish.

6.3.4 Experimental Results

In this section, we present the experimental results for the three different allocation strategies: cooling-aware, performance-aware and our joint optimization technique. We first demonstrate the job allocation decision of each strategy on a single row of the data center. We then experiment with multiple-row allocation for our target data center with 40 nodes. We also compare our joint allocation policy against the cooling-aware and performance-aware policies under dynamically changing workload.

Single-row Job Allocation

In the single-row job allocation test case, we assume four jobs to be allocated sequentially. The jobs have sizes of $4, 5, 6$ and 3 nodes, respectively. Figure 6·10 illustrates how each policy assigns the jobs to the nodes. Red and blue colors respectively represent busy and free nodes. The numbers in the circles show which jobs are running on the nodes. Cooling-aware policy assigns jobs to the nodes located at the right side

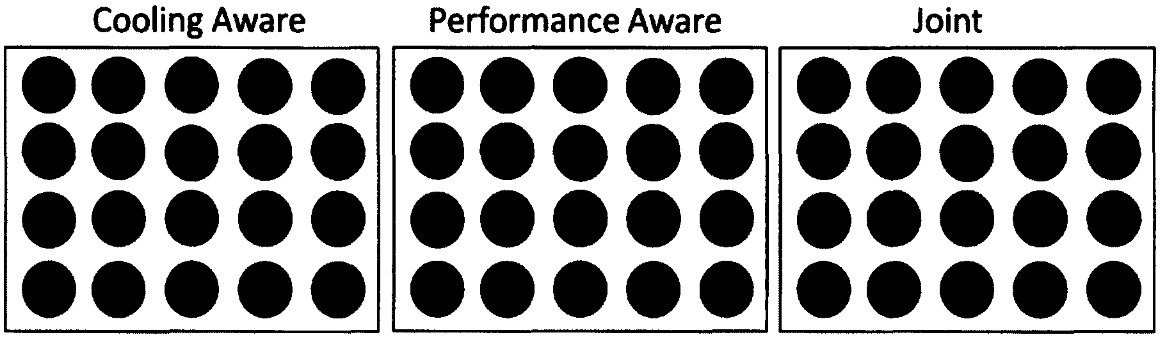


Figure 6·10: Allocation scheme for the three policies.

of the data center and avoids the nodes that are high recirculation contributors. This result is in parallel with previous the characteristics of our data center as shown in Figure 5·4.

Communication-aware policy, on the other hand, tries to confine the allocated nodes to the smallest area possible. Therefore, the resulting allocation for each job is more compact. Our joint allocation policy finds the cooling-efficient areas and assigns the jobs to the nodes as close to each other as possible without causing notable temperature increases. Joint policy does not always result in the same minimum inlet temperature as the cooling-aware policy, but follows closely.

Table 6.4 shows the percentage of active nodes, maximum inlet temperatures ($maxT$) in ^oC, individual job communication cost (CC), and cooling power (P) in

Table 6.4: Simulation results for the single-row job allocation.

Policy		**Perf-aware**			**Cooling-aware**			**Joint-opt**		
Job	**Util**	**CC**	$maxT$	P	**CC**	$maxT$	P	**CC**	$maxT$	P
Job1	20%	4.0	25.0	9.4	4.0	19.9	6.3	4.0	19.9	6.3
Job2	45%	6.4	25.1	13.4	9.6	20.5	9.4	8.0	20.3	9.2
Job3	75%	8.3	32.1	35.8	13.3	23.3	15.6	14.7	23.3	15.6
Job4	90%	2.7	32.1	40.4	5.3	28.1	27.0	2.7	28.5	28.0

kW for all the three allocation schemes. As we can see in Table 6.4, performance-aware policy gives the lowest job communication cost (CC) for each job; however, it reaches the high inlet temperatures very fast. Cooling-aware policy keeps the temperatures low, but results in very high communication latency for all the jobs. As expected, our joint policy's performance is in between the two policies.

Multiple-row Job Allocation

In order to evaluate the job allocations across the multiple rows of the data center, we use a job sequence that is similar to the sequence in the previous experiment. Figure 6·11 shows the percentage of the active nodes and the size of each job in terms of number of nodes required. Figure 6·12 shows the cooling power over time for the three allocation policies. Joint policy follows the cooling-aware policy closely and all policies converge at the 100% utilization point. However, performance-aware allocation reaches high cooling power values much faster than the joint policy.

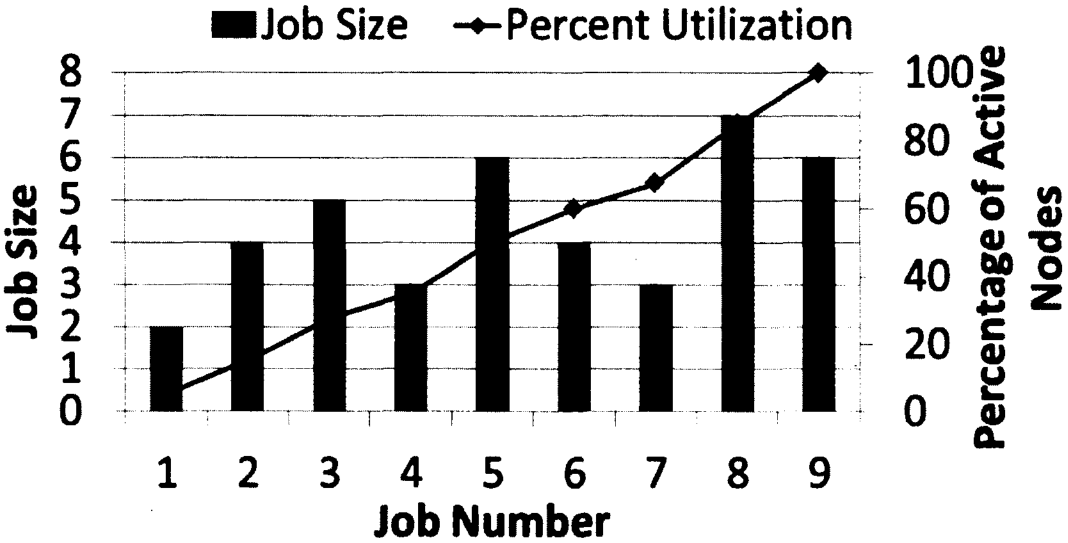


Figure 6·11: Job sizes and percentage of active nodes for multiple-row allocation.

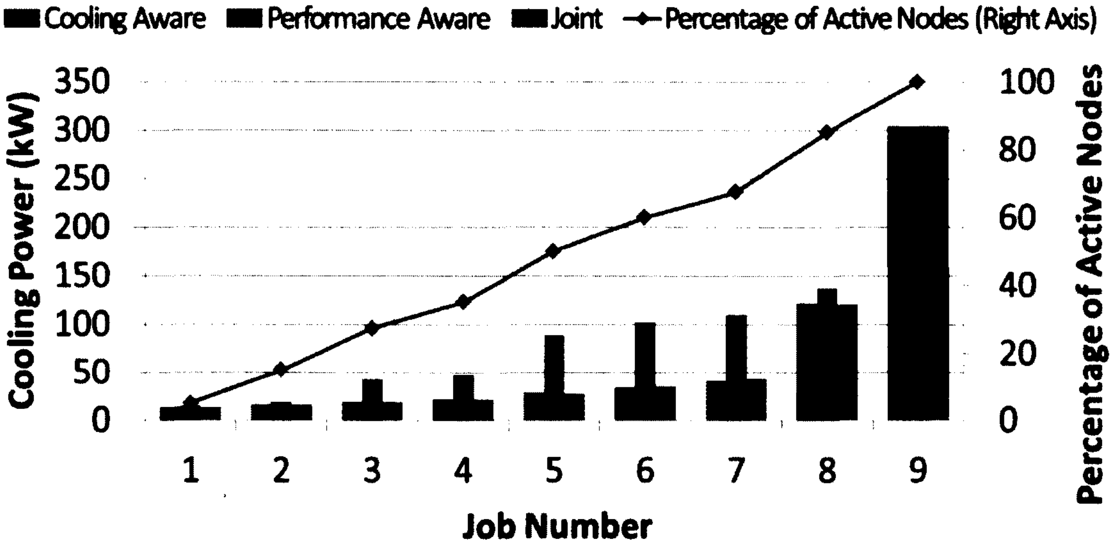


Figure 6·12: Cooling power for multiple-row allocation.

In Figure 6·13, we present the communication cost of each job and observe that cooling-aware assignment results in high communication cost. The reason is that the cooling-aware assignment distributes the jobs across different rows to minimize inlet

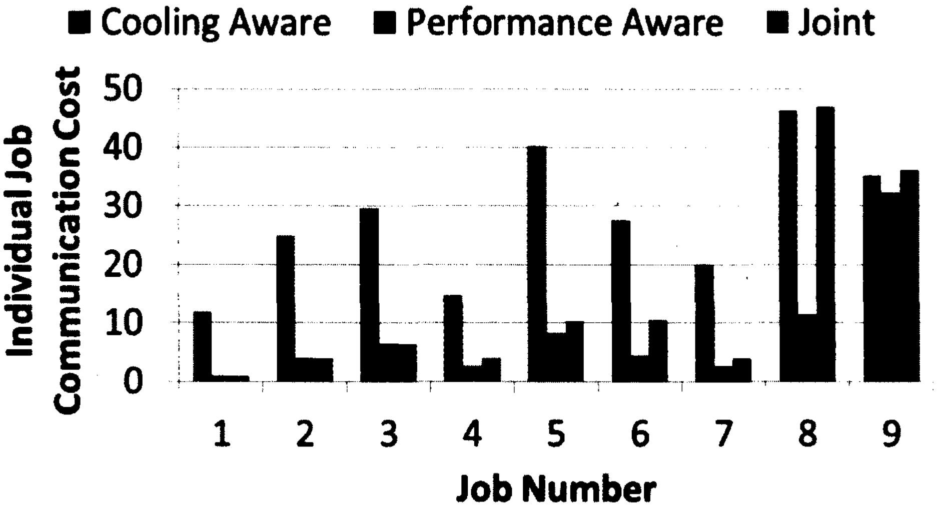


Figure 6·13: Individual job communication costs for multiple-row allocation.

temperature. As a result, communication cost is significantly affected by the distance between the communicating nodes. Joint policy resolves this issue by sacrificing some cooling efficiency. It assigns the job within a row in the most cooling-efficient way possible, and alternates the rows as more jobs arrive. However, if the number of available nodes in a row is not sufficient to service an incoming job, joint allocation also results in high communication cost. An example is seen for jobs 8 and 9 in Figure 6·13, where the jobs are allocated across the two rows.

We observe that our joint policy reduces the average cooling power by 30.8% compared to the performance-aware policy while increasing the power by only 0.5% compared to the cooling-aware policy. On the other hand, in comparison to the cooling-aware policy resulting in 2.45times larger average communication cost compared to the performance-aware policy, our joint policy causes only 0.69times larger cost. This is expected as our joint policy sacrifices some performance for improving cooling efficiency, and vice versa. Note that our results for the single and multiple-row allocation do not consider the change in application execution time as the communication cost changes. In other words, larger communication costs may change the power-performance characteristics of jobs, hence, also affect the cooling power. Next, we investigate such interactions between performance and cooling power in detail.

Dynamic Job Allocation

We investigate a dynamically changing workload scenario and compare our joint policy with the baseline policies. We generate a job queue with arrival time following an exponential distribution, which has been widely used in data center workload models (Hacker and Mahadik, 2011). We use an arrival rate of 15jobs/hour. In this experiment, we update the data center status as some of the jobs finish executing. We adjust the power and runtime of the jobs according to the communication latency to have a realistic model. The allocation is based on a first-come-first-serve policy.

When there are no available nodes, we wait for other jobs to finish. We simulate a total time of 4hours and use the last 3hours of the simulation in which 41jobs arrive. We record the maximum inlet temperature at each time step and cooling cost for each

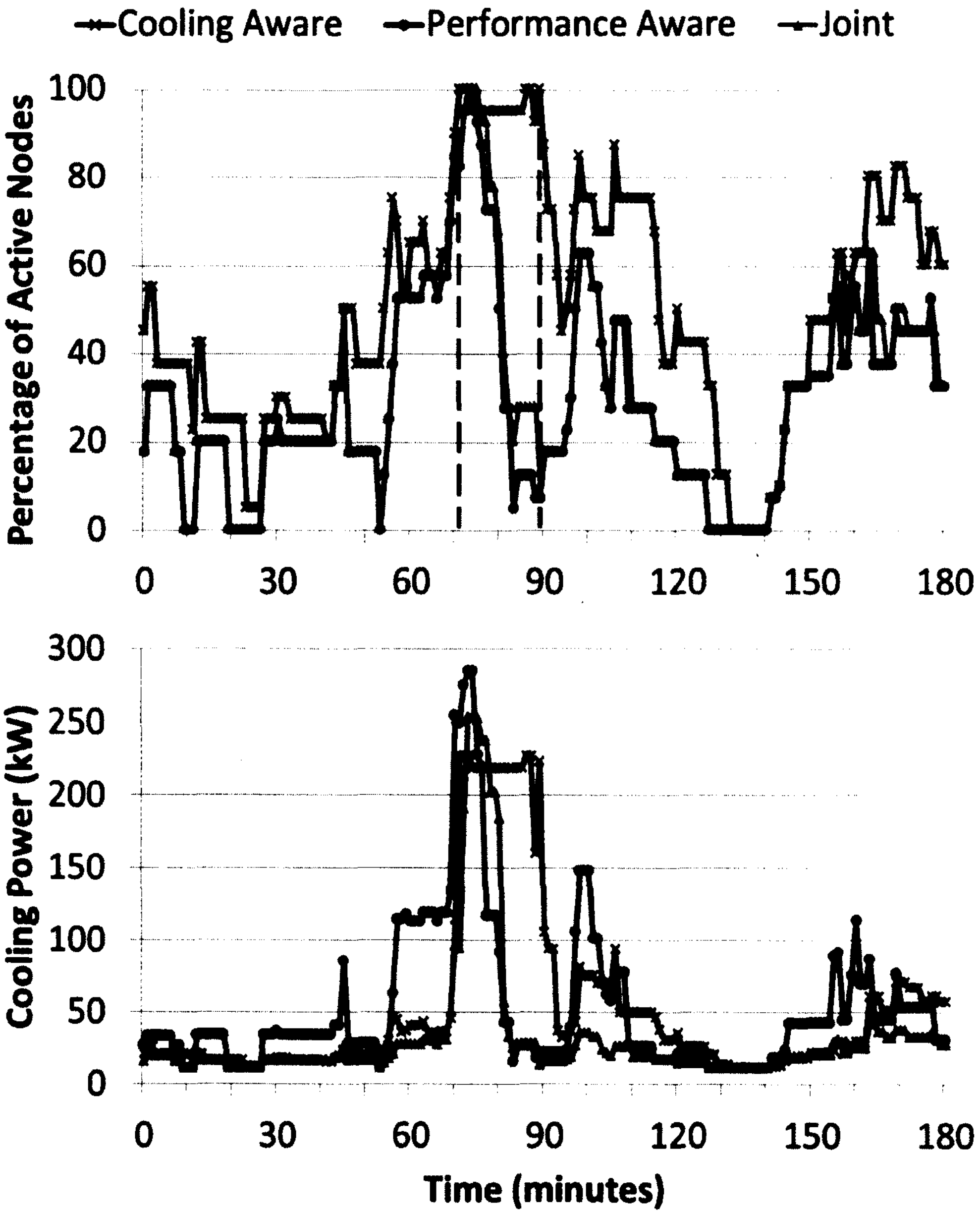


Figure 6·14: Percentage of the active nodes and the cooling power traces for dynamic allocation.

job. At each time step, the current available node list, power values of active nodes and the finishing time of the jobs are updated according to the model in Section 5.2.2. We set the communication level for all the applications, C%, as 20%.

The percentage of active nodes over time and the cooling power for all three allocation policies are illustrated in Figure 6·14. An important observation is that, in the dynamic case, the active node percentage is higher for the cooling-aware policy. This is because cooling-aware allocation results in high communication latency, which means that C% part of the application is running slower and thus results in longer runtime. Therefore, not only the nodes dissipate power for longer time, but also the next job is allocated in a less efficient way due to more limited allocation freedom.

On the other hand, joint optimization policy manages to overcome this problem by following a pattern similar to the MC1X1 algorithm. For example, during the time between the black dashed lines (70-90minutes), cooling-aware case has almost 100% of its nodes active, while for performance-aware and joint allocation cases, a job finishes after 75minutes and some nodes are freed. This performance effect translates into changes in the cooling cost, as shown in the bottom plot in Figure 6·14. Cooling power for our joint policy closely follows the cooling-aware policy from time 0 to 80minutes. However, when cooling-aware policy starts losing its efficiency because of the performance overheads, joint policy starts following the performance-aware policy (see Figure 6·14). These results show that for a data center running HPC applications with intensive communication, even a cooling-aware policy may result in inefficient cooling if it does not take into account the communication latency.

The average cooling power for the 3-hour period is 53.1kW for the cooling-aware policy while it is 53.3kW and 32.2kW for performance-aware and joint policies, respectively. This corresponds to close to 40% cooling power savings in comparison to both cooling-aware and performance-aware policies. We also evaluate the energy con-

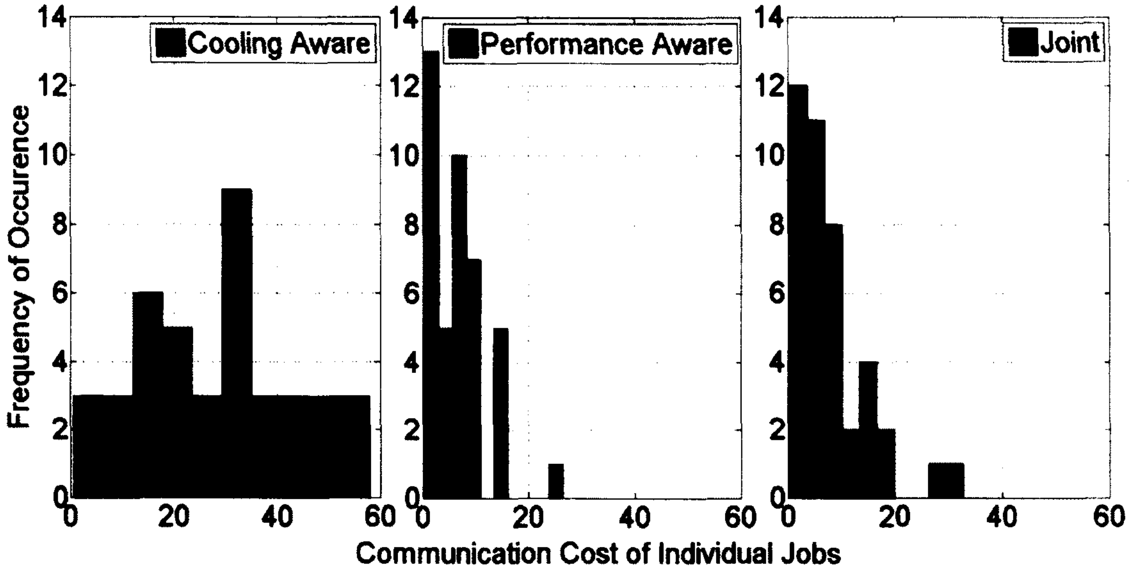


Figure 6·15: Histogram of the communication cost for the dynamic allocation experiment.

sumption of the data center for different allocation schemes and observe 170.7kWh, 163.3kWh, 98.4kWh for cooling-aware, performance-aware and joint allocation policies, respectively.

The comparison of the communication costs for the performance-aware, cooling-aware, and joint job allocation policy is presented in Figure 6·15. It shows that the frequency of the occurrence of communication costs for the total number of jobs allocated. For the performance-aware policy, data points are confined to the lower communication cost area, while for cooling-aware policy it is distributed across the spectrum. For the performance-aware policy, all the jobs have communication costs lower than 30, while 97.6% of the jobs have $CC < 30$ for the joint policy.

We conduct the same experiments with a higher communication level per application of $C = 30\%$. We observe the average cooling power as 74.2kW, 50.8kW and 32.1kW, while the corresponding energy consumptions are 238.96kWh, 154.96kWh, 98.3kWh for cooling-aware, performance-aware and joint allocation schemes, respectively. This corresponds to 56.7% cooling power saving compared to the cooling-aware policy and 36.8% compared to the performance-aware policy.

In order to include reliability awareness during job allocation, we set a minimum MTTF constraint of4 years and achieve an average cooling power of 20kW without total runtime change. Even though the allocation stalls in order to meet the MTTF constraint (i.e., waits for other jobs to finish so that temperatures decrease), total runtime of the job set is not affected under the given job arrival rate. When we increase the arrival rate to 25jobs/hour and compare the results with and without reliability constraint, we observe a 63% increase in the total runtime. Note that our runtime job allocation policy has low overhead. We measure the time spent on running the allocation algorithm for each job for the dynamic queue of 41jobs and observe that the time each job allocation decision takes is less than 1second in our Matlab-based implementation.

6.4 Summary

Performance, cooling cost and reliability have become serious concerns of many-core systems in HPC data centers as high performance computing moves towards exascale. In addition to causing reliability degradation, high temperatures increase the required cooling energy. Communication cost, on the other hand, has a significant impact on system performance in HPC data centers.

In this chapter, we propose a topology-aware workload allocation policy that maximizes system reliability by selecting between workload clustering and balancing approaches. Our policy improves the system reliability by up to 123.3% compared to existing temperature balancing policies. We also introduce a job allocation methodology to jointly optimize the communication cost and cooling energy in a data center while considering reliability constraints. Experimental results demonstrate that cooling-aware policies alone do not minimize overall energy if the job allocation results in large communication overheads. Our joint optimization policy minimizes cooling cost along with the communication time, providing better performance-energy tradeoffs in HPC data centers. Experimental results demonstrate that our joint optimization policy reduces the cooling cost by 40% compared to cooling-aware and performance-aware policies, while achieving comparable performance to performance-aware policy.

Chapter 7

Conclusion and Future Research Directions

7.1 Conclusion

Many-core systems, ranging from small-scale processors to large-scale high performance computing (HPC) data centers, have become the main trend for computing system design. The energy-efficient and reliable design of many-core high performance computing systems has been an active research area in the last decade. In comparison to single-core systems, many-core systems provide higher energy efficiency owing to their potential to deliver higher throughput per watt. However, power densities and temperatures increase following the performance improvement and bring major challenges in power delivery, cooling costs, and reliability. This thesis has addressed the energy and reliability challenges in both single-chip 3D many-core processors and many-core systems in HPC data centers.

7.1.1 A Simulation Framework and Runtime Optimization for Boosting Energy Efficiency in 3D Many-core Processors

In this thesis, we have presented our research on the modeling and runtime management for 3D many-core processors. Conventional 2D many-core systems have not been able to reach their peak performance capacity due to the memory latency and bandwidth restrictions. 3D many-core systems with on-chip stacked memory have the potential to dramatically improve performance owing to lower memory access latency and higher bandwidth, thus have the ability to significantly boost system energy efficiency. However, the performance increase may cause 3D many-core systems to exceed the power budgets, create thermal hot spots, increase cooling costs, and degrade reliability. This thesis contributes to addressing these challenges from two aspects: modeling and management.

A comprehensive modeling framework of 3D many-core systems is essential to provide efficient management policies and accurate analysis. We have introduced a methodology for constructing a simulation framework to address the complex interplay between performance, energy, and temperature in 3D systems. Our work is the first to jointly analyze performance, power, and thermal characteristics for both DRAM and processor layers. We have then utilized this simulation framework to design and evaluate runtime optimization and management policies for achieving high performance under power and temperature constraints.

We have proposed several management and optimization policies for improving the energy efficiency and reliability of 3D many-core systems with on-chip DRAM. Leveraging the detailed modeling and analysis of on-chip DRAM layers, we have introduced a memory management policy that targets applications with spatial variations in DRAM accesses and performs temperature-aware mapping of memory accesses to DRAM banks. In order to further exploit the performance potential of 3D systems while maintaining the peak power and temperature constraints, we have proposed a runtime optimization policy that dynamically monitors workload behavior and selects among low-power and turbo operating modes accordingly.

We have demonstrated that our policies provide up to 88.5% reduction in energy delay product (EDP) for a 16-core 3D system with stacked DRAM compared to equivalent 2D systems, while also delivering an average performance improvement of 36.1% in comparison to a statically optimized 3D system. Our runtime optimization policy also achieves an EDP reduction of up to 61.9% compared to a 3D system managed by a temperature-triggered DVFS policy.

7.1.2 Optimizing the Reliability, Performance, and Cooling Cost of Many-core Systems in HPC Data Centers via Workload Allocation

Performance, cooling energy, and reliability are also critical aspects in HPC data centers. In comparison to single-chip processors, high temperatures increase the required cooling energy in data centers and cause system-level reliability degradation. Also, communication cost of parallel applications has a significant impact on system performance in HPC data centers. In this thesis, we have addressed the energy and reliability challenges of many-core systems in HPC data centers from both modeling and management aspects.

Motivated by the analysis results of the reliability of a real-life multi-chip many-core system using a detailed reliability modeling approach, we have proposed a topology-aware workload allocation policy to dynamically optimize the reliability of multi-chip many-core systems in HPC data centers. We have evaluated our policy with simulations of real-world scenarios and demonstrated that our policy improves the reliability of multi-chip systems by up to 123.3% compared to thermal balancing. We have also studied the scalability of the policy. For a system with 16 cores, our policy improves system reliability by up to 101.7% compared to existing thermal balancing policies.

In order to jointly address the cooling energy and communication cost challenges in data centers, we have proposed a joint job allocation policy to optimize both cooling power and communication latency in HPC data centers. Our policy first uses the cooling-aware optimization algorithm to find the most cooling-efficient nodes to allocate a job and then applies the modified MC1X1 algorithm to allocate the job on cooling-efficient nodes while keeping the average L1 distance at a minimum. We have showed that for static allocation, our joint policy reduces the average cooling power by 30.8% compared to the performance-aware policy while it increases the power by only 0.5% compared to the cooling-aware policy. We have demonstrated that for dynamically changing workloads, solely using a cooling-aware policy does not give the minimum cooling power due to the resulting high communication latency. We have validated our joint policy under dynamically changing workloads and observed that, for HPC applications with a communication-to-computation ratio of 20%, our policy decreases the cooling power by 40% in comparison to cooling-aware and performance-aware policies.

7.2 Future Research Directions

7.2.1 3D Stacked Systems

Many open research problems exist in the design and management of 3D stacked systems, such as identifying killer applications for 3D processor, cost-aware 3D IC design, advanced techniques for 3D manufacturing, and modeling and validation for 3D system with liquid cooling.

One future direction in our research on 3D many-core processors is to explore the flexible heterogeneity of 3D stacked processors with cache resource pooling. 3D stacked processors, owing to the short communication latency achieved by vertically stacking and connecting poolable resources using TSVs, enable efficient resource pooling among different layers.

In many-core processors, resource pooling allows the share and management of architectural components among different cores. With well designed management policies, resource pooling has the potential to exploit the *flexible heterogeneity* in a many-core processor to the maximum extent. In the conventional 2D processors, however, the efficiency of resource pooling is limited by the large latency of accessing remote shared resources in the horizontal direction. Such limitation causes resource pooling in 2D not scalable to a large number of cores.

Most of the prior work on many-core 3D processors exploits the performance or energy efficiency benefits of 3D processors by considering fixed, homogeneous computational and memory resources (Black et al., 2006; Loh, 2008; Coskun et al., 2009a). However, the fixed resources are not able to satisfy applications with varying resource requirements, such as different memory uses. The *flexible heterogeneity* provided by resource pooling can address this challenge by including cores with different architectural resources in a single chip (Ipek et al., 2007; Ponomarev et al., 2006), and thus brings substantial benefits in reducing the energy consumption and cost in 3D stacked many-core processors.

A recent technique proposes pooling performance-critical microarchitectural resources such as register files in a 3D processor (Homayoun et al., 2012). Their work, however, does not address the memory requirements of applications. Considering the significance of the memory requirement in determining application performance in 3D many-core processors, we believe that the pooling of memory resources can provide additional heterogeneity of resources among the cores in a low-cost way and bring substantial energy efficiency improvements.

Cache resource pooling has the potential to further improve system energy efficiency due to the fact that different workloads require different amounts of cache resources to achieve their highest performance. Figure 7·1 shows the instructions per cycle (IPC) of the SPEC benchmarks when running on systems with various L2 cache sizes (from 512KB to 2MB). Among all the workloads, *soplex* has the largest throughput improvement at larger L2 cache sizes. We call such benchmarks *cache-hungry* workloads. On the other hand, benchmarks such as *libquantum* barely have any performance improvement at larger L2 cache size. This observation motivates

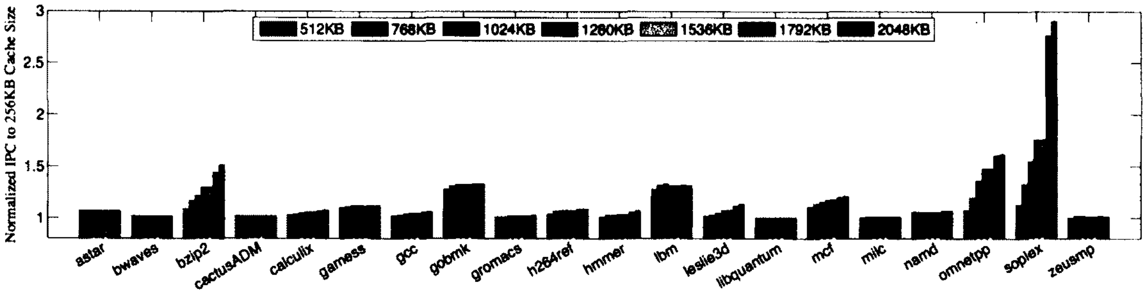


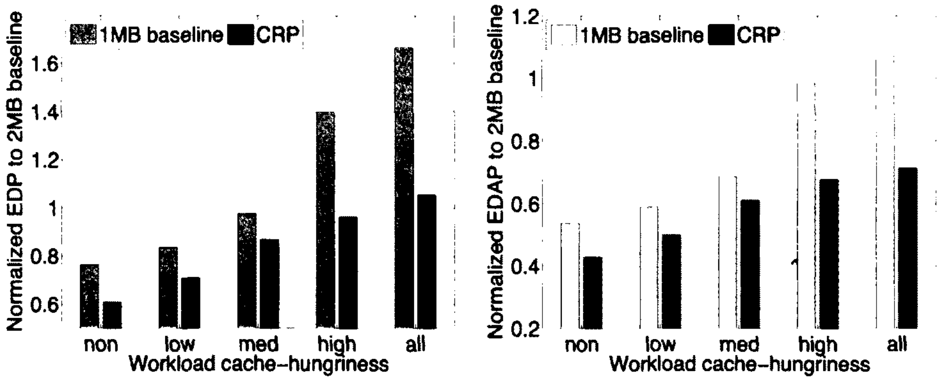
Figure 7·1: IPC of SPEC benchmarks for increasing L2 cache size. The IPC values are normalized with respect to using a 256KB L2 cache.

us to pool the cache resources in the adjacent layers in 3D stacked processors. By allocating the cache-hungry jobs in adjacent layers in the 3D processor with less cache-hungry jobs, we allow them to share a pool of cache resources thus provide the ability to improve the system energy efficiency.

As the first step to exploit resource pooling in 3D many-core processors, we implement the cache resource pooling on a four-layer 3D system, which has one core on each layer with a private L2 cache. The core on each layer is able to share the cache resources on its adjacent layers.

The preliminary results that compare the energy-delay product (EDP) and energy-delay-area product (EDAP) of the 3D systems with and without cache resource pooling are shown in Figure 7·2. We use two baseline 3D systems with 1MB and 2MB static cache resources, respectively, to compare their energy efficiency with the 3D system with cache resource pooling.

Figure 7·2 (a) presents the energy efficiency benefits of the 3D cache resource pooling for the 4-core system. We see that for all the workloads, 3D cache resource pooling provides lower EDP in comparison to the 1MB baseline. For all-cache-hungry workload, 2MB baseline provides the best EDP because of the larger cache size. Our results show that 3D cache resource pooling reduces EDP by up to 36.9% and 39.2% compared to 1MB and 2MB baselines, respectively.



(a) Normalized EDP

(b) Normalized EDAP

Figure 7·2: EDP and EDAP of the 3D system with cache resource pooling and its 3D baseline with 1MB static caches, normalized to its 2MB baseline.

Due to the fact that the die costs are proportional to the fourth power of the area (Rabaey et al., 2003), we consider area as a very important metric for evaluating the 3D systems. We use EDAP as a metric to evaluate the energy area efficiency (Li et al., 2009). As shown in Figure 7·2 (b), 3D cache resource pooling outperforms both baseline systems for all workload sets, reducing EDAP by up to 57.2% compared to the 2MB baseline.

From the preliminary results, we can see that 3D stacked processors with cache resource pooling have the potential to provide us higher energy efficiency by exploiting the *flexible heterogeneity* on the vertical dimension. In our future research, we will further explore such *flexible heterogeneity* on 3D many-core systems by providing more advanced management policies.

7.2.2 HPC Data Centers

HPC data centers face new challenges in performance, energy, reliability, and scalability. The interplays among these challenges are quite complex. Performance increase results in high temperature and high processing power; as a result, the scalability of data centers is limited by their power and cooling capacity. It is possible to reduce the cooling energy by allowing the data center temperatures to rise; however, the reliability constraints for computer components impose thermal thresholds as failure rates are exponentially dependent on the processor temperatures. How to concurrently analyze and jointly optimize the performance, energy, and reliability of HPC data centers is still an open problem.

In order to address these challenges, our future research directions on HPC data centers include developing simulation framework to provide design guidelines for HPC data centers, formulating and solving the joint optimization problem to reduce the communication cost of HPC applications and the cooling energy in a data center, and leveraging the communication patterns of HPC applications to further improve the performance through task mapping.

Simulation Approaches

Addressing the challenges of HPC data centers requires design guidelines from simulation approaches. It is impractical to explore the vast design space of data centers without a detailed system-level simulation framework. Existing simulators mostly address the performance and energy of HPC data centers separately, or are not able to scale to large-scale systems. So far, there is no simulation approach that is able to conduct concurrent evaluation of performance, energy, and reliability for large-scale HPC data centers running distributed-memory applications.

The Structural Simulation Toolkit (SST) is developed by Sandia National Laboratories to evaluate the performance of large-scale parallel computer architectures. It allows us to configure data centers with different network topologies, estimate the performance of processing and network components, and evaluate the communication cost between different nodes of data centers. However, the current SST simulation framework does not model the power, energy, and reliability for HPC data centers. It is highly desirable to integrate the data center thermal, energy, and reliability models into the SST simulation framework.

In Chapter 5, we have discussed the power and cooling energy model for HPC data centers, and the reliability model for multi-chip many-core servers. In our future work, we plan to scale the reliability model to larger-scale data center level and also integrate the implementation of power, cooling energy, and reliability models into SST framework.

Formalization of Joint Optimization Problem

In Chapter 6, we have presented a heuristic algorithm which jointly optimizes the communication cost of HPC applications and the cooling energy in data centers. In order to provide the ability of optimizing the overall costs for users of data centers who have different preferences to performance or cooling energy saving, we need a formalization of the joint optimization job allocation problem with adjustable weighting factors to communication cost and cooling energy cost.

Taking the formalization of the joint optimization problem as one of our future work directions, we propose a formulation with this joint goal as shown in Equation (7.1):

$$\begin{aligned} \underset{X_{job}}{\text{minimize}} \quad & \alpha \cdot Cost_{comm}(X_{job}) + \beta \cdot Cost_{cool}(X_{job}) \\ \text{subject to} \quad & E \times X_{job} = n \end{aligned} \tag{7.1}$$

where $X_{job} = \{x_1, x_2, ..., x_N\}$ is a vector that represents the job allocation decision. N is the number of total nodes within the data center and x_i $(i = 1, \ldots, N)$ are the integer variables denoting whether a node is busy or idle. E is a $1 \times N$ vector with all elements set to 1, n is the total number of nodes required by a job. The optimization problem is subject to the linear constraint $E \times X_{job} = n$, which means the job requires n nodes in the data center.

and the cooling cost of the data center, respectively. α and β are the corresponding weighting factors for the communication cost and the cooling cost. α and β can be adjusted to adapt to optimization requirements in different data centers. A larger ratio of α/β indicates that reducing the communication cost is more significant compared to decreasing the cooling cost. For example, when $\alpha = 1$ and $\beta = 0$, the joint optimization problem is converted to the job allocation problem that only considers the communication cost. If $\alpha = 0$ and $\beta = 1$, the job allocation problem solely considers the cooling cost.

The cooling energy cost model of the data center is based on the linear thermal model as introduced in Chapter 5. The communication cost of each job arriving at the cluster can be expressed as in Equation (7.2):

$$Cost_{comm}(X_{job}) = \frac{X_{job} \cdot H \cdot X_{job}^T}{n} \tag{7.2}$$

where H is an $N \times N$ matrix, whose elements represent the communication delay between each pair of nodes within the data center. Thus, Equation (7.2) calculates the total communication cost among all the nodes that are assigned to the current job. The total cost is then normalized to the job size, n. The communication cost matrix H is determined by the data center's network topologies. By utilizing data center level simulation framework (e.g., SST), we are able to generate the H matrix for various data center network topologies.

By integrating the formulations of the communication cost and the cooling energy cost into the formulation of our joint optimization problem in Equation (7.1), we obtain the Equation (7.3). As the constants in the goal function do not affect the optimization decisions, we simplify the equation and only use the quadratic part of the communication cost function and the linear part of the cooling cost function while computing the total cost.

$$\begin{aligned} \underset{X_{job}}{\text{minimize}} \quad & \frac{\alpha}{n} \cdot X_{job} \cdot H \cdot X_{job}^{T} + \frac{\beta}{U} \cdot (D \cdot P_{util} \cdot X_{job}) \\ \text{subject to} \quad & E \times X_{job} = n \end{aligned} \tag{7.3}$$

As shown in Equation (7.3), we express the joint optimization problem as a binary quadratic programming (BQP) problem, which is a combinatorial optimization problem. BQP is an NP-hard problem; however, in practice, it can be efficiently solved using well-known discrete optimization techniques such as the branch and bound algorithm (Trinh et al., 2012). The joint optimization problem is solvable using the TOMLAB/CPLEX solver, which provides a Matlab interface to solve complex optimization problems, such as BQP problem.

Task Mapping

Performance is the first-order constraint in data center design and management. Today's HPC data centers run highly parallel applications, such as scientific and financial computing applications, which typically require a large set of computing nodes for achieving high performance. Most prior work on job allocation assumes *all-to-all* communication patterns for HPC applications. In order to further reduce the communication delay of HPC applications, we leverage the communication patterns of each application.

For this reason, one of our future research directions on HPC data centers is to improve the performance of data centers by optimizing task mapping with consideration of HPC applications with different communication patterns. Task mapping with consideration of application communication patterns becomes more important as the number of nodes in data centers grows significantly. Mapping the task onto the allocated nodes in a data center by utilizing the extracted communication patterns

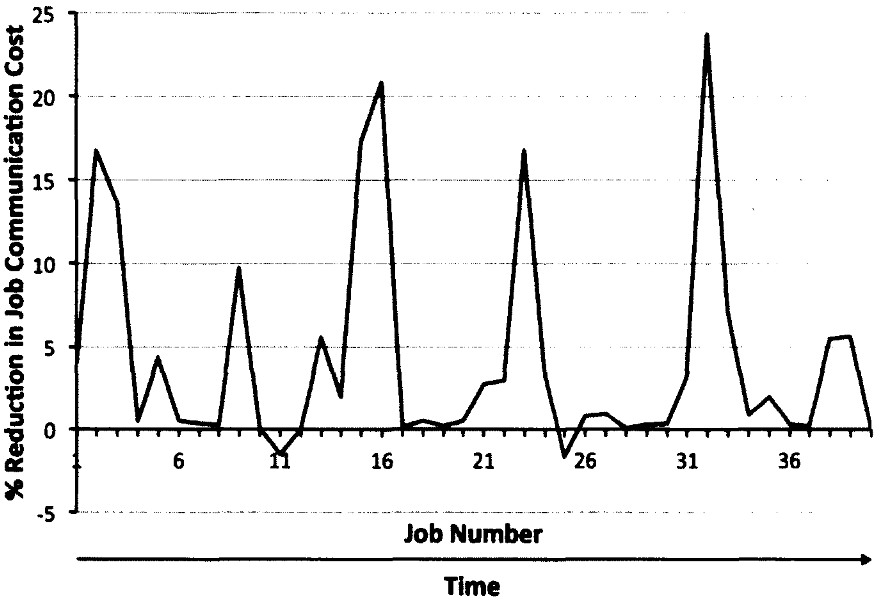


Figure 7·3: Percentage reduction in job communication cost using RCB-based task mapping for the dynamic allocation scenario.

from HPC applications brings us more flexibility in reducing communication cost.

We present the benefits of tasking mapping with considerations of communication pattern using the preliminary results shown in Figure 7·3, where we compare the communication cost resulting from the RCB task mapping algorithm (Hoefler and Snir, 2011) against the communication cost resulting from the in-order task mapping algorithm. In RCB algorithm, the logical communication pattern of an application is represented using a weighted graph and the physical data center nodes with a certain network topology is presented using a separate graph. RCB algorithm determines the task mapping by recursively splitting both graphs into equal halves using minimum weighted edge-cuts. In-order task mapping algorithm, which allocates the tasks of a job starting from the top left of the data center, traverses the assigned nodes from left to right and from top to bottom.

Figure 7·3 shows the reduction in the job communication cost using RCB-based task mapping in comparison to the baseline in-order policy for a dynamic job queue with 40 jobs. We observe that, on average across all the jobs, using RCB task mapping policy with consideration of communication patterns achieves 4.3% reduction in communication cost in comparison to using the baseline in-order policy.

These preliminary results demonstrate that tasking mapping with considerations of communication pattern could bring us considerable performance improvements for HPC applications running in data centers. In our future research, we will further explore the benefits of tasking mapping by integrating tasking mapping into our joint optimization algorithm to reduce communication cost and cooling energy cost for data centers at the same time.

References

Alam, M. A., Kufluoglu, H., Varghese, D., and Mahapatra, S. (2007). A comprehensive model for PMOS NBTI degradation: recent progress. *Microelectronics Reliability*, 47(6):853–862.

Atienza, D. (2010). Thermal-aware design for 3D multi-processor. *Flash informatique EPFL, Special Issue on High-Performance Computing*, (10):34–37.

Awasthi, M. et al. (2010). Handling the problems and opportunities posed by multiple on-chip memory controllers. In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 319–330.

Ayoub, R. Z. and Rosing, T. S. (2009). Predict and act: dynamic thermal management for multi-core processors. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 99–104.

Bailey, D. et al. (1994). The NAS parallel benchmarks. Technical Report RNR-94-007.

Belden (2007). Data center design guidelines. `http://www.belden.com/pdfs/techbull/datacenterguide.pdf`.

Bender, M. A., Bunde, D. P., Demaine, E. D., Fekete, S. P., Leung, V. J., Meijer, H., and Phillips, C. A. (2008). Communication-aware processor allocation for supercomputers: finding point sets of small average distance. *Algorithmica*, 50(2):279–298.

Benini, L., Bogliolo, A., and De Micheli, G. (2000). A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3):299–316.

Bhattacharya, S. and Tsai, W. (1994). Lookahead processor allocation in mesh-connected massively parallel multicomputer. In *International Parallel Processing Symposium*, pages 868–875.

Bienia, C. (2011). *Benchmarking Modern Multiprocessors.* PhD thesis, Princeton University.

Binkert, N. L., Dreslinski, R. G., Hsu, L. R., Lim, K. T., Saidi, A. G., and Reinhardt, S. K. (2006). The M5 simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60.

Biswas, S. et al. (2011). Fighting fire with fire: modeling the datacenter-scale effects of targeted superlattice thermal management. In *International symposium on Computer Architecture (ISCA)*, pages 331–340.

Black, B. et al. (2006). Die stacking (3D) microarchitecture. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 469–479.

Bose, P. et al. (2010). Power-efficient, reliable microprocessor architectures: modeling and design methods. In *Great lakes symposium on VLSI (GLSVLSI)*, pages 299–304.

Brightwell, R., Fisk, L. A., Greenberg, D. S., Hudson, T., Levenhagen, M., MacCabe, A. B., and Riesen, R. (2000). Massively parallel computing using commodity components. *Parallel Computing*, 26(2-3):243–266.

Brown, D. J. and Reams, C. (2010). Toward energy-efficient computing. *Communications of the ACM*, 53(3):50–58.

Brunschwiler, T., Paredes, S., Drechsler, U., Michel, B., Cesar, W., Toral, G., Temiz, Y., and Leblebici, Y. (2009). Validation of the porous-medium approach to model interlayer-cooled 3D-chip stacks. In *IEEE International Conference on 3D System Integration (3DIC)*, pages 1–10.

Cochran, R., Hankendi, C., Coskun, A. K., and Reda, S. (2011). Identifying the optimal energy-efcient operating points of parallel workloads. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 608–615.

Cong, J., Luo, G., Wei, J., and Zhang, Y. (2007). Thermal-aware 3D IC placement via transformation. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 780–785.

Coskun, A., Meng, J., Atienza, D., and Sabry, M. (2011). Attaining single-chip, high-performance computing through 3D systems with active cooling. *IEEE Micro*, 31(4):63–75.

Coskun, A. K., Atienza, D., Rosing, T. S., Brunschwiler, T., and Michel, B. (2010). Energy-efficient variable-flow liquid cooling in 3D stacked architectures. In *Conference on Design, Automation and Test in Europe (DATE)*, pages 111–116.

Coskun, A. K., Ayala, J. L., Atienza, D., Rosing, T. S., and Leblebici, Y. (2009a). Dynamic thermal management in 3D multicore architectures. In *Conference on Design, Automation and Test in Europe (DATE)*, pages 1410–1415.

Coskun, A. K., Rosing, T. S., and Gross, K. C. (2009b). Utilizing predictors for efficient thermal management in multiprocessor SoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1503–1516.

Coskun, A. K., Rosing, T. S., Mihic, K., Leblebici, Y., and Micheli, G. D. (2006). Analysis and optimization of mpsoc reliability. *Journal of Low Power Electronics (JOLPE)*, 2(1):56–69.

Coskun, A. K., Rosing, T. S., Whisnant, K. A., and Gross, K. C. (2008). Static and dynamic temperature-aware scheduling for multiprocessor socs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(9):1127–1140.

Coskun, A. K., Strong, R., Tullsen, D. M., and Rosing, T. S. (2009c). Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors. In *International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 169–180.

Crovella, M., Bianchini, R., Leblanc, T., Markatos, E., and Wisniewski, R. (1992). Using communication-to-computation ratio in parallel program design and performance prediction. In *IEEE Symposium on Parallel and Distributed Processing (IPDPS)*, pages 238–245.

Dhiman, G., Marchetti, G., and Rosing, T. S. (2010). vGreen: A system for energy-efficient management of virtual machines. *ACM Transactions on Design Automation of Electronic Systems*, 16(1):1–27.

Donald, J. and Martonosi, M. (2006). Techniques for multicore thermal management: Classification and new exploration. In *International Symposium on Computer Architecture (ISCA)*, pages 78–88.

Dong, X., Zhao, J., and Xie, Y. (2010). Fabrication cost analysis and cost-aware design space exploration for 3-D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(12):1959–1972.

Ferreira, K. et al. (2011). Evaluating the viability of process replication reliability for exascale systems. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12.

Ferri, C., Reda, S., and Bahar, R. I. (2008). Parametric yield management for 3D ICs. *ACM Journal on Emerging Technologies in Computing Systems*, 4(19):1023–1030.

Ghosh, M. and Lee, H. S. (2007). Smart refresh: An enhanced memory controller design for reducing energy in conventional and 3D die-stacked DRAMs. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 134–145.

Golshani, N., Derakhshandeh, J., Ishihara, R., Beenakker, C., Robertson, M., and Morrison, T. (2010). Monolithic 3D integration of SRAM and image sensor using two layers of single grain silicon. In *IEEE International 3D Systems Integration Conference (3DIC)*, pages 1–4.

Hacker, T. J. and Mahadik, K. (2011). Flexible resource allocation for reliable virtual cluster computing systems. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12.

Hanson, H. et al. (2007). Thermal response to dvfs: Analysis with an intel pentium m. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 219–224.

Hanumaiah, V. and Vrudhula, S. (2011). Reliability-aware thermal management for hard real-time applications on multi-core processors. In *Conference on Design, Automation and Test in Europe (DATE)*, pages 1–6.

Healy, M. et al. (2007). Multiobjective microarchitectural floor-planning for 2-D and 3-D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(1):38–52.

Heath, T. et al. (2006). Mercury and freon: temperature emulation and management for server systems. In *International conference on Architectural support for programming languages and operating systems (ASPLOS)*, pages 106–116.

Hoefler, T. and Snir, M. (2011). Generic topology mapping strategies for large-scale parallel architectures. In *Proceedings of the International Conference on Supercomputing*, pages 75–84.

Homayoun, H., Kontorinis, V., Shayan, A., Lin, T., and Tullsen, D. (2012). Dynamically heterogeneous cores through 3D resource pooling. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–12.

Howard, J. et al. (2010). A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 108–109.

Huang, L., Yuan, F., and Xu, Q. (2009). Lifetime reliability-aware task allocation and scheduling for MPSoC platforms. In *Conference on Design, Automation and Test in Europe (DATE)*, pages 51–56.

Hung, W. et al. (2006). Interconnect and thermal-aware floorplanning for 3D microprocessors. In *International Symposium on Quality Electronic Design (ISQED)*, pages 98–104.

Ipek, E. et al. (2008). Self-optimizing memory controllers: A reinforcement learning approach. In *International Symposium on Computer Architecture (ISCA)*, pages 39–50.

Ipek, E., Kirman, M., Kirman, N., and Martinez, J. F. (2007). Core fusion: accommodating software diversity in chip multiprocessors. In *International symposium on Computer Architecture (ISCA)*, pages 186–197.

Isci, C., Contreras, G., and Martonosi, M. (2006a). Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 359–370.

Isci, C. et al. (2006b). An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 347 –358.

JEDEC (2006). Failure mechanisms and models for semiconductor devices, Technical report. `http://www.jedec.org`.

Jin, Y., Yum, K. H., and Kim, E. J. (2008). Adaptive data compression for high-performance low-power on-chip networks. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 354–363.

Kang, K., Kim, J., Yoo, S., and Kyung, C. (2010). Temperature-aware integrated DVFS and power gating for executing tasks with runtime distribution. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(9):1381–1394.

Khan, N. H., Alam, S. M., and Hassoun, S. (2011). Power delivery design for 3D ICs using different through-silicon via (TSV) technologies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(4):647–658.

Kim, J., Ruggiero, M., and Atienza, D. (2012). Free cooling-aware dynamic power management for green datacenters. In *International Conference on High Performance Computing and Simulation (HPCS)*, pages 140 –146.

Kongetira, P., Aingaran, K., and Olukotun, K. (2005). Niagara: a 32-way multithreaded sparc processor. *IEEE Micro*, 25(2):21–29.

Koomey, J. G. (2008). Toward energy-efficient computing. *Environmental Research Letters*, 3(034008).

Koyanagi, M., Kurino, H., Lee, K. W., Sakuma, K., Miyakawa, N., and Itani, H. (1998). Future system-on-silicon LSI chips. *IEEE Micro*, 18(4):17–22.

Kumar, A. et al. (2006). HybDTM: a coordinated hardware-software approach for dynamic thermal management. In *ACM/IEEE Design Automation Conference (DAC)*, pages 548–553.

Kumar, R., Farkas, K. I., Jouppi, N. P., Ranganathan, P., and Tullsen, D. M. (2003). Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 81–92.

Kumar, S., Sabharwal, Y., Garg, R., and Heidelberger, P. (2008). Optimization of all-to-all communication on the Blue Gene/L supercomputer. In *International Conference on Parallel Processing*, pages 320–329.

Leung, V., Arkin, E., Bender, M., Bunde, D., Johnston, J., Lal, A., Mitchell, J., Phillips, C., and Seiden, S. (2002). Processor allocation on Cplant: achieving general processor locality using one-dimensional allocation strategies. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 296–304.

Li, S., Ahn, J. H., Strong, R. D., Brockman, J. B., Tullsen, D. M., and Jouppi, N. P. (2009). McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 469–480.

Li, T. and John, L. K. (2003). Run-time modeling and estimation of operating system power consumption. In *International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 160–171.

Liu, C. C., Ganusov, I., Burtscher, M., and Tiwari, S. (2005). Bridging the processor-memory performance gap with 3D IC technology. *IEEE Design Test of Computers*, 22(6):556–564.

Liu, S., Leung, B., Neckar, A., Memik, S. O., Memik, G., and Hardavellas, N. (2011). Hardware/software techniques for DRAM thermal management. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 515–525.

Lively, C. et al. (2011). Energy and performance characteristics of different parallel implementations of scientific applications on multicore systems. *International Journal of High Performance Computing Applications*, 25(3):342–350.

Loh, G. H. (2008). 3D-stacked memory architectures for multi-core processors. In *International Symposium on Computer Architecture (ISCA)*, pages 453–464.

Loi, G. L., Agrawal, B., Srivastava, N., Lin, S.-C., Sherwood, T., and Banerjee, K. (2006). A thermally-aware performance analysis of vertically integrated (3-D) processor-memory hierarchy. In *ACM/IEEE Design Automation Conference (DAC)*, pages 991–996.

Mache, J., Lo, V., and Windisch, K. (1997). Minimizing message-passing contention in fragmentation-free processor allocation. In *International Conference on Parallel and Distributed Computing Systems (ICPADS)*, pages 120–124.

Meng, J., Chen, C., Coskun, A. K., and Joshi, A. (2011). Run-time energy management of manycore systems through reconfigurable interconnects. In *ACM/IEEE Great Lakes Symposium on VLSI (GLSVLSI)*, pages 43–48.

Moore, J., Chase, J., Ranganathan, P., and Sharma, R. (2005). Making scheduling "cool": temperature-aware workload placement in data centers. In *Proceedings of the USENIX Annual Technical Conference*, pages 5–15.

Mulas, F. et al. (2008). Thermal balancing policy for streaming computing on multiprocessor architectures. In *Conference on Design, Automation and Test in Europe (DATE)*, pages 734–739.

Pakbaznia, E. and Pedram, M. (2009). Minimizing data center cooling and server power costs. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 145–150.

Ponomarev, D., Kucuk, G., and Ghose, K. (2006). Dynamic resizing of superscalar datapath components for energy efficiency. *IEEE Transactions on Computers*, 55(2):199–213.

Puttaswamy, K. and Loh, G. H. (2007). Thermal herding: microarchitecture techniques for controlling hotspots in high-performance 3D-integrated processors. In *IEEE International Symposium on High-Performance Computer Architecture*, pages 193–204.

Rabaey, J., Chandrakasan, A., and Nikolic., B. (2003). *Digital Integrated Circuits: A Design Perspective, 2nd edition.*

Rad, P., Karki, K., and Webb, T. (2008). High-efficiency cooling through computational fluid dynamics. *Dell Power Solutions.*

Rajic, J. (2009). Evolving toward the green data center. `http://http://stack.nil.si`.

Sabry, M., Coskun, A., and Atienza, D. (2010). Fuzzy control for enforcing energy efficiency in high-performance 3D systems. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 642–648.

Sansottera, A. and Cremonesi, P. (2011). Cooling-aware workload placement with performance constraints. *Performance Evaluation*, 68(11):1232–1246.

Skadron, K., Stan, M. R., Huang, W., Velusamy, S., Sankaranarayanan, K., and Tarjan, D. (2003). Temperature-aware microarchitecture. In *International Symposium on Computer Architecture (ISCA)*, pages 2–13.

Srinivasan, J., Adve, S. V., Bose, P., and Rivers, J. A. (2004a). The case for lifetime reliability-aware microprocessors. In *International Symposium on Computer Architecture (ISCA)*, pages 276–287.

Srinivasan, J., Adve, S. V., Bose, P., and Rivers, J. A. (2004b). The impact of technology scaling on lifetime reliability. In *International Conference on Dependable Systems and Networks (DSN)*, pages 177–186.

Srinivasan, J. et al. (2003). Ramp: A model for reliability aware microprocessor design. Technical Report IBM-RC23048 (W0312-122).

Srinivasan, J. et al. (2005). Exploiting structural duplication for lifetime reliability enhancement. In *International symposium on Computer Architecture (ISCA)*, pages 520–531.

Stavros Harizopoulos, Mehul A. Shah, J. M. P. R. (2009). Energy efficiency: The new holy grail of data management systems research. In *Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 1–8.

Su, H., Liu, F., Devgan, A., Acar, E., and Nassif, S. (2003). Full chip leakage estimation considering power supply and temperature variations. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 78–83.

Sun, G., Dong, X., Xie, Y., Li, J., and Chen, Y. (2009). A novel architecture of the 3D stacked MRAM L2 cache for CMPs. In *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 239–249.

Tang, Q., Gupta, S. K. S., and Varsamopoulos, G. (2008). Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: a cyber-physical approach. *IEEE Transactions on Parallel and Distributed Systems*, 19(11):1458–1472.

Tang, Q., Mukherjee, T., Gupta, S., and Cayton, P. (2006). Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters. In *International Conference on Intelligent Sensing and Information Processing (ICISIP)*, pages 203–208.

Teng, Q., Sweeney, P. F., and Duesterwald, E. (2009). Understanding the cost of thread migration for multi-threaded java applications running on a multicore platform. In *IEEE International Symposium on Performance Analysis of Systems and Software*, pages 123–132.

Teodorescu, R. and Torrellas, J. (2008a). Variation-aware application scheduling and power management for chip multiprocessors. *ACM SIGARCH Computer Architecture News*, 36(3):363–374.

Teodorescu, R. and Torrellas, J. (2008b). Variation-aware application scheduling and power management for chip multiprocessors. In *International Symposium on Computer Architecture (ISCA)*, pages 363–374.

Thoziyoor, S., Muralimanohar, N., Ahn, J. H., and Jouppi, N. P. (2008). CACTI 5.1. Technical report, HP Laboratories, Palo Alto.

Topaloglu, R. (2011). Applications driving 3D integration and corresponding manufacturing challenges. In *ACM/IEEE Design Automation Conference (DAC)*, pages 220–223.

Trinh, H., Fan, Q., Gabbur, P., and Pankanti, S. (2012). Hand tracking by binary quadratic programming and its application to retail activity recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1902–1909.

U.S. Environmental Protection Agency (2007). EPA report to congress on server and data center energy efficiency. `http://www.energystar.gov`.

Walsh, E. et al. (2010). From chip to cooling tower data center modeling: Part II influence of chip temperature control philosophy. In *IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pages 1–7.

Wang, L. et al. (2009). Towards thermal aware workload scheduling in a data center. In *I-SPAN*, pages 116–122.

Wang, S. and Chen, J.-J. (2010). Thermal-aware lifetime reliability in multicore systems. In *International Symposium on Quality Electronic Design (ISQED)*, pages 399–405.

Winter, J. and Albonesi, D. (2008). Scheduling algorithms for unpredictably heterogeneous cmp architectures. In *International Conference on Dependable Systems and Networks (DSN) with FTCS and DCC*, pages 42–51.

Wu, X. et al. (2010). Cost-driven 3D integration with interconnect layers. In *ACM/IEEE Design Automation Conference (DAC)*, pages 150–155.

Xiang, Y. et al. (2010). System-level reliability modeling for MPSoCs. In *International Conference on Hardware/Software Codesign and System Synthesis*, pages 297–306.

Zhou, X. et al. (2008). Thermal management for 3D processors via task scheduling. In *International Conference on Parallel Processing (ICPP)*, pages 115–122.

Zhu, C. et al. (2008). Three-dimensional chip-multiprocessor run-time thermal management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(8):1479–1492.

CURRICULUM VITAE

Jie Meng

Education

Ph.D., Boston University, 09/2013
Electrical and Computer Engineering Department
Advisor: Professor Ayse K. Coskun
Dissertation Title: "Modeling and Optimization of High-Performance Many-core Systems for Energy-Efficient and Reliable Computing"
GPA: 3.96/4.00

M.A.S., McMaster University, 07/2008
Electrical and Computer Engineering Department
Advisor: Professor John W. Bandler
Thesis Title: "Microwave Circuit Optimization Exploiting Tuning Space Mapping"
GPA: 4.00/4.00

B.S., University of Science and Technology of China, 07/2004
Electrical Engineering and Information Science Department
GPA: 3.72/4.00

Professional Experience

Intel Corporation, 04/2012 to 08/2012
Graduate Intern, *Supervisor*: Xin (Max) Ma
Conducted design and validation for on-chip power regulator and thermal sensors on Intel server microprocessors.

Sandia National Laboratories, 05/2011 to 08/2011
Research Intern, *Supervisor*: Dr. Arun Rodrigues
Developed performance, power, and reliability simulation tools and management techniques for high-performance computing systems.

Semiconductor Manufacturing International Corporation (SMIC)

Software Engineer & System Administrator, 07/2004 to 08/2006
Worked on designing and maintaining Unix-based computer-integrated manufacturing (CIM) systems for semiconductor device fabrication.

Research Experience

Performance and Energy-Aware Computing Laboratory

Research Assistant at Boston University, 01/2010 to present
Conducted research on performance, energy, and temperature modeling and energy efficiency optimization of high-performance computing systems and 3D-stacked processors.

Networking and Information System Laboratory

Research Assistant at Boston University, 05/2009 to 12/2009
Implemented a TCP/IP based networking data synchronization agent interface for network communication.

Simulation Optimization Systems Research Laboratory

Research Assistant at McMaster University, 09/2006 to 07/2008
Developed the Tuning Space Mapping optimization method for improving the accuracy and computational efficiency of RF/Microwave circuit design.

Honors & Awards

- Best Paper on High Performance Extreme Computing Conference, Sep. 2012.
- DAC A. Richard Newton Graduate Scholarship Award, June 2011.
- Google scholarship for Google 2010 GRAD CS Forum, January 2010.
- Outstanding Graduate Teaching Fellow, BU, School of Eng., June 2009.
- Best Graduate Teaching Fellow of the Year, BU, ECE Dept., May 2009.
- Graduate Research Fellowship of McMaster University, 2006 to 2008.
- Best Project Annual Award for CIM, SMIC, 2005.

Refereed Journal Publications

1. M. Sabry, A. Sridhar, **J. Meng**, A. Coskun, D. Atienza. "GreenCool: an Energy-efficient Liquid Cooling Design Technique for 3D MPSoCs via Channel Width Modulation". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.32 no.4, pp.524-537, 2013.

2. A. Coskun, **J. Meng**, D. Atienza, M. Sabry. "Attaining Single-Chip, High-Performance Computing through 3D Systems with Active Cooling". In *IEEE Micro, Special Issue on Big chips*, pp.63-73, August 2011.

3. S. Koziel, **J. Meng**, J. Bandler, M. Bakr, and Q. Cheng. "Accelerated Microwave Design Optimization with Tuning Space Mapping Method". In *IEEE Trans. on Microw.Theory Techn.*, vol.57 no.2, pp.383-393, 2009.

Refereed Conference Publications

1. **J. Meng**, Tiansheng Zhang, A. Coskun. "Dynamic Cache Pooling for Improving Energy Efficiency in 3D Stacked Multicore Processors". To appear in *Proceedings of IEEE Conf. on VLSI and System-on-Chip (VLSI-SoC)* , October 2013.

2. F. Kaplan, **J. Meng**, A. Coskun. "Optimizing Communication and Cooling Costs in HPC Data Centers via Intelligent Job Allocation". To appear in *Proceedings of IEEE Intl. Green Comp. Conf. (IGCC)*, June 2013.

3. **J. Meng**, F. Kaplan, M. Hsieh, A. Coskun. "Topology-aware Reliability Optimization for Multiprocessor Systems". In *Proceedings of IEEE Conf. on VLSI and System-on-Chip (VLSI-SoC)*, pp.243-248, October 2012.

4. **J. Meng**, K. Kawakami, A. Coskun. "Optimizing Energy Efficiency of 3D Multicore Systems with Stacked DRAM under Power and Thermal Constraints". In *Proceedings of Design Automation Conference (DAC)*, pp.648-655, June 20.25.

5. Mingyu Hsieh, **Jie Meng**, Michael Levenhagen, Kevin Pedretti, Ayse K. Coskun. "SST + gem5 = A Scalable Simulation Infrastructure for High Performance Computing". In *Proceedings of International ICST Conference on Simulation Tools and Techniques*, pp.648-655, May 2012.

6. **J. Meng**, A. Coskun. "Analysis and Runtime Management of 3D Systems with Stacked DRAM for Boosting Energy Efficiency". In *Proceedings of Design Automation and Test in Europe (DATE)*, pp.611-616, May 2012.

7. **J. Meng**, D. Rossell, A. Coskun. "3D Systems with On-Chip DRAM for Enabling Low-Power High-Performance Computing". In *IEEE High Performance Extreme Computing Conference (HPEC)*, September 2011.
8. **J. Meng**, D. Rossell, A. Coskun. "Exploring Performance, Power, and Temperature Characteristics of 3D Systems with On-Chip DRAM". In *Proceedings of IEEE Intl. Green Comp. Conf. (IGCC)*, pp.1-6, July 2011.
9. Chao Chen, **Jie Meng**, Ayse K. Coskun, Ajay Joshi. "Express Virtual Channels with Taps (EVC-T): A Flow Control Technique for Network-on-Chip (NoC) in Manycore Systems". In *Proceedings of Hot Interconnects (HOTI)*, pp.1-10, August 2011.
10. **J. Meng**, C. Chen, A. Coskun, A. Joshi. "Run-time Energy Management of Manycore Systems through Reconfigurable Interconnects". In *Proceedings of ACM Great Lakes Symposium on VLSI*, pp.43-48, May 2011.
11. **J. Meng**, S. Koziel, J. Bandler, M. Bakr, and Q. Cheng. "Tuning Space Mapping: a Novel Technique for Engineering Design Optimization". In *IEEE International Microwave Symposium Digest*, pp.991-996, June 2008.