Software-Configured Compute Environments

Ulrich Drepper <drepper@redhat.com>

November 2018
Driving Factors

- Complexity
- Cost
- Performance
- Standardized Components
- Abstraction
- Scaling
Limitations

Complexity

Performance

Bargain
More Realistic

Complexity

Performance

Bargain
"Automatic"

Complexity → Bargain → Dennard Scaling → Performance
“Automatic”
“Automatic”

Complexity

Bargain

Performance

Dennard Scaling
Abstraction Problem

```cpp
int cnt1;
int cnt2;

int main() {
    auto f1 = [](){ while (! f1done) { ++cnt1; f1work(); } };
    auto f2 = [](){ while (! f2done) { ++cnt2; f2work(); } };
    std::thread t1(f1), t2(f2);
    t1.join();
    t2.join();
}
```
Abstraction Problem

```cpp
int cnt1;
int cnt2;
}
(Possibly) False Sharing

int main() {
    auto f1 = [](){
        while (! f1done) {
            ++cnt1;
            f1work();
        }
    };
    auto f2 = [](){
        while (! f2done) {
            ++cnt2;
            f2work();
        }
    };
    std::thread t1(f1), t2(f2);
    t1.join();
    t2.join();
}
```
Breaking the Abstraction

Matrix Multiplication
Breaking the Abstraction

Matrix Multiplication

Partial !!
While We Are at Matrix Math...

```cpp
Eigen::Matrix<M,N> m1;
Eigen::Matrix<N,P> m2;
auto m3 = m1 * m2;
```

Quality of Implementation
While We Are at Matrix Math...

```cpp
Eigen::Matrix<M,N> m1;
Eigen::Matrix<N,P> m2;
auto m3 = m1 * m2;
```

Quality of Implementation

- Scalar
- SIMD
- "real" vectors (à la Cray)
- Offloaded
  - Transfer in\&out
  - Keep on device
  - ...
One-Size-Fits-All Architecture
Cache Number Explosion

Many different access times:

1+2  Level 1 Cache
3    Level 2 Cache
4    Last Level Cache
5    Caches on neighboring core
6    Caches on distant core
7    Local DRAM
8    Local NVRAM
9    Cache on remote cores
10   On var remote core
11   RAM attached to remote socket
12   NVRAM attached to remote socket
Hot Issue in NNs

Data types
- FP8, FP16, FP32, FP64
- Int4, Int8, Int16, Int32
- Coarse granularity

Why not FP12?
Kernel Steering

- Execution
- Processes
- Kernel Threads
- Memory
- Interrupts

Scheduler

OS

CPU

Core 0

Core 1

Core 2

Core 3

Interrupt Logic

MC

MC

PCIe Hub

PCle Slot

PID 223

PID 975

PID 1079

PID 57

PID 4100

PID 1

User Level

System Level

Software

Hardware
Predictability Matters!

PDF

CDF

min gen = 0 max gen = 0
min gen = 0 max gen = 0
Normal Networking
Kernel Bypass

Process

TCP
IP
Phy

Process

Socket Buffer

OS

UDP
TCP
ICMP
SCTP
IP
Phy

Buffer

NIC
Queue
Queue
Firmware

PCI virtual function
IO Port + DMA Range

PCI virtual function
IO Port + DMA Range
Real-Time

One Approach

Real-Time Kernel

RT Process
RT Process
RT Process

Linux Process
Linux Process
Linux Process

Linux Kernel Process
What If …?
Adjust Compute Environment Dynamically for each Process
Bump-In-The-Wire
Bump-In-The-Wire

Instead of kernel bypass:
• Implement function on FPGA
• Implement decoding on FPGA
  • Reduced communication with host
  • Complex operations on host
Cache Allocation

Restrict shared resource use
- Some hardware support available
- Not process property
- Static configuration
Cache Allocation

Reconfigure hardware
- Generate softcore processor for problem
- For VMs/containers, not processes
- Common hypervisor?
Deploy Unikernels

- Normal OS on bare metal
Deploy Unikernels

- Normal OS on bare metal
- Direct access to devices
  - Virtual functions
  - SR-IOV
- Consecutive memory range
- Dedicated CPU sockets or cores
Architecture

SMART SWITCH

Softcore

Routing Table

Crypto

Network Stack

User Application

Softcore Instruction Memory

Network Stack

CPU NIC

Softcore Programmer

PCIe

(from Ahmed's slides)
Summary
Tasks

- **CPU research**
  - Softcore implementation
  - Reconfigurable with IP blocks addressed through custom instructions
  - Accelerator IP blocks
  - Cache isolation
- **Integrated development platform**
  - Open source HDL toolchain
  - Integration of softcore as accelerator in OpenMP/…
- **Research compiler techniques**
  - C/C++ vectorization
  - DSL, translate to C/C++
  - Unikernel binary generation

- **OS research**
  - Run unikernels as executables
    - Dedicated resources
    - For efficiency, latency, RT
  - (automatic) configure cache isolation
- **FPGA “OS” research**
  - API, basic services, security
  - Toolchain to create services
- **Runtime research**
  - Automatic parallelization control
  - Automatic resource allocation and deallocation
  - Automatic selection of accelerator and workload split
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHatNews