Communication Requirements for FPGA-Centric Molecular Dynamics

Md. Ashfaquzzaman Khan Martin C. Herbordt
Department of Electrical and Computer Engineering
Boston University, Boston, MA

***Abstract*—FPGA-centric clusters use FPGAs for both computation and communication and thereby address three fundamental problems of future High Performance Clusters: efficient use of silicon, power, and removing communication bottlenecks. In this study we report on the plausibility of using such clusters for Molecular Dynamics simulations, in particular by determining the communication requirements for such a cluster. We begin by reviewing MD on a single FPGA-based node and use that performance to determine the time budget for the communication. We then characterize the data communication characteristics for a production MD code (NAMD) in two ways: analytically and by instrumenting the code. We apply this information to clusters of various sizes and node complexity. The conclusion is that a cluster with 256 FPGAs distributed in 64 nodes is appropriately provisioned, even for modest simulations, with a bidirectional 3D torus where each link consists of 1-2 of an FPGAs serial ports.**

***Keywords*-High Performance Reconfigurable Computing; Molecular Dynamics; FPGA-Centric Compute Clusters**

I. Introduction

Molecular Dynamics (MD) is one of the workhorses of computational science. In medicine and biology its impact can be seen at PUBMED which shows over 2,000 articles in the last year alone. MD is also compute bound. Large scale experiments involve thousands compute node-days. These have included simulations of viruses (9,000 node-days [1]) and ion channels (50,000 node-days per significant event [2]). Achieving high performance for MD has therefore received much attention, including effecting scalability on large clusters [3], [4], creating dedicated hardware [5], [6], and deploying GPUs [7] and FPGAs [8], [9] as accelerators. There yet remains a many order-of-magnitude gap between the largest current simulations and the potential physical systems to be studied. This is the difference between current heroic computations, which simulate millions of particles for microseconds, and the simulation of cell-level phenomena at biologically relevant time-scales (billions of particles and more for milliseconds to seconds).

A critical issue in MD is data movement. By Amdahls Law, scalability of parallel applications that have significant communication depends on the number of nodes, the problem size, and the ratio of communication to computation (comm/comp) hardware support. Right now MD can scale well for thousands of nodes for well-designed codes (e.g., [3], [4]) and sufficiently large simulations, or for much smaller simulations using dedicated hardware [10]. The problems are as follows:

1) **Technology trend.** Nodes will continue to get ever more powerful. Process technology appears able to advance with Moores Law for a few more generations even while operating frequencies remain static or decrease. And the fraction of silicon available for computation is increasing both through the use of accelerators (on and off chip) and with the increased emphasis on performance in new generation high-end CPUs [11]. All of these factors decrease the comm/comp support ratio.
2) **Scaling with accelerators.** Using accelerators makes scalability more challenging, as seen, e.g., with respect to three of the most prominent MD codes: NAMD [7], [12], AMBER [13], and GROMACS [14]. The problem is two-fold: (i) there is additional overhead to move data onto and off of the accelerator and (ii) the reduction in compute time per time-step makes the communication latency harder to hide.
3) **Overall cost.** Large computer systems have enormous costs for acquisition, management, power, and cooling, necessarily making them a limited resource. This provides a fundamental limit to the number and size of problems that can be addressed.
4) **Long time-scale simulations.** At some point, no matter what the hardware, the number of nodes will become too large or the problem size too small to achieve strong scaling. It appears, however, that significant MD problems in the range of 10s of thousands of particles can currently achieve strong scaling only with specialized hardware [10]. This lower limit on problem size increases with both number of nodes and node size. As a consequence, achieving long time scales for problems of *any* size is difficult. But this lower limit *increases* with better integration of communication and computation to reduce overhead and latency.

The implications are as follows. Overall cost will be reduced by continued use of ever more powerful commodity accelerators and integrated components as they become available. But running simulations at biological time-scales

This work was supported in part by the NIH through award #R01-RR023168-01A1 and by the MGHPCC.

will require ever tighter integration of communication and computation. That is, not only is low-latency communication critical now, but it is only going to get more so. In current high-end clusters this communication includes transfers between accelerator and CPU. While projected integration of accelerator and CPU will help, the other problems remain. The obvious solution has two parts: (i) provisioning nodes with appropriate bandwidth and (ii) reducing latency by integrating communication directly into the computation chip. The first part is straightforward, while the second, although more challenging, has been well studied. Here are some solutions, past, present, and future:

Past: In the early 1990s several projects looked at integrating communication and run-time systems into the chip fabric (e.g., [15], [16]) and reduced latency of the entire communication stack to a few cycles.
Present: The Anton processor [6] uses dedicated hardware to pipeline communication with computation and effectively reduce amortized communication latency to zero. An alternative, described here, is using FPGAs for both communication and computation.
Future: The need for direct communication solutions for GPUs has been stated by Patterson as one of his Top Three Next Challenges [17] and proposed by Dally in his plan for GPUs as Exascale accelerators [18]. The Intel Xeon Phi has an integrated network interface that could potentially be used for direct coprocessor interconnects [19].

What we examine in this work is a solution for the present that is built entirely with commodity hardware. In particular, we examine the possibility of, and requirements for, large-scale clusters scalable for MD based on FPGAs as the central component *for both computation and communication*. This design is motivated as follows.

- In previous work we have demonstrated that FPGAs can be competitive for single node MD acceleration. In particular, the range-limited parts of the 93K APOa1 benchmark can be computed in less than 25ms per time-step [9] using 2008-era technology. The range-limited force is computed with full electrostatics and is compatible with production MD codes. Updating to the (current) Stratix-V should more than double this performance.
- Many prototype FPGA-centric clusters have been built and include products from BEEcube [20] and SciEngines [21]. One example, the Novo-G Reconfigurable Supercomputer at the University of Florida, has a peak performance of 10 PetaOPs (32 bit) or 100 GigaFLOPs while drawing less than 12KW power and requiring no additional cooling infrastructure [22].
- The primary market for high-end FPGAs is as communication processors, especially in high-end routers [23], [24]. The use of FPGAs for communication offload is also well established. For example BittWare (www.bittware.com) has long had products that use this approach for large-scale DSP. For HPC, computers with FPGAs for communication are currently in use for Physics computations [25], [26].

Expanding on the idea of using FPGA for HPC communication, there are three attributes that make them ideal for this purpose: (i) huge available bandwidth, (ii) low latency through direct connection with the application processor, and (iii) intelligent channels, or the ability to process data as it streams through the interface.

We begin by reviewing MD on a single FPGA-based node and use that performance to determine the time budget for the communication. We then characterize the data communication characteristics for a production MD code (NAMD) in two ways: analytically and by instrumenting the code. We apply this information to clusters of various sizes and node complexity. We find that a cluster with 256 FPGAs distributed in 64 nodes is appropriately provisioned, even for modest simulations, with a bidirectional 3D torus where each link consists of 1-2 serial ports. This is a small fraction of the available communication capability in current high-end FPGAs.

II. Background

A. MD Overview

MD is an iterative application of Newtonian mechanics to ensembles of atoms and molecules. MD simulations generally proceed in iterations (time-steps) each of which consists of two phases, force computation and motion integration. In general, the forces depend on the physical system being simulated and may include LJ, Coulomb, hydrogen bond, and various covalent bond terms:

$$\mathbf{F}^{total} = F^{bond} + F^{angle} + F^{torsion} + F^{HBond} + F^{non-bonded} \quad (1)$$

Because the hydrogen bond and covalent terms (bond, angle, and torsion) affect only neighboring atoms, computing their effect is $O(N)$ in the number of particles N being simulated. The motion integration computation is also $O(N)$. Although some of these $O(N)$ terms are easily computed on an FPGA, their complexity is low and we ignore them for the rest of this preliminary study. The LJ force for particle i can be expressed as:

$$\mathbf{F}_i^{LJ} = \sum_{j \neq i} \frac{\epsilon_{ab}}{\sigma_{ab}^2} \left\{ 12 \left(\frac{\sigma_{ab}}{|r_{ji}|} \right)^{14} - 6 \left(\frac{\sigma_{ab}}{|r_{ji}|} \right)^{8} \right\} \mathbf{r}_{ji} \quad (2)$$

where the ϵ_{ab} and σ_{ab} are parameters related to the types of particles, i.e. particle i is type a and particle j is type b. The Coulombic force can be expressed as:

$$\mathbf{F}_i^{C} = q_i \sum_{j \neq i} \left(\frac{q_j}{|\mathbf{r}_{ji}|^3} \right) \mathbf{r}_{ji} \quad (3)$$

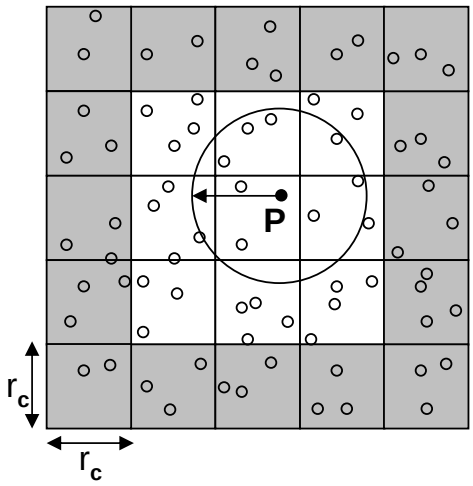

Figure 1. Shown is part of the simulation space about particle P. Its two dimensional *cell neighborhood* is shown in white; cells have edge size equal to the cut-off radius. The cut-off circle is shown; particles within the circle are in P's neighbor list.

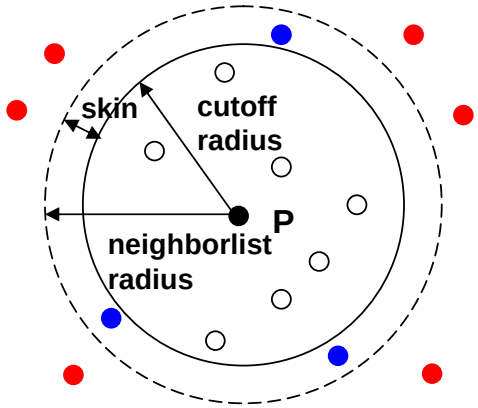

Figure 2. Neighbor lists are often computed for a larger radius than the cutoff.

A standard way of computing the non-bonded forces (Lennard-Jones or LJ and Coulombic) is by applying a cut-off. Then the force on each particle is the result of only particles within the cut-off radius r_c. Since this radius is typically less than a tenth of the size per dimension of the system under study, the savings are tremendous, even given the more complex bookkeeping required. Moreover, this allows the range-limited computation to scale linearly with simulation size.

The problem with cut-off is that, while it may be sufficiently accurate for the rapidly decreasing LJ force, the error introduced in the slowly declining Coulombic force may be unacceptable. A number of methods have been developed to address this issue with some of the most popular being based on Ewald Sums (see, e.g., [27]). Here we use the standard convention of calling *range-limited* the LJ force and the Coulombic force generated within a cut-off radius. We refer to the Coulombic force generated outside the cut-off radius as *long-range*.

We now briefly describe the characteristics of the MD computation that affect communication. Calculating the range-limited force involves, for each particle, summing the contributions of all particles within the cut-off radius. Two methods are used to take advantage of this cut-off: cell lists and neighbor lists (see Figure 1). With cell lists, the simulation space is typically partitioned into cubes with edge-length roughly equal to r_c. Non-zero forces on the *reference particle* P can then only be applied by other particles in its *home cell* and in the 26 neighboring cells (the 3x3x3 *cell neighborhood*). We refer the second particle of the pair as the *partner particle*. With neighbor lists, P has associated with it a list of exactly those partner particles within r_c.

Most MD codes reuse the neighbor lists for multiple iterations and so amortize the work in their creation. But because particles move during each iteration, particles can enter and exit the cut-off region leading to potential error. The solution is to make the neighborlist cut-off larger than the force cut-off, e.g., 13.5Å versus 12Å (see Figure 2).

Calculating the long-range force using, say, PME [27] involves approximating particle charges with a grid and then applying a transform to that grid. The mapping stage is mostly local as the particles only contribute to grid points in the immediate neighborhood. The transform generally involves a transpose which requires all-to-all communication.

B. FPGA-Based Systems

We briefly state our assumptions about the target systems with FPGA-based accelerators. They are typical for current products.

- The overall system consists of some number of standard nodes. Typical node configurations have 1-4 accelerator boards plugged into a high-speed connection (e.g., the Front Side Bus or PCI Express). The host node runs the main application program. CPUs communicate with the accelerators through function calls.
- Each accelerator board consists of 1-4 FPGAs, memory, and a bus interface. On-board memory is tightly coupled to each FPGA typically through several interfaces (e.g., 3x128-bit) which can be virtualized into a much larger number of independent streams, using, e.g., the PROCMultiPort tool from Gidel [28]. 4GB-64GB of memory per FPGA is currently standard.
- Besides configurable logic, the FPGA has dedicated components such as independently accessible multiport memories (e.g., 2000 x 1KB) called Block RAMs (or BRAMs) and a similar number of multipliers. FPGAs used in High Performance Reconfigurable Computing typically run at 200 MHz, although with optimization

substantially higher operating frequencies can sometimes be achieved.

- FPGAs have substantial I/O and communication capability. High-end FPGAs have on the order of 1000 I/O pins which have latency of 5-6 ns. They also have dozens Multi-Gigabit/s Tranceivers (MGTs). For example, the Xilinx XC7VH870T has 16 28.05Gb and 72 13.1Gb MGTs [29] while the Altera Stratix 5SGXBB has 66 14.1Gb MGTs [30]. For tightly coupled applications, end-to-end latency using the MGTs for FPGA-FPGA communication can be less than 100ns. FPGAs on a board typically communicate via I/O with latency of a few cycles while internode FPGA-FPGA communication is typically done via the Gb interconnects.

We are initially targeting the Novo-G, the High Performance Reconfigurable Supercomputer at the University of Florida [22], which has nearly 400 FPGAs and both a COTS interconnect and direct FPGA-FPGA connections.

C. MD on FPGA-Based Systems

We now give an overview of the overall accelerator design (see Figure 3; for details see [9]) with the goal of justifying the communication budget in the next Section.

Our accelerated MD system is currently running on four FPGAs of a Gidel PROCStar III board [31]. The PROCStar III is a PCI based system with an 8-lane PCI Express (PCIe x8) host interface. Each processing unit contains an Altera Stratix III SE260 FPGA and three memory banks, each of which has a 128-bit interface. The system has also been tested in simulation on an Altera Stratix-V, the current generation FPGA, and has been integrated into and validated with the full NAMD system. The host CPU runs the main application program and communicates with the accelerator board through function calls. The program is partitioned as follows. The accelerators process the range-limited forces, while the CPU cores process the balance of the computation. Each iteration, new particle positions are downloaded to the accelerator and forces are uploaded to the CPU.

Main computation pipeline
The main computation pipeline is partitioned into two levels. The first is the filter pipeline; it determines whether the particle pair has a non-zero force. The second level, the force pipeline, accepts the particle pairs that pass the filter and computes their mutual force. Six to eight force pipelines fit on the 260E, each with 8-10 filter pipelines. This number doubles for the Stratix-V.

Host-accelerator data transfers
At the highest level, processing is built around the timestep iteration and its two phases: force calculation and motion update. During each iteration, the host transfers position data to, and acceleration data from, the coprocessor's on-board memory (POS SRAM and ACC SRAM, respectively).

Board-level data transfers

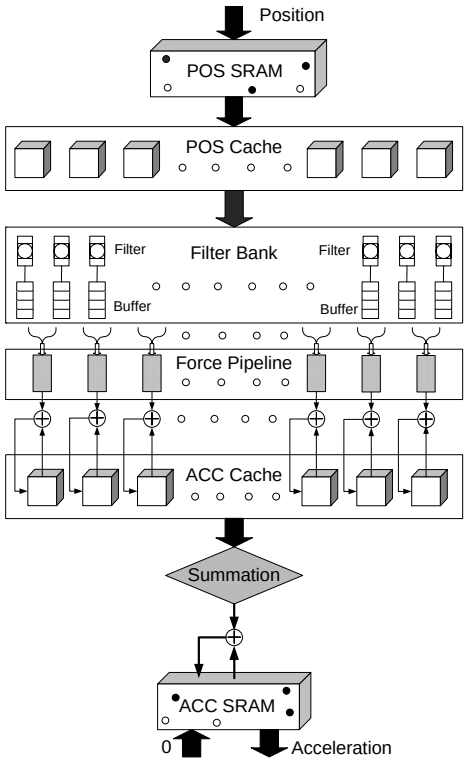

Figure 3. Schematic of the FPGA part of the single node MD system.

Force calculation is built around the processing of successive home cells. Position and acceleration data of the particles in the cell set are loaded from board memory into on-chip caches, POS and ACC, respectively. When the processing of a home cell has completed, ACC data is written back. Focus shifts and a neighboring cell becomes the new home cell.

Force pipelines to ACC cache.
To support an optimization due to Newton's Third Law, two copies are made of each computed force. One is accumulated with the current reference particle. The other is stored by index in one of the large BRAMs on the Stratix.

The performance for the range-limited force computation is as follows. Each of the 12-16 pipelines on the Stratix-V runs at 200MHz and completes a payload (neighbor list) force calculation every cycle. All data transfer latencies are hidden as described in [9] yielding a compute time of less than 20ms per time-step, or roughly 10x the speed of an 8-core CPU.

We have not yet implemented the 3D FFT, but highly efficient IP is available for 1D and 2D FFTs from both Xilinx and Altera. For charge assignment from particles to the 3D grid and force mapping from 3D grid to particles we use methods previously developed in our lab for computing electrostatics with Multigrid using FPGAs [32], [33]. The remaining parts of the computation are mostly the communication described in the next Section.

III. MD Communication and Support Requirements

In the first Subsection we describe the major types of computations. In the second we quantify the communication analytically and experimentally.

A. MD Communication Description

Several tiers of inter-processor communication take place during a parallel run of MD. The majority of these transfers are due to the non-bonded force computation, while most the rest are required for maintenance of the simulation, e.g., re-assignment of particles to nodes as particles move during each timestep.

1) Range-limited: Position data of each particle needs to be updated every timestep before the range-limited non-bonded forces can be computed for that particle. The owner node of a particle (the node that is responsible for updating the motion data of that particle) sends position data (and also charge data as required) to all nodes that require this data at the beginning of a timestep. After computing the forces, these nodes send the force data back to the owner node. The owner node then combines these results to update the motion of that particle for that timestep.

With cell-based decomposition (or neighbor lists) the amount of communication per particle is a constant with respect to the problem size and the number of processors. Assuming spatial decomposition and assignment of those spatially decomposed cells to nodes we have two scenarios defined by the ratio of problem size to cluster size.

* For large problem to cluster size, multiple cells are assigned to each node, potentially cubes of slabs. If the number of cells per node is the same for all nodes, each node can simply compute forces for 13 neighboring cells for each of the cells that is assigned to it. If that is not the case, e.g., if some nodes own two cells, some own 3 cells, then additional decomposition involving the force computation improves load-balance.

* For small problem to cluster size, multiple nodes are assigned to each cell. There is further decomposition of force computations between neighbor pairs to nodes. That is, the force computation of a cell will be decomposed such that it can be done in parallel by multiple nodes. In the simplest case, there are 13 cell-pairs for each cell. These 13 cell-pairs are assigned to multiple nodes for computation. The motion integration will still take place in one node.

Data communication for range-limited force computation is typically limited to neighboring nodes, as long as node assignment corresponds to the physical simulation space.

2) Grid Interpolation: For the long-range part of the electrostatic force computation, there are potentially two sets of three communication operations. First the charge contributions of particles to nearby grid points needs to be interpolated. This contribution is computed by the owner node and is sent to the processors responsible for those particular grid points. Second, the charge grid is redistributed in preparation for the FFT, e.g., so that slabs of the charge grid are distributed among nodes. Third, these grid data are then used to compute forces in Fourier space, using the 3D FFT. This involves a 3D transpose. These operations are then repeated in reverse order until the forces are applied to the particles by the owner node for motion update.

Grid interpolation thus requires two sets of two communications: one to sum (apply) the grid contributions and one to linearize (delinearize) the grid. The first is local while the second involves one-to-many (many-to-one) communication along dimensions.

3) FFT/Transpose: The 3D FFT-based long-range force computation requires all-to-all communication among nodes. if the 3D FFT uses slab-based decomposition, then this involves sending/receiving transpose data once each for the forward and reverse FFT. For pencil-based decomposition, this needs to happen twice for both the forward and reverse FFT.

4) Others (Migration, Bonded Forces, etc.): While the above three types of communication comprise the majority of the data communication, there are a few other types of communication required to maintain the simulation. These include migration of particles and other system-wide communication among nodes, e.g. synchronization. Migration becomes necessary when a particle crosses the boundary of its current cell by a predetermined margin. At that point, it is re-assigned to another cell and this information must be communicated among the involved nodes. Communication for bonded forces is necessary when bonded particles reside on different nodes. In both cases, data communication is only a fraction of other communication and is limited to among neighboring nodes.

B. MD Communication Characterization

Communication in MD is primarily determined by the size of the problem, the number of computing nodes, and the compute capability of the nodes. In this Subsection we provide some simple formulas to predict this communication amount and validate them using a production MD code. The data are shown per node, assuming (unrealistically for large systems) a direct communication link between every node-pair. We use NAMD2.8 [34] and the ApoA1 benchmark for measuring data communication. The source code of NAMD2.8 was instrumented for this purpose. The ApoA1 benchmark consists of 92,224 particles and uses periodic boundary condition with an original simulation box of $108\mathring{A} \times 108\mathring{A} \times 78\mathring{A}$. It uses a cut-off radius of $12\mathring{A}$ for the range-limited force computation and a switching function is applied to smooth the force when the inter-particle distance is between $10\mathring{A}$ and $12\mathring{A}$. The Coulomb force is evaluated using PME. The cell (*patch* in NAMD terminology) dimension used for cell-decomposition is approximately $18\mathring{A}$,

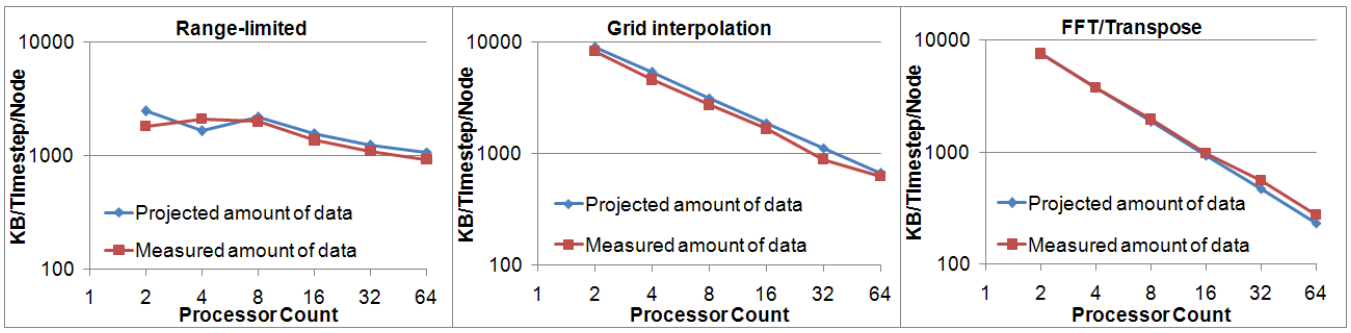

Figure 4. Projected and measured data communication

which results in $6 \times 6 \times 4$ cells. The number of grid points used for PME is $108 \times 108 \times 80$.

1) Range-limited: For a large number of cells per node, the communication required for the range-limited force can be approximated by assuming the import region to be a skin of a sphere with a depth of r_c. If the volume of region owned by a node is $4 \times \pi \times r^3/3$, then the import volume would be $4 \times \pi \times (r + r_c)^3/3 - 4 \times \pi \times r^3/3$, where r is the radius of the spherical volume owned by the node and r_c is the inter-particle interaction distance. The number of particles can be derived by using the density of the system, which is about 0.1 for a typical bio-medical system.

For a small number of cells per node, this approximation does not hold due to the cell-based decomposition. However, assuming that neighboring cells are assigned to each node, we can predict the communication using the following equation

$$D = (C + 13) \times P \times d \times 2 \quad (4)$$

where D is the amount of data per node, C is the number of cells per node, P is the number of particles per cell and d is the amount of data amount per particle. The 2 at the end of the equation is for a node's contribution to other nodes.

For multiple nodes per cell, most of the computation requires importing data from other nodes. This results in a communication amount of (Number of compute objects per node + 1) $\times$ (Number of Particles per cell) $\times$ (Amount of Data per particle) $\times$ 2.

The left panel of Figure 4 shows the projected and measured data communication for range-limited force computation. For low processor counts (2 and 4), where there are many cells per node, we use import volume; for the rest we use Equation 4

2) Grid Interpolation: The communication for the grid interpolation can be approximated by the following equation

$$D = (G/n) \times d \times 4 \times n^{1/4} \quad (5)$$

where D is the amount of data per node, G is the total number of grid points, n is the number of nodes and d is the amount of data per grid point. The 4 in the equation accounts for sending/receiving of grid data before and after the force computation. The $n^{1/4}$ is empirically determined and not yet accounted for analytically.

The center panel of Figure 4 shows the projected and measured data communication for grid interpolation.

3) FFT/Transpose: The communication for 3D FFT can be approximated by the following equation (for slab-based decomposition)

$$D = (G/n) \times d \times 4 \quad (6)$$

where D is the amount of data per node, G is the total number of grid points, n is the number of nodes and d is the amount of data per grid point. The 4 in the equation accounts for sending/receiving of grid data in the forward and reverse FFTs.

The right panel of Figure 4 shows the projected and measured data communication for FFT/transpose stages.

IV. FPGA Cluster Communication: Design Sketch and Requirements

In this Section we estimate the communication requirements for MD of an FPGA-centric cluster. We first summarize its generic characteristics.

Nodes. We assume direct FPGA-FPGA connections with MGTs among boards making the board a convenient designation of node granularity. FPGAs within a board have a tighter interconnect based on the LVDS I/O. We assume that each of the four FPGAs replaces 64 cores giving the node a capability of 256 cores.

Topology. To take advantage of spatial locality we assume a 3D torus.

Communication intelligence. The programmability of the FPGA enables the creation of channels capable of processing data. Examples are as follows.

- For the range limited force computation, particles can be filtered by position so that only those with a non-trivial partners get transferred to any neighboring node.

- For the transpose during the FFT, data reordering facilitates highly efficient communication [35], [36].
- In general, data can be reordered according to a script to support vector chaining enabling correct data to arrive at the computation units as needed. This technique has been used by the Anton processor with great success.

Bandwidth. The overall available bandwidth of the aggregate MGTs of most high-end FPGA is over 5Tb/s. The communication mechanisms just outlined increase the effective bandwidth, but more likely will be used to reduce the fraction that must actually be implemented.

Latency. Within the FPGA, data can be transferred to the communication channels in just a few cycles yielding a time-of-flight latency from application to application of less than 100ns. The communication mechanisms just outlined are critical in actually achieving this latency.

Our goal now is to determine the channel bandwidth needed. We start by establishing a first-order time per timestep budget from the computation alone. We then extend this to an ideal fully connected system. After that, we add the assumption of the 3D torus with in-channel filtering.

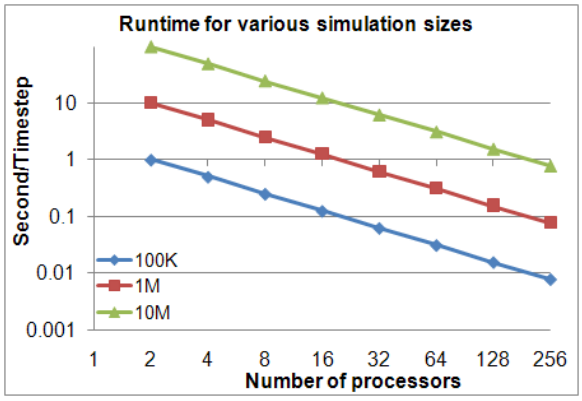

Figure 5. Time per timestep for various simulation sizes and core counts assuming perfect scaling. This is computation only and gives the time budget for communication.

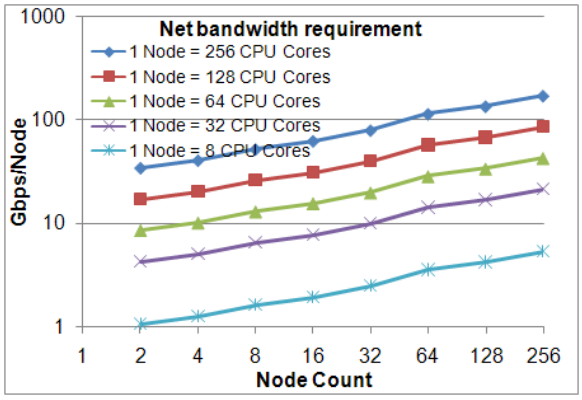

Figure 6. Bandwidth requirement for various systems for a 100K particle problem size. Systems are ideal with all-to-all interconnect and no in-channel particle filtering.

Figure 5 shows an estimate of required time per timestep for various simulation sizes on a CPU-only system, assuming perfect scalability, and as derived from the 92K ApoA1 benchmark results at the NAMD web site [34]. This data, along with the communication characterization from the previous Section, can be used to determine the required bandwidth per node. Figure 6 shows these results. The values for node counts 128 and 256 are computed using the analytical methods from the previous Section.

The bandwidth requirement of an accelerator-based system can be roughly determined by comparing to an equivalent number of CPU cores. With high-end FPGAs, it should be possible to achieve order of magnitude speed-up over an 8 core CPU, assuming communication is also improved as required. This is shown in Figure 6 in terms of CPU core-equivalence. For example, an FPGA-based node equivalent to 256 CPU cores will have the bandwidth requirement of the top-most line in this graph.

It should be noted that several adjustments are likely to be necessary as system and implementation are specified further. The bandwidth requirement shown is the absolute amount of data, assuming an all-to-all network. In practice, we must consider the header, the number of hops, and other implementation issues. Therefore, actual bandwidth requirement is likely to be at least twice that shown in Figure 6. Although this data is derived using a benchmark of about 100K particles, it should give a reasonable estimate for larger simulations too, since both runtime/timestep and amount of data communication will increase with the size of the simulation. Another point to note is that this estimate includes intra-node communication, which is significant at low node counts, but can be ignored for high node counts. That is, at low node counts, the observed inter-node bandwidth requirement will be significantly lower than what we present here.

We now examine the communication at the packet level to determine whether latency (time-of-flight) is likely to have an impact on channel provisioning. That is, we determine whether packet count must be considered (in this preliminary study) or bandwidth alone is adequate. A packet for range-limited computation consists of all position/charge/force data of all particles in a cell. With an $18\mathring{A}$ cell and a particle density of 0.1 particles per cubic angstrom, the average number of particles in a cell is approximately 600. With double-precision floating point, the position/charge data size is 32 bytes per particle, while force size amount is 24 bytes per particle giving an average packet size of 16KB. With a 14Gbps MGT, this corresponds to 8.72 microseconds per data packet. This time is significantly larger than what we assume for inter-node data latency (a few hundred nanoseconds). An efficient implementation is likely to be able to hide this latency. Similar arguments hold for grid interpolation

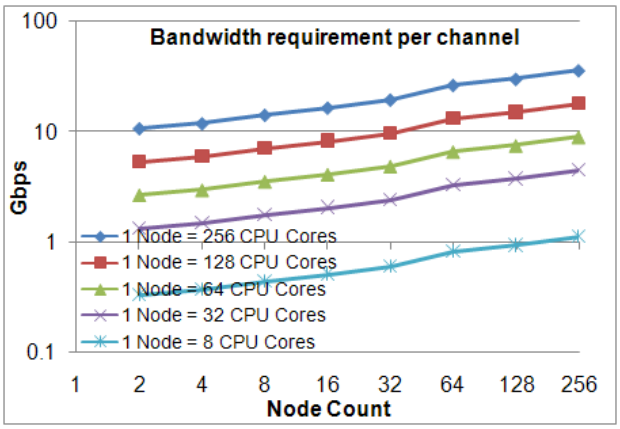

Figure 7. Bandwidth per channel requirement for various systems for a 100K problem size. Some likely system information is integrated such as number of hops per packet and in-channel particle filtering.

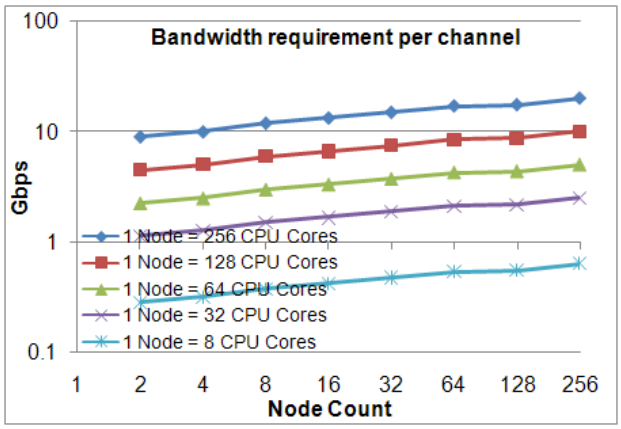

Figure 8. Bandwidth per channel requirement for various systems for a 1M problem size. Some likely system information is integrated such as number of hops per packet and in-channel particle filtering.

and FFT/transpose. Therefore, for further discussion, we only consider bandwidth.

Next we determine the actual bandwidth requirement, assuming a 3D bi-directional torus network. Our goal here is to determine the bandwidth needed for each of the 12 channels on a node in such a network topology. Data communication for the range-limited non-bonded force computation is contained within neighboring nodes at 1-3 hops. This will on average cause about a $2\times$ increase in data communication. At the same time, however, the FPGA easily supports in-channel filtering to remove particles not needed by a particular neighbor. For cell/patch and cut-off sizes described earlier, this results in a reduction of data to be transferred (weighted by number of hops) to 73% of the original. For long-range communication, all-to-all communication is required which roughly doubles the data amount for a $4 \times 4 \times 4$ node system and further doubles it on an $8 \times 8 \times 8$ node system.

The final bandwidth requirement for a 100K simulation is shown in Figure 7. Again, the series `node = 256 cores` represents projected performance of a 4 FPGA node. For a system with 64 such nodes, configured in a 3D bidirectional torus, each channel must support 27Gb/s bandwidth; this is possible with 2 14Gb/s serial links. The aggregate of 24 links is a small fraction of the roughly 200 available among the four FPGAs on such a node.

Figure 8 shows the same results for a 1M particle simulation. As expected, increasing the problem size significantly reduces the communication requirement. Again examining the series `node = 256 cores` we see that the last three points (64, 128, and 256 nodes) require bandwidths of 17.2, 17.7, and 20.3 Gb/s, respectively. This approaches what can be achieved with a single serial port, especially in the next generation of FPGAs where the link capacity is likely to double.

V. Discussion and Future Work

We have described an initial requirements study for the communication network for an FPGA-centric cluster executing Molecular Dynamics simulations. We find that current production boards (with 4 FPGAs) are appropriate for use as nodes in such a system. Even for relatively small simulations which are hard to scale to large clusters ($<$100K particles) we find that only a small fraction of the FPGAs communication capability is required. Some of this is due to the fact that the FPGA channel is programmable which can significantly reduce the amount of data that needs to be transferred. We are investigating the implementation of such a network on the Novo-G. The next steps are to formalize the internal communication of such a node, including integration of the various force types.

References

[1] P. Freddolino, A. Arkhipov, S. Larson, A. McPherson, and K. Schulten, "Molecular dynamics simulations of the complete satellite tobacco mosaic virus," *Structure*, vol. 14, pp. 437–449, 2006.

[2] F. Khalili-Araghi, E. Tajkhorshid, and K. Schulten, "Dynamics of K^+ ion conduction through Kv1.2," *Biophysical Journal: Biophysical Letters*, vol. 91, pp. L72–L74, 2006.

[3] Bowers, K.J., et al., "Scalable algorithms for molecular dynamics simulations on commodity clusters," in *Proc. ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis – Supercomputing*, 2006.

[4] J. Phillips, G. Zheng, and L. Kale, "NAMD: biomolecular simulation on thousands of processors," in *Proc. ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis – Supercomputing*, 2002.

[5] Y. Komeiji, M. Uebayasi, R. Takata, A. Shimizu, K. Itsukashi, and M. Taiji, "Fast and accurate molecular dynamics simulation of a protein using a special-purpose computer," *Journal of Computational Chemistry*, vol. 18, no. 12, pp. 1546–1563, 1997.

[6] Shaw, D.E., et al., "Anton, a special-purpose machine for molecular dynamics simulation," in *Proc. International Symp. on Computer Architecture*, 2007, pp. 1–12.

[7] J. Stone, D. Hardy, I. Ufimtsev, and K. Schulten, "GPU-Accelerated Molecular Modeling Coming of Age," *Journal of Molecular Graphics and Modelling*, vol. 29, pp. 116–125, 2010.

[8] S. Alam, P. Agarwal, M. Smith, J. Vetter, and D. Caliga, "Using FPGA devices to accelerate biomolecular simulations," *Computer*, vol. 40, no. 3, pp. 66–73, 2007.

[9] M. Chiu and M. Herbordt, "Molecular dynamics simulations on high performance reconfigurable computing systems," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 3, no. 4, pp. 1–37, 2010.

[10] D. Shaw, R. Dror, J. Salmon, J. Grossman, K. Mackenzie, J. Bank, C. Young, M. Deneroff, B. B. K.J., E. Chow, M. Eastwood, D. Ierardi, J. Klepeis, J. Kuskin, R. Larson, K. Lindorff-Larsen, P. Maragakis, M. Moraes, S. Piana, Y. Shan, and B. Towles, "Millisecond-scale molecular dynamics simulations on Anton," in *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 1–11.

[11] A. Vladimirov, "Arithmetics on Intel's Sandy Bridge and Westmere CPUs: not all FLOPs are created equal," Colfax International, Tech. Rep. http://research.colfaxinternational.com/, 2012.

[12] J. Phillips, J. Stone, and K. Schulten, "Adapting a message-driven parallel application to GPU-accelerated clusters," in *Proc. ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis – Supercomputing*, 2008.

[13] G. Shainer, A. Ayoub, P.Lui, and T.Liu, "Raising the Speed Limit: New GPU-GPU Communications Model Increases Cluster Efficiency," *Scientific Computing: Information Technology For Science*, 2011.

[14] GROMACS Group, www.gromacs.org/ Downloads/ Installation_Instructions/ GPUs.

[15] W. J. Dally and et al., "The Message-Driven Processor: A multicomputer processing node with efficient mechanisms," *IEEE Micro*, vol. 12, no. 2, pp. 194–205, 1994.

[16] D. Henry and C. Joerg, "A tightly coupled processor network interface," in *Proc. International Conference on Architectural Support for Programming Languages and Operating Systems*, 1992, pp. 111–122.

[17] D. Patterson, "The Top 10 Innovations in the New NVIDIA Fermi Architecture, and the Top 3 Next Challenges," Fermi White Paper, available at http://www.nvidia.com/ content/PDF/ fermi_white_papers/ D.Patterson_Top10InnovationsInNVIDIAFermi.pdf, 2009.

[18] W. Dally, "Throughput computing," Keynote Talk, International Conference on Supercomputing, May 2010.

[19] Intel, "Programming Models for Intel Xeon processors and Intel Xeon Phi Coprocessors," Intel Sponsor Talk, Symposium on Application Accelerators for High Performance Computing, July 10 2012.

[20] *BEEcube web site*, BEEcube, Inc., www.beecube.com, Accessed 5/2012.

[21] *SciEngines web site*, SciEngines Massively Parallel Computing, www.sciengines.com, Accessed 5/2012.

[22] A. George, H. Lam, and G. Stitt, "Novo-G: At the Forefront of Scalable Reconfigurable Computing," *Computing in Science and Engineering*, vol. 13, no. 1, 2011.

[23] J. Bolaria and J. Byrne, *A Guide to FPGAs for Communications*. The Linley Group, 2009.

[24] Cisco Systems, Inc., http://www.cisco.com, accessed 5/2012.

[25] Z. Baker, T. Bhattacharya, P. Graham, R. Gupta, J. Inman, A. Klein, G. Kunde, A. McPherson, M. Stettler, and J. Tripp, "The PetaFlops Router: Harnessing FPGAs and Accelerators for High Performance Computing," in *Proc. High Performance Embedded Computing*, 2009.

[26] S. Rinke and W. Homberg, "QPACE: energy-efficient high performance computing," in *PRACE Workshop: New Languages and Future Prototypes*, 2007.

[27] T. Darden, D. York, and L. Pedersen, "Particle Mesh Ewald: an $N \log(N)$ method for Ewald sums in large systems," *J. of Chemical Physics*, vol. 98, pp. 10 089–10 092, 1993.

[28] *PROCDeveloper's Kit: PROCMutiPort DRAM Controller*, Gidel Reconfigurable Computing, http://www.gidel.com/ProcDev.htm, 2012.

[29] *7 Series FPGAs Overview*, Xilinx, Inc., 2012.

[30] *Altera Product Catalog: Version 12.0*, Altera Corporation, 2010.

[31] *PROCStar III*, Gidel Reconfigurable Computing, http://www.gidel.com/PROCStar

[32] Y. Gu and M. Herbordt, "FPGA-based multigrid computations for molecular dynamics simulations," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2007, pp. 117–126.

[33] T. VanCourt and M. Herbordt, "Application-dependent memory interleaving enables high performance in FPGA-based grid computations," in *Proc. IEEE Conference on Field Programmable Logic and Applications*, 2006, pp. 395–401.

[34] *NAMD web site*, Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign, http://www.ks.uiuc.edu/Research/namd, Accessed 5/2012.

[35] M. Herbordt and P. Swarztrauber, "Towards scalable multicomputer communication through offline routing," Department of Electrical and Computer Engineering, Boston University, Tech. Rep. TR2003-01, 2003.

[36] C. Young, J. Bank, R. Dror, J. Grossman, J. Salmon, and D. Shaw, "A 32x32x32, spatially distributed 3D FFT in four microseconds on Anton," in *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 1–11.