Topology-Aware Space-Shared Co-Analysis of Large-Scale Molecular Dynamics Simulations

Preeti Malakar*‡, Todd Munson*, Christopher Knight*, Venkatram Vishwanath*, Michael E. Papka*†
*Argonne National Laboratory ‡Indian Institute of Technology Kanpur †Northern Illinois University
pmalakar@cse.iitk.ac.in {tmunson, knightc, venkat, papka}@anl.gov

***Abstract*—Analysis of scientific simulation data can be concurrently executed with simulation either in time- or space-shared mode. This mitigates the I/O bottleneck, however it results in either stalling the simulation for performing the analysis or transferring data for analysis. In this paper, we improve the throughput of space-shared in situ analysis of large-scale simulations by topology-aware mapping and optimal process decomposition. We propose node interconnect topology-aware process placement for simulation and analysis to reduce the data movement time. We also present an integer linear program for optimal 3D decompositions of simulation and analysis processes. We demonstrate our approach using molecular dynamics simulation on Mira, Cori and Theta supercomputers. Our mapping schemes, combined with optimal 3D process decomposition and code optimizations resulted in up to 30% lower execution times for space-shared in situ analysis than the default approach. Our mappings also reduce MPI collective I/O times by 10–40%.**

***Index Terms*—MD simulation; analysis; optimization; mapping**

I. Introduction

Supercomputers have enabled high-throughput computational science simulations in several domains such as cosmology, material science and climate. Such simulations have helped scientists to better understand the underlying complex physics in phenomena such as earthquakes, combustion and blood flow in the human body [1]–[5]. These applications solve several complex equations on large domains for high fidelity and produce terabytes of data [6]. This data subsequently needs to be analyzed and visualized to gain a complete understanding of the simulated phenomena. Analysis is typically performed after the simulation has completed, once the entire data is available on stable storage. However, the network bandwidth of modern interconnects are in the range of Gigabytes/s, whereas one can achieve more than a Teraflop/s compute speed per node [7]. Therefore, writing terabytes of data for post hoc analysis not only increases the simulation time due to high I/O times, but also increases the lag between the time when data was generated and when it is analyzed. Thus, with the growing size of output data and the widening gap between I/O bandwidths and processing speeds, there have been increasing efforts to develop in situ analysis techniques [8]–[13] to gain faster insight into simulations. Overlapping I/O with computation [14] may not enable fast analysis of large data [15] due to overheads in writes and subsequent reads. It also incurs significant time in writing large data, as well as requires dedicated I/O servers.

There are a number of ways to perform in situ analysis [16]–[19]. Tightly-coupled (or time-shared) analysis alternately runs on the same process as the simulation. Loosely-coupled analysis can be executed as the same job (space-shared) or as a different application on a different cluster; in both cases, simulation transfers data to the analysis. In the latter case, the two jobs need to run simultaneously for inter-process data transfer, and may require certain features in the cluster workload manager. Simulation and analysis run on the same core in case of time-shared analysis. This affects the runtime due to compulsory or capacity cache misses from two different execution flows and stalls the simulation when analysis executes. Also, the scalability of simulation and analysis can be drastically different. We empirically found that time-shared performs worse than space-shared co-analysis due to the above reasons.

In this work, we consider the space-shared co-analysis workflow, where simulation and analysis run as a single application. Co-analysis can be performed on a subset of MPI processes by splitting the global MPI communicator for simulation and analysis [19]–[22]. Simulation being more compute-intensive runs on a larger fraction of the allocated nodes. This is a viable approach to in situ analysis, especially in today's era which is marked by continued growth in number of cores/node [23]–[26]. Faster parallel analysis can be accomplished by sparing a few cores per node, because increasing cores/node do not necessarily scale the applications [27]. We noted that by using all the 64 cores of Intel KNL [28] for simulation of 10^5 atoms on 8 nodes, there is no noticeable speedup, whereas by sparing 8 cores for analysis, it can be done almost for free. It is challenging to determine the optimal ratio of simulation and analysis processes for co-analysis. This depends on a number of factors such as the simulation and analyses computation profiles, their communication and scaling characteristics, the system architecture, including the number of cores/node, etc. Here, we do not focus on determining the optimal number of simulation and analysis processes. Our work focuses on reducing the overall execution time by better process mapping, given a particular ratio of simulation and analysis tasks. The experimental results are shown with an empirically determined process ratio (from experiments with various ratios on multiple systems, see Appendix).

Space-shared mode incurs data movement overhead, especially when simulation and analysis nodes are several hops away. This occurs with default mapping of processes, where simulation is mapped on processes 0:M-1 and analysis mapped to processes M:N-1 (N is the total number of processes, M is the number of simulation processes). Additionally, since

SC18, November 11-16, 2018, Dallas, Texas, USA

the number of simulation processes are larger than analysis, there is more congestion at cores running analysis due to multiple simultaneous data transfers. In this work, we study executions of simulation and analysis on the same node, which avoids inter-node data transfers. This also mitigates application scalability saturation bottlenecks [29] when running on several cores per node. We further investigate several process mappings that leverage the intra-node interconnect more efficiently for data transfer between simulation and analysis. Efficient mappings are able to avoid congestion within a node, by placing analysis processes on cores that are physically closer to the cores running the simulation. Some of our node-aware mappings are generic, and some are specifically tailored to commonly available networks on chip, such as 2D mesh and ring interconnects [28], [30], [31]. We show that execution times can be reduced by 30% with better process placements in a node. We demonstrate scalability using a scientific simulation code and an MPI collective I/O benchmark on Intel Haswell, Intel Knights Landing (KNL) and IBM PowerPC processors.

Next, we study optimal process decomposition for simulation and analysis, where both require a 3D process grid decomposition for execution. Mapping a 3D grid onto the physical processor topology is an NP-hard problem [32]; mapping two 3D grids in the most communication-efficient layout on any given network is more challenging. We identify a few constraints to solve this in an optimal way using a mixed integer linear program. Using our 3D decompositions, we are able to reduce the execution times by up to 22% due to fewer network stalls and lower communication times across many systems with different network topologies. Finally, we also optimize the LAMMPS molecular dynamics (MD) simulation co-analysis code execution flow and are able to reduce the overall runtime. MD codes typically distribute sub-domains of particles across processes. For co-analysis, it is important to maintain some degree of synchronicity between sub-domains in simulation and analysis processes, thus frequent data transfers might be implied to produce correct results. We profiled the co-analysis execution flow and found several bottlenecks impeding concurrent executions of simulation and analysis. We modified the code and the sequence of steps in analysis to better overlap with computation, which reduced synchronization costs between simulation and analysis due to lower data transfer overheads. For typical long-running simulations, our code optimizations and co-analysis execution can help to significantly reduce runtimes and improve disk utilization, saving precious core-hours and storage space.

Our node interconnect topology-aware mapping schemes, combined with optimal 3D process decomposition and code optimizations resulted in up to 30% lower execution times for space-shared in situ analysis of molecular dynamics simulations, as compared to the default approach. The mappings result in 20–70% lower network stalls, indicating lesser network congestion due to reduced data movement. Our mappings are applicable to other coupled codes (such as [33], [34]) and also reduce MPI collective I/O times. We achieved 10–40% reduction in I/O times of up to 2 TB shared files on Theta and Cori.

II. Related Work

There have been several application-driven efforts to develop in situ analysis of simulations to meet the science needs [9], [12], [19], [21], [35], [36]. Data from large-scale simulations is too large to store and analyze with conventional post hoc analysis [10], [15]. More than $1000\times$ savings in time is possible by doing in situ analysis [37], [38]. Researchers in [21] presented an in situ/co-analysis framework for the HACC cosmological simulation code. They executed a few analyses in time-shared mode and others in space-shared mode. In [19], space-shared analysis for molecular dynamics code was studied to optimize the frequency of the analysis computation. The authors in [12] combined in situ and in-transit analyses to perform topological analyses and generate descriptive statistics. In this case, certain analyses were performed in an in-situ mode and the intermediate results were transferred to staging nodes using DataSpaces [39] for further analysis. In this work, we developed a scalable space-shared in situ analysis for molecular dynamics simulation that optimizes the placement of the simulation and analysis processes as well as improves the 3D grid decomposition for both the simulation and analysis. Our work is directly applicable in cases where the job partition is shared among simulation and analysis processes [20], [21].

Several in situ infrastructures have been developed [40]–[46]. Damaris [47] uses a few dedicated cores on each node to do asynchronous I/O and analysis. Our mappings can improve its performance. Paraview Catalyst [44] couples simulation and in situ visualization/analysis tasks. VisIt's Libsim [45], an in situ visualization library, requests data from simulation code as needed. Dreher et al. [22] assemble processing components of analysis workflow in a dataflow graph to deploy the workflow. SENSEI [46] is a generic in situ interface used to instrument a simulation code, which can use other infrastructures to transfer data between simulation and analysis that run as different jobs. Our approach can be leveraged by these infrastructures to optimize the placement of analysis and simulation processes. Also, we exploit application domain knowledge to reduce synchronization overhead and better scale in situ analysis, and, thus the overall execution of MD simulations.

Deciding an optimal task/process placement is an NP-complete problem. There are several efforts to improve application performance using topology-aware mappings [48]–[53]. Generic mapping schemes that use application traces [53] lack the ability to distinguish between a simulation's communication from that of the analysis. Graph algorithms [49] have been proposed to use process affinity and communication matrices for process binding. However, in our case, we have coupled simulation and analysis executions within a job, each with their own set of MPI communications and scalabilities, which makes it more challenging to apply the above methods. We consider node interconnect-level congestion to decide the mappings that minimize performance impact on both simulation and analysis. Furthermore, we incorporate dimensionality constraints in our integer linear program formulation to minimize the overall execution times on any given system. Aupy et al. [29] use cache partitioning to maximize performance of multiple iterative applications co-scheduled on the same node. In our work, we co-schedule two different execution flows of the same application using NoC-aware mappings to reduce data movement times.

III. Challenges of MD Simulation Co-analysis

Molecular dynamics (MD) simulations help to understand the physics and chemistry governing a vast array of systems such as liquids, biomolecules and materials [54]. Particle trajectories in MD are generated by integrating Newton's equations of motion propagating them forward in time using the forces derived from interaction potentials. The integration time step is sufficiently small to adequately sample the highest frequency motions in the system, typically 0.5–2 femtoseconds for atomistic models. Thus, sufficiently long trajectories resulting from multi-million step simulations are necessary in order to sample physically relevant timescales (e.g. nanoseconds), with significantly longer simulations required (e.g. microseconds) in order to explore increasingly more complex phenomena (e.g. protein folding, polymer entanglement). When combined with the relatively large length scales required to properly address some scientific questions, large-scale MD simulations requiring multi-million particle systems can be computationally expensive.

Co-analysis requires two groups of processes (MPI communicator split) to perform simulation and analysis – simulation usually runs on the larger partition. We focus on space-shared co-analysis because time-shared performs $\sim 20\%$ worse than space-shared. MD analysis consists of computation of per-atom, or local and global properties of groups of atoms, such as mean square displacement, kinetic energy etc. Thus, the simulation partition sends particle positions, velocities and tags to the analysis partition for analyses. The simulation processes maintain neighbor lists that provide lists of those nearby particles within a specified cutoff distance that are most likely to contribute to short-range interactions. These lists are periodically updated and it is at this time that particles are also exchanged between neighboring processors if they have traveled beyond the local sub-domain of a process (see below). When these exchanges occur, additional per-atom data, such as property histories and bond partners, are also communicated to the neighboring processors. This implies that the analysis partition also needs to exchange particles in sync with the simulation partition to correctly keep track of the per-atom data unique to each particle, in the most general of cases. Thus, both partitions are required to update neighbor lists and exchange particles within their respective partitions at the same time step. We found using the CrayPat tool [55] that this constraint leads to significant synchronization bottlenecks. Our code optimizations to improve performance of the overall execution of simulation and analysis are detailed in §IV.

Scientific simulations decompose simulation data into 2D/3D sub-domains distributed across many MPI processes (virtual process topology). Each sub-domain consists of a portion of the input domain. For E.g., in cosmological and MD simulations, each sub-domain consists of a number of particles. In MD, each MPI rank computes key tasks for all owned particles, such as computing interactions with nearby particles and integrating equations of motion [56]. Since the sub-domains are spread across hundreds of nodes, communication consumes a non-trivial percentage of total execution time [57]. The virtual process topology directly affects execution time on a system with a certain network topology. The virtual process decomposition determines the layout of sub-domains on the physical processors. There are multiple communications in a time step between neighboring processes in MD simulations. The MPI process layout on physical process grid determines the number of network links (hops) between two neighbors on virtual process topology [53]. This affects communication times, thus process mapping is crucial to minimize the impact of inter-node communications. Placement for co-analysis is more challenging, because communication occurs among simulation processes as well as between simulation and analysis processes, and among the analysis processes. Thus, it is important to determine the optimal process mapping and decomposition.

IV. Code Optimizations in LAMMPS

We now highlight the optimizations that we developed in LAMMPS to reduce the execution time by mitigating synchronization overheads between simulation and analysis partitions. Figure 1A (left workflow) depicts the main simulation phases of LAMMPS (Verlet algorithm), the names of which are loosely based on function names in LAMMPS. The first simulation step is an initial integration method to update velocities by a half-step, and coordinates by a full step (S1) [58]. Next, the neighbor_build() method (S2), representing several functions in LAMMPS, determines whether neighbor lists are rebuilt and particles exchanged in the current time step. If requested, the processes exchange particle data, insert/delete atoms from data structures, and update neighbor lists. To ensure that simulation and analysis partitions remain synced with per-atom information, it is necessary for the simulation partition to inform the analysis partition that particle exchange was attempted. However, this synchronization is only required in steps where particle exchange actually occurs. We observed significant

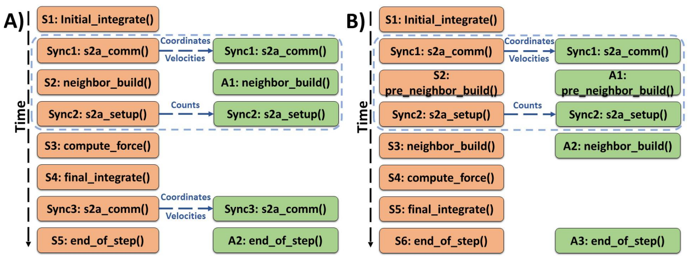


Fig. 1: A) Original and B) optimized simulation and analysis workflows. There is reduced synchronization and improved computation overlap in the optimized workflow.

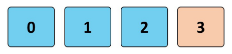
performance bottlenecks, using CrayPat, due to unnecessary forceful synchronizations in steps where particle exchanges are not attempted. We modified the algorithm to instead perform particle exchange and neighbor list rebuilds at a pre-defined frequency *neigh_sync_freq* (based on the nature of the simulation and minimum analysis frequency). The simulation processes then compute the forces using updated particle coordinates (S3), complete the second half of integration by updating velocities a second half-step (S4), and next, finalize any tasks needed at the end of a simulation step (S5).

The right workflow of Figure 1A depicts the original analysis workflow. The analysis processes require updated atom coordinates from the simulation partition after the first integration step to ensure particles are correctly exchanged between neighboring processors and to ensure partitions remain synced (steps within blue dotted line of Figure 1A). Rebuilding neighbor lists on the analysis partition is necessary as they are used by some analyses to efficiently compute properties that involve pairwise distance computation, such as radial distribution functions. It is more efficient (and simpler) to request the analysis partition to rebuild neighbor lists as opposed to communicating them between partitions. Each analysis process also updates the expected number of atoms as per their simulation process senders (Sync2) to validate that partitions are correctly synced. After the final integration step performs velocity Verlet updates in the simulation (S4), analysis processes receive the updated velocities of their atoms (Sync3). Analyses are then computed (A2) on time steps where requested; this is where most time is spent by analysis partition.

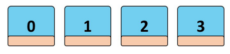
Profiling revealed that time spent communicating per-particle data between partitions is insignificant compared to rebuilding neighbor lists. Since there are fewer analysis processes, each analysis process owns a larger sub-volume of the system. Therefore, time to rebuild neighbor lists on analysis processes is longer (e.g. on BG/Q, it takes more than 3× time). Thus, this execution flow requires simulation processes to wait for the analysis processes at Sync2. Therefore, in our optimized workflow (Figure 1B), we delay the actual neighbor list build until after Sync2 (first optimization). In Verlet algorithm, velocities are updated a second half-step using current forces. The atomic coordinates were updated a full step during initial_integrate() phase. For statistical analyses that are only functions of atomic coordinates or velocities and not both, it is sufficient to simply use half-step velocities for analyses and forego the second s2a_comm() synchronization (Sync3). This second optimization allows the two partitions to make progress asynchronously; simulation partition computes simulation steps and analysis partition builds neighbor lists and computes analyses. The optimized workflow is shown in Figure 1B. The two partitions sync up again at the next specified sync-step.

V. Process Mapping

In this section, we study process mapping strategies for simulation and analysis that co-execute on allocated nodes. Typically, simulation runs on processes $0{:}M{-}1$ and analysis runs on processes $M{:}N{-}1$, where N is the total number of processes, M is number of simulation processes and the ratio r of simulation to analysis processes is $\frac{M}{N-M}$. In the multicore era, this implies that the default process mapping will place several simulation and analysis processes on different nodes (shown in Figure 2(a)). This approach has been used in prior studies of in-transit analysis [19]–[21]. However, this does not lead to optimal performance because the ratio r is usually greater than 1. This leads to congestion at analysis nodes due to simultaneous data transfers from multiple simulation nodes to analysis nodes, limited node reception bandwidth and network congestion on the global network links.Here, ranks $0{:}M{-}1$ are on nodes 0–2 and ranks $M{:}N{-}1$ are placed on node 3.



(a) Simulation (blue) and analysis (cream) on different nodes.



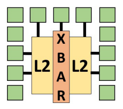
(b) Simulation (blue) and analysis (cream) on same nodes.

Fig. 2: Placement overview of simulation and analysis.

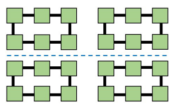
We propose process mapping heuristics to place simulation and analysis processes on the same nodes (Figure 2(b)). There are tens of cores per node in today's era, such as 68 cores/node in Cori [25]. Therefore, simulation and analysis rank placements within a node plays a crucial role. The on-chip interconnect topology-aware mapping heuristics exhibit better performance. Our proposed schemes leverage the network topology within a node (NoC) and NUMA domains in a node. First, we give a short overview of contemporary on-chip interconnects in §V-A and then explain our mapping schemes in §V-B.

A. On-chip interconnects

Multicore architectures have different types of on-chip interconnects or network-on-chip (NoC) topologies [59], [60]. Figure 3 shows a close approximation of different contemporary representative on-chip interconnects, such as a central crossbar switch (3(a)), ring interconnect (3(b)) and 2D mesh (3(c)). These three classes of NoCs can be found in 15 out of 20 top fastest supercomputers [23]. For example, the processing units in an IBM Blue Gene/Q (BGQ) chip communicate via a central crossbar switch [61]. The 16-core AMD Opteron processor also uses crossbar for intra-node communication [62].



(a) Crossbar.


(b) Ring.

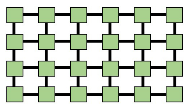

(c) 2D mesh.

Fig. 3: Examples of on-chip (intra-node) interconnects.

Another class of processors (such as the Intel processor family – Ivybridge, Haswell, Broadwell) have dual-sockets, 4–18 cores per socket and ring interconnect for on-chip communication [63], [64]. They vary in die sizes, core counts, QPI speed, DRAM bandwidth, L2 bytes/cycle etc. However, they have similar on-chip ring interconnects, which yield lower communication times than crossbar and bus-based NoCs. The cores are partitioned into two equal-sized NUMA domains [65]. These processors are also the building blocks for 7 out of top 20 fastest supercomputers, and 7 out of top 10 supercomputers in the Green500 list [66]. The latest Intel Xeon Phi processor ("Knights Landing"/KNL) has a 2D mesh interconnect connecting up to 36 tiles [28], with a YX routing order. A tile in a KNL chip comprises of 2 cores, and a shared 1 MB L2 cache. These are used in 6 out of top 20 fastest supercomputers [23]. 2D mesh interconnects have also been used in several other academic and commercial processors [26], [30], [67]–[70]. Next, we describe our mapping heuristics that account for the various types of interconnects, which affect the data transfer time from the simulation to analysis ranks.

B. NoC-aware Mapping Heuristics

Our NoC-aware mapping schemes place simulation and analysis ranks on the same node to reduce inter-node communications, which is useful for network topologies such as dragonfly that suffer from job interference [71]. Thus, they share on-chip memory, network and other resources depending on where they are placed in the node. This affects the execution time because of concurrent executions of two different codes (simulation and analysis) on the same node, which in turn affects the TLB misses, cache misses and on-chip network contention. The extent of variation in a simulation's execution time due to an analysis kernel using shared resources depends on memory hierarchy, cache sharing protocols, spatial locality, execution sequence (computation and communication phases) of the codes, and the NoC. The NoC topology determines how cache misses will be fulfilled and the path along which the data is routed from simulation to analysis processes. Furthermore, the data transfer time from simulation to analysis processes is also affected by the distance (number of hops) between simulation and analysis processes. We designed process-to-processor mapping schemes, with objectives to minimize (1) cache sharing and pollution, and (2) data transfer time from simulation to analysis processes. We considered a few different architectures with different memory hierarchies and NoCs.

1) Contiguous: The simulation and analysis processes share nodes, i.e. every node is divided into two groups of processes (Figure 4(a)). We use the physical core numbering to divide the cores. If the ratio of simulation to analysis processes is $r:1$, then the simulation processes are placed on core numbers $0 : \frac{r}{r+1}N - 1$ and the analysis processes are placed on core numbers $\frac{r}{r+1}N : N-1$. This is a generic scheme and improves performance in all multicore architectures.

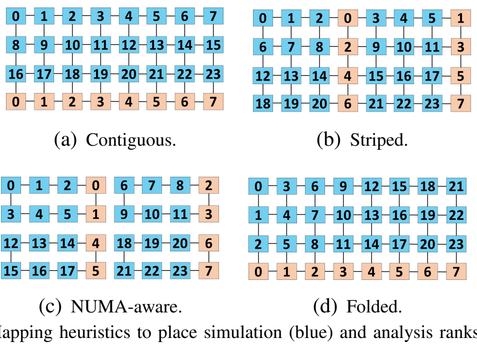


Fig. 4: Mapping heuristics to place simulation (blue) and analysis ranks (cream).

2) Striped: In this scheme, the two process groups are interleaved (Figure 4(b)). Analysis processes are placed at a regular interval, equi-distant from each other. If the ratio of simulation to analysis processes is $r:1$, then the analysis processes are placed on every r^{th} core, and simulation processes are placed on rest of the cores. This is a generic scheme, and improves performance in all multicore architectures. It is more beneficial for a chip like BG/Q, which has a central L2 cache.

3) NUMA-aware: This mapping benefits multi-socket processors with multiple NUMA domains. Analysis processes are spread on each socket so that sender and receiver ranks are placed within a socket. Figure 4(c) shows a case where cores communicate via ring interconnect in each socket. This is applicable to Intel processor series, such as Ivybridge and Haswell. We place simulation/analysis ranks such that traffic to a receiver from its senders are contained within a ring.

4) Folded: This heuristic is specific to 2D mesh interconnects on chip, such as Intel KNL [28] and TILE64 [30]. We reorder the ranks in column-major order (Figure 4(d)) because KNL has a YX routing policy. We modify the placement of simulation and analysis so that traffic from senders to receivers are contained within 1 or a few columns, depending on the ratio r. In the figure, simulation ranks in rows $1-3$ send data to the analysis rank in the last row. This minimizes cross-traffic in the links during data transfer phase due to internal routing and because each row and column of the mesh is a half ring.

The contiguous and striped mappings apply to any architecture, NUMA-aware is specifically tailored for ring interconnects, and folded is intended for chips with 2D mesh interconnects.

VI. Process Decomposition

Simulation and analysis require two process partitions, each with its own 3D process grid decomposition. Let ps_x, ps_y and ps_z refer to x, y, z process grid dimensions for simulation. Let pa_x, pa_y and pa_z refer to x, y, z process grid dimensions for analysis. Let $s = ps_x \times ps_y \times ps_z$ and $a = pa_x \times pa_y \times pa_z$ be the total number of simulation and analysis processes respectively. It is challenging to determine the 3D grid sizes because there are several constraints while determining the optimal process decompositions. First, both 3D decompositions have to be as cube-like as possible. This maximizes the number of paths between communicating tasks [48], [51] and leads to the difference between every pair of dimensions to be as small as possible. A skewed decomposition leads to unbalanced communication volume as well as unbalanced distribution in the number of hops between communicating processes while cubic decomposition minimizes the communication time for nearest-neighbor communications due to more effective mapping.

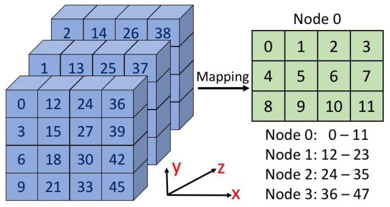


Fig. 5: Process mapping of $4 \times 4 \times 3$ grid on 4 12-core nodes.

Second, ps_x, ps_y and ps_z have to be multiples of pa_x, pa_y and pa_z respectively due to implementation constraints for a one-to-one mapping of simulation to analysis processes. Third, simulation process dimensions have to be in decreasing order so that the z dimension is the smallest. This ensures that more neighbors (in x and y dimensions) can be packed on a single node or consecutive nodes assigned to the job, since processes are numbered in zyx order by default (see Figure 5). Note, the most preferred numerical order between the simulation process dimensions depends on the order of the process layout.

MILP for optimal process decomposition

Performance is affected by optimal process decomposition. Finding the most cubic-like decompositions for s and a, while satisfying the above constraints is a combinatorial optimization problem. We formulate a mixed integer linear program (MILP) to solve this. We use the prime factorization of s and a to formulate a MILP that is quickly solvable. Let f_i be the set of prime factors of the number of simulation processes, s, and number of analysis processes, a. For example, if $s = 1536 = 2^9 * 3^1$ and $a = 512 = 2^9$, then $f = \{2, 3\}$. Let $\mathcal{I}$ be the set of index of prime factors of s and a. In the above example, $\mathcal{I} = \{1, 2\}$ since there are 2 unique prime factors. s_x, s_y, s_z and a_x, a_y, a_z are our decision variables. Then s_x_i, s_y_i and s_z_i are the number of occurrences of i^{th} factor in each of the three dimensions x, y, z for decomposing s into a 3D grid, where the sum $s_x + s_y + s_z$ has to match the required multiplicity $\{9, 1\}$ for each factor. Similarly, a_x_i, a_y_i and a_z_i are the number of occurrences of i^{th} factor in each of the three dimensions x, y, z for decomposing a into a 3D grid. Since ps_x has to be a multiple of pa_x, therefore each element in s_x should be greater than or equal to the corresponding element in a_x, and the same holds true for y and z dimensions. For example, in the above case, one possible solution is $s\_x = \{4, 0\}$, $s\_y = \{2, 1\}$ and $s\_z = \{3, 0\}$, and $a\_x = \{4, 0\}$, $a\_y = \{2, 0\}$ and $a\_z = \{3, 0\}$. The 3D grid can then be computed from these decision variables; in this example $s = 16 \times 12 \times 8$ and $a = 16 \times 4 \times 8$.

Note that $ps_x = \prod_{i \in \mathcal{I}} {f_i}^{s\_x_i}$, $ps_y = \prod_{i \in \mathcal{I}} {f_i}^{s\_y_i}$ and $ps_z = \prod_{i \in \mathcal{I}} {f_i}^{s\_z_i}$. Similarly pa_x, pa_y and pa_z can be computed. It is preferable that simulation dimensions are ordered in decreasing dimension size, leading to the constraints $ps_x \geq ps_y \geq ps_z$. We do not enforce a strict ordering on the dimensions of a. Our objective is to minimize the difference between each pair of dimensions for both s and a, i.e. to obtain cubic decompositions for both. For simulation, a cube-like ordering is obtained by minimizing the mean of the ratios $(1, ps_x/ps_y, ps_x/ps_z)$ due to ordering of the dimensions. For analysis, we determine pa_{max}, the maximum of pa_x, pa_y and pa_z, by adding constraints. The objective function for analysis is then to minimize the mean of the ratios $(pa_{max}/pa_x, pa_{max}/pa_y, pa_{max}/pa_z)$. We use geometric mean of all six ratios resulting in an objective of the form $1 * ps_x/ps_y * ps_x/ps_z * pa_{max}/pa_x * pa_{max}/pa_y * pa_{max}/pa_z$. We then use logarithms to simplify the objective function to Equation 1, where a_max_i is the maximum of a_x_i, a_y_i, a_z_i for the i^{th} prime factor and w is the weight. We use weighted mean with w=2 to favor a cubic decomposition of the simulation processes s. The simplified version using geometric mean and logarithm reformulation is solvable quickly.

$$minimize \sum_{i \in \mathcal{I}} log(f_i) * (w * (2 * s\_x_i - s\_y_i - s\_z_i) + 3 * a\_max_i - a\_x_i - a\_y_i - a\_z_i) \quad (1)$$

$$s\_x_i + s\_y_i + s\_z_i = s\_prod_i \; \forall i \in \mathcal{I} \quad (2)$$

$$a\_x_i + a\_y_i + a\_z_i = a\_prod_i \; \forall i \in \mathcal{I} \quad (3)$$

Next, we describe the constraints. Equations 2 and 3 specify correctness of total number of simulation and analysis processes. s_prod_i and a_prod_i are the number of occurrences of i^{th} prime factor (from $\mathcal{I}$) in s and a respectively. For our example, where s=1536, and a=512, s_prod=$\{9, 1\}$ and a_prod=$\{9, 0\}$. Thus, s=$\prod_{i \in \mathcal{I}} {f_i}^{s\_prod_i}$ and a=$\prod_{i \in \mathcal{I}} {f_i}^{a\_prod_i}$. Thus, 1536=16 $\times$ 12 $\times$ 8=$2^4 \times (2^2 \times 3) \times 2^3$=$2^9 * 3^1$, where 4+2+3=9. Equations 4–5 specify ordering constraints on the grid dimensions for the simulation processes and equations 6–8 compute the maximum dimensions for the analysis processes.

$$\sum_{i \in \mathcal{I}} log\,(f_i) \cdot s\_y_i \leq \sum_{i \in \mathcal{I}} log\,(f_i) \cdot s\_x_i \quad (4)$$

$$\sum_{i \in \mathcal{I}} log\,(f_i) \cdot s\_z_i \leq \sum_{i \in \mathcal{I}} log\,(f_i) \cdot s\_y_i \quad (5)$$

$$\sum_{i \in \mathcal{I}} log\,(f_i) \cdot a\_x_i \leq \sum_{i \in \mathcal{I}} log\,(f_i) \cdot a\_max_i \quad (6)$$

$$\sum_{i \in \mathcal{I}} log\,(f_i) \cdot a\_y_i \leq \sum_{i \in \mathcal{I}} log\,(f_i) \cdot a\_max_i \quad (7)$$

$$\sum_{i \in \mathcal{I}} log\,(f_i) \cdot a\_z_i \leq \sum_{i \in \mathcal{I}} log\,(f_i) \cdot a\_max_i \quad (8)$$

Each dimension in the 3D grid for simulation and analysis process decomposition is greater than one for the cases considered here. Therefore there should be at least 1 factor from f in each dimension, as specified in Equations 9–14.

$$\sum_{i \in \mathcal{I}} s\_x_i \geq 1 \quad (9) \qquad \sum_{i \in \mathcal{I}} a\_x_i \geq 1 \quad (12) \qquad a\_x_i \leq s\_x_i \; \forall i \in \mathcal{I} \quad (15)$$

$$\sum_{i \in \mathcal{I}} s\_y_i \geq 1 \quad (10) \qquad \sum_{i \in \mathcal{I}} a\_y_i \geq 1 \quad (13) \qquad a\_y_i \leq s\_y_i \; \forall i \in \mathcal{I} \quad (16)$$

$$\sum_{i \in \mathcal{I}} s\_z_i \geq 1 \quad (11) \qquad \sum_{i \in \mathcal{I}} a\_z_i \geq 1 \quad (14) \qquad a\_z_i \leq s\_z_i \; \forall i \in \mathcal{I} \quad (17)$$

Equations 15–17 ensure integer divisibility of simulation and analysis process dimensions. Equations 18–23 specify bound constraints for each factor from f in each dimension. Note that since $s \geq a$, the number of prime factors for s may be more than a. Therefore, a factor may be present in s and may not be present in a. Hence one or more terms may be 0 (as shown in our example of 1536 and 512).

$$s\_x_i \geq 0 \; \forall i \in \mathcal{I} \quad (18) \qquad a\_x_i \geq 0 \; \forall i \in \mathcal{I} \quad (21)$$

$$s\_y_i \geq 0 \; \forall i \in \mathcal{I} \quad (19) \qquad a\_y_i \geq 0 \; \forall i \in \mathcal{I} \quad (22)$$

$$s\_z_i \geq 0 \; \forall i \in \mathcal{I} \quad (20) \qquad a\_z_i \geq 0 \; \forall i \in \mathcal{I} \quad (23)$$

VII. Experiments

We describe the simulation and analyses, the three systems used for evaluation and the various experiments performed to understand the efficacy of our approaches.

Application: We used the classical molecular dynamics simulation code – Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [56] for most experiments. Lennard-Jones model was used with a cutoff to compute pairwise interactions and PPPM for long-range electrostatics [72]. We evaluate in situ analysis performance using a problem that spans a range of conditions explored in molecular simulations of liquids, materials and biological systems. We simulated a box of water molecules solvating two types of ions. We consider the following representative analyses: two radial distribution functions averaged over all molecules, three velocity auto-correlation functions and mean squared displacements averaged over each molecule and ion. We also used an MPI collective I/O benchmark [73] for evaluation of our NoC-aware mappings.

Systems: We performed experiments on three supercomputers with diverse NoCs (refer §V-A): (1) The IBM Blue Gene/Q *Mira* [74] system has 48K 16-core nodes interconnected by 5D torus, each core has 16 KB L1 cachesand shared 32 MB L2 cache per node (2) Intel Haswell partition of *Cori* [25] has 2388 32-core Intel Xeon nodes interconnected by Dragonfly, each core has 32 KB L1 caches, 256 KB L2 cache, and shared 40 MB L3 cache per socket and (3) Intel KNL *Theta* [24] has 4392 64-core nodes interconnected by Dragonfly, each core has 32 KB L1 caches, two cores share 1 MB L2 cache.

Results: We used our MILP solution to decide the optimal 3D process grid for LAMMPS simulation and analysis. This was specified in LAMMPS input configuration file via processors command. The MILP was modeled in AMPL [75] and solved using MINLP solver. Solve times were less than 0.08s.

The custom mappings on Mira were specified using the `RUNJOB_MAPPING` environment variable. On Cori and Theta, we used `MPICH_RANK_REORDER_METHOD`, a Cray environment variable for process-mapping. In this case, when it's set to 0/1/2(0=default), system-defined(Cray) placement schemes are used. To use our custom mapfile, we set `MPICH_RANK_REORDER_METHOD=3` and use `MPICH_RANK_ORDER` as mapfile. Examples can be found in [76]. Custom mappings alter the locations of simulation ranks (0:M-1) and analysis ranks (M:N-1) in the allocated job partition. We also compared to other system mappings (e.g. `MPICH_RANK_REORDER_METHOD`=0/1/2 on Theta and Cori [77]). However, those did not perform well and are not presented here. On Theta, the `flat quad` mode of execution was used; it performed slightly better than other memory-cluster modes and the results are representative of other modes. Next, we show results for scalability, effect of our code optimizations, process decomposition and mappings. LAMMPS was run for 1000 time steps, with analysis frequency of 50. The ratio of the simulation to analysis processes was 7:1 on Theta, as it has a larger number (64) of cores per node. We used a ratio of 3:1 on Cori (Haswell) and Mira, which have 32 and 16 cores per node respectively. These ratios have been empirically determined (Please refer to Appendix for the experiments to determine these ratios). Performance modeling can also be used [37]. We profiled a few configurations on Cori and Mira for core utilization and synchronization between ranks. The variation in instructions per cycle and number of average and maximum cycles across all cores was less than 1%. We noted a variation of 90% between synchronization time in simulation and analysis ranks in the case of default mapping, whereas for our NUMA-aware mapping, the variation across all cores was less than 2% on Cori. The `neigh_sync_freq` was set to 50 (see §IV). 4 threads were used on BG/Q and KNL, and 2 threads were used on Haswell. Comparison to time-shared mode and overhead of co-analysis is shown in §VII-4.

1) Effect of Code Optimization: Figure 6 shows the effect of our two optimizations to the original LAMMPS co-analysis workflow (see §IV). The simulation and analysis execution times of 12, 34 and 51 million atoms on 2048 Mira cores are shown. For all data points, we achieved ∼3% improvement with reduced synchronization and additional ∼4% by better overlap of computations between simulation and analysis partitions (consistent over 5 runs). Cori speedup was ∼7% with reduced synchronization and further 5% with computation overlap for 34M atoms. Speedup on Theta was ∼7% with reduced synchronization and further 12% with computation overlap for 6M atoms. This is because the simulation and analysis can progress faster without unnecessarily syncing up too frequently. The impact on analysis results with and without these optimizations are statistically insignificant; typical differences occur after the 6^{th} decimal place. Typically, scientists simulate millions of time steps for complex problems and these code optimizations could lead to significant savings in time-to-solution.

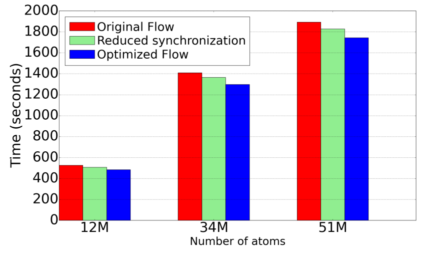


Fig. 6: Execution times of simulation and analysis on 2048 cores of Mira.

2) Process Decomposition: Table I shows the utility of our MILP to determine the optimal 3D process grids for simulation and analysis (refer §VI). Column 2 shows 1.5 million atom simulation times on 256 nodes (8 ranks/node) on BG/Q with various 3D process decompositions. Column 1 shows simulation (s) and analysis (a) process grids with a ratio of 3:1. Row 1 shows the result from MILP. We compare this with other decompositions (selected at random) in rows 2–4. The arithmetic mean of the ratios between the x,y,z dimensions of the simulation grid are 1.61, 3.83, 11.67, 21.33 for rows 1, 2, 3, 4, respectively. Note that, larger the difference between dimensions, longer is the execution time. The MILP solution gives up to a 22% lower execution times than the random decompositions considered. This is because we specify the constraints of minimizing the difference between x, y, z dimensions and ordering the dimensions in the formulation. Out of the many different decompositions satisfying the constraints mentioned in §VI, MILP outputs the best configuration.

Columns 3 – 5 show communication times for the most time consuming MPI calls in LAMMPS on BG/Q. The times were collected using IBM HPCT profiler [78]. Each row depicts the average of maximum MPI time across all processes from 4 independent runs. The maximum MPI time across all processes represents the time taken by the slowest process, which determines the overall execution times. Due to optimal process decomposition, row 1 has lowest communication times.

We ran similar configurations on 2048 cores of Theta. The

TABLE I: Effect of 3D process decompositions on runtimes.

Process grids	Wallclock (s)	Send (s)	Alltoallv (s)	Wait (s)
16 × 12 × 8 (s) 16 × 4 × 8 (a)	107.63	9.17	8.67	5.91
24 × 16 × 4 (s) 8 × 16 × 4 (a)	116.24	16.36	11.82	11.01
16 × 48 × 2 (s) 16 × 16 × 2 (a)	124.51	16.33	13.62	9.89
2 × 96 × 8 (s) 2 × 32 × 8 (a)	138.02	33.80	14.85	25.16

number of network request and response stalls (from CrayPat [55]) were 52% and 33% lower (averaged from 5 runs) for the optimal decomposition. On Cori, the MILP solution resulted in 12% lower runtimes and incurred 23% lower L3 caches misses than other 3D decompositions. This shows the importance of MILP to quickly identify the best 3D decomposition as opposed to semi-randomly sampling multiple decompositions in hope of choosing that which yields the best runtime.

3) Strong Scaling: Figure 7 compares the default placement of ranks to contiguous, striped and NUMA-aware mappings on 768 – 4096 Haswell processors of Cori. The x-axis shows the number of ranks and the y-axis shows the boxplot of corresponding total execution time of simulation and analysis of 21 million atoms using LAMMPS. The analysis runs on one-fourth of the total number of processes in all cases. Each group of boxplots in the x-axis compares the 4 mappings. Our mappings always result in lower execution times for all 4 node counts because we avoid inter-node data transfers. The contiguous and striped rank placements are 5% and 14% (mean values) better than the default rank ordering (shown in red). In the case of contiguous mapping, though the analysis ranks are placed on the same node, they are all placed within the same socket. In the case of striped, the simulation and analysis rank ordering is interleaved. However, both mappings do not consider the two ring interconnects in each Haswell socket.

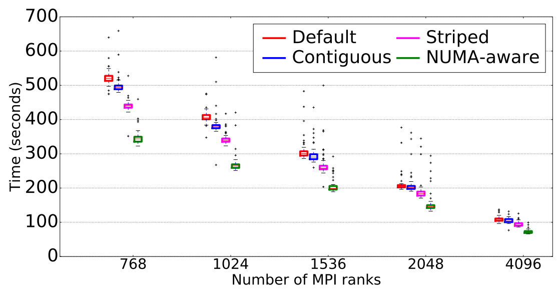


Fig. 7: Execution times of 21 million atom 1000-step LAMMPS simulation and analysis (3:1 ratio) on 768 – 4096 ranks (24, 32, 48, 64, 128 nodes) of Cori Haswell.

The NUMA-aware heuristic (shown in green) outperforms the other mappings in all cases. On average, the execution time is reduced by 31% on all node counts compared to the default rank placement. This approach tries to ensure that the data transfer from each block of senders to each receiver (analysis rank) does not interfere with the other concurrent data transfers in the same socket/node. Clearly, mapping the ranks in a way to avoid intra-node NoC congestion by leveraging the ring interconnects in each socket (see Figure 4(c)) helps in achieving the best performance. The consumed energy (from the Slurm job scheduler accounting data on Cori) for the default, contiguous, striped, NUMA-aware mappings on 128 nodes were 5.74M, 4.66M, 5.10M, 3.87M Joules respectively. Thus, NUMA-aware mapping is also the most energy-efficient due to reduced data movement in NoC. These mappings are also expected to benefit other coupled codes, such as the verlet/split parallelization available in LAMMPS, whereby long-range electrostatic interactions (involving 3D FFTs and MPI_Alltoallv) are parallely computed with short-range interactions on separate processor partitions [33]. We noted 33% improvement for 21 million atom verlet/split simulation on 768 ranks of Cori with NUMA-aware mapping. This demonstrates the general applicability of our mappings.

We now summarize the strong scaling results from Theta. The NoC-aware mappings reduced the execution time of 220 million atom simulation and co-analysis on 64, 128, 256, 512 nodes (with 64 ranks per node) by 132, 58, 39 and 10 seconds on average compared to the default mapping. As data size per core decreases, the data transfer time overhead reduces too, and thus absolute improvements are lesser at higher core counts. However, for long running simulations and/or with larger problem sizes, these improvements lead to substantially reduced runtimes ranging from a few hours to days. The folded mapping performs slightly better than contiguous and striped. Unlike on Haswell, the architecture-specific mapping on KNL (folded) gives little benefit. This is because the Intel KNL NoC is not as precisely known as in the Intel Haswell architecture. The proprietary die layout of the 64 cores on the 2D mesh of a Theta node is only approximately known.

TABLE II: Profiles of 1000-step simulation and analysis using default and striped mappings on 16384 ranks of Mira.

#Atoms	Wallclock (s)	L1p miss	L2 miss	MPI_Send (s)
0.8M	78.4 62.8	4.2×10^8 3.6×10^8	5.4×10^7 4.2×10^7	3.8 3.5
2M	84.4 71.4	4.9×10^8 4.4×10^8	6.9×10^7 5.4×10^7	5.2 5.0
2.7M	90.5 83.0	5.7×10^8 5.4×10^8	9.5×10^7 7.2×10^7	6.7 6.3

4) Data Scaling: Table II shows the profiles of LAMMPS simulation and analysis (1000 time steps) on 1024 nodes (16 ranks per node, 4 threads per rank) of Mira for different system sizes (column 1). In each row (columns 2 – 5), the data for default and striped mappings are shown in the first and second sub-rows, respectively. The total execution time for striped mapping is lower than the default in all cases. The reason for this can be explained from columns 3 – 5. The cache misses (both L1 prefetch and L2) are higher for the default mapping. MPI_Send is used in both simulation and analysis code paths and is one of the most time consuming and called functions. The maximum time for an MPI_Send (i.e. the slowest communication time) is also higher in the default case (column 5). This is partly because the analysis ranks are a few hops away from their senders (simulation ranks) in case of default mapping. Overall, the striped mapping results in up to 20% lower execution times than the default mapping. The execution times of 51, 73, 100 million-atom simulation and analysis on 4096 cores of Theta are shown in rows 1–3 of Table III. The maximum time is taken by the time-shared analysis mode (column 1) because the simulation and analysis stall each other to execute; this also causes more cache misses. The contiguous, striped, and folded mappings perform better than the default mapped space-shared mode (column 2) because of more efficient data transfers within the node. The folded mapping (column 5), which is particularly suited for 2D mesh interconnects, performs the best and is within 10% of simulation execution times (column 6). The time-shared mode takes $\sim$ 28% longer than simulation and $\sim$ 20% more than folded mapping.

TABLE III: Execution times for 51M, 73M, 100M-atom simulation+analysis.

Time-shared	Default	Contiguous	Striped	Folded	Simulation
624.83	512.60	488.39	489.35	486.93	434.33
868.94	726.04	698.09	696.53	695.69	621.46
1171.28	992.75	944.22	944.97	942.86	850.45

The number of network request stalls (from CrayPat) for simulation and analysis of 6 and 34 million atoms are 2.4e+06 and 9.8e+06 respectively for default mapping, and 5e+05 and 7e+05 for folded mapping, on 1024 cores of Theta. This is due to more congestion at the analysis nodes in case of default mapping and more impact of network congestion for larger data sizes. Clearly, with increasing data size, improved process placement has a substantial impact. The L3 cache misses with NUMA-aware mapping for 12M, 21M, 34M and 51M-atom simulations on Cori were 4–12% lower than default mapping.

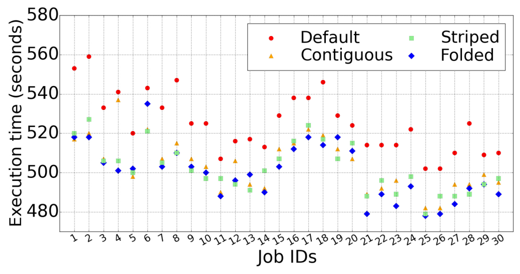


Fig. 8: Execution times using different mappings for different job runs. Default mapping (red) performs the worst and folded (blue) gives the lowest times in most cases.

5) Effect of Runtime Variation: The impact of job placement and inter-job interference has been well studied [79]–[81]. We analyzed 30 job logs to study the impact of runtime variability on all mappings. Figure 8 shows the execution times of LAMMPS simulation and analysis of 133 million atoms on 16384 cores of Theta using the various mappings for the 30 jobs examined. We don't include Mira results because there wasn't much variation. For each job, four subsequent runs with different mappings were executed within the same job script to ensure identical node placements. The runtimes were observed to vary for all mappings across the different jobs to similar degrees. The variation may be due to differences in allocated partitions as well as other job interference. Thus, in addition to statistics, comparisons on a per-job basis helps understand the relative performance improvements. For all job runs, the default mapping resulted in the highest execution times and folded mapping had the lowest times for most cases - thus, depicting the importance of node-aware mapping.

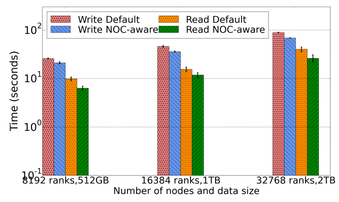
6) Network Analysis: One of the main bottlenecks in parallel computing at large-scale is the extensive usage of network. Given that this resource is commonly shared on large-scale systems today, sharing this resource with many other jobs can potentially lead to congestion and application slowdown. Our proposed mappings mitigate to some degree the effects of slowdown due to the default mapping of simulation and analysis ranks, by placing the analysis ranks closer to their senders. In this section, we analyze some of the available Aries network counters [82], [83] using the CrayPat tool (v7.0.0) [55] on Theta. We focus on the Theta system as the congestion results were only available here. We show a few representative network counters (column 1) in Table IV for simulation and analysis of 100 million atoms on 128 nodes (64 ranks per node), collected from 5 job runs. Columns 2–4 show the corresponding counter values (mean) for default, contiguous and folded mappings, respectively. The first row indicates the aggregate request traffic injected by the NIC and the second row indicates the aggregate response traffic received by the NIC from the Aries network, which interconnects the compute nodes on Theta. The third and fourth rows indicate the back-pressure arising at the sender and receiver nodes. The default mapping leads to 20% and 23% more packets injected/received at the NIC than contiguous and folded mappings, respectively. This is due to more off-node data transfers to analysis ranks in the case of default. The reduction in request stalls for contiguous and folded were 68% and 71% and the reduction in response stalls were 27% and 34% respectively, as compared to the default mapping. This indicates lesser back-pressure at the nodes for our mappings. A high ratio of stalls to packets indicate possible network congestion [83]. Clearly, NoC-aware mappings lead to lower congestion benefiting both our job and other running jobs.

TABLE IV: Network counters for LAMMPS simulation and analysis of 100 million atoms on 8192 cores of Theta, using default and NoC-aware mappings.

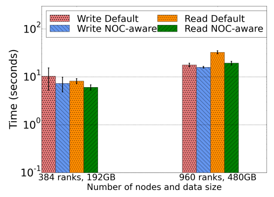
Counters	Default ($\times 10^7$)	Contiguous ($\times 10^7$)	Folded ($\times 10^7$)
Request packets	12.15	9.62	9.35
Response packets	12.15	9.62	9.34
Request stalls	77.57	24.67	22.11
Response stalls	26.90	19.44	17.67

7) Effect on MPI Collective I/O: We show the utility of our mappings on two-phase MPI collective I/O performance [84] using our benchmark [73]. Figure 9(a) depicts the mean read and write times for the default and NoC-aware placement of aggregator ranks, with standard deviation shown as error bars (from 5 runs). This was run on 128, 256 and 512 nodes (64 ranks/node) of Theta, which has a Lustre file system [85], [86]. This experiment reads and writes contiguous data from each rank, using `MPI_File_[write/read]_at_all`. The largest file size written was 2 TB from 512 nodes. Maximum number of object storage servers (OSS) and 16 MB stripe size were used to yield the best performance in all cases. In these runs, compute node to aggregator node ratio was set to 16:1 using the Cray MPI environment variable `cb_nodes`. Lower ratios led to poor I/O bandwidths in both cases. Since the effective I/O bandwidth has some deviation, we ran both approaches using the same partition and OSS allocation as part of the same job for fair comparison. Cray MPI v7.7.0 was used.

Our NoC-aware mapping on Theta was able to reduce the mean write times on 128, 256 and 512 nodes by 17%, 21% and 24%, respectively. The corresponding reduction in mean read times were 35%, 24% and 35%, respectively. Our mapping places the aggregator ranks on the same nodes as the MPI ranks from which they send/receive data. On the KNL we used the striped mapping such that the distance of each aggregator rank from the corresponding compute ranks is minimized on the KNL 2D mesh interconnect. In the default case, the aggregator ranks are chosen in round robin fashion, which often leads to multi-hop communication for aggregation. Aggregators help in improving performance of large-scale I/O [87]. However, it is extremely important to carefully place them such that the data exchange time of the two-phase collective I/O is reduced. We used the Cray timers (`MPICH_MPIIO_TIMERS`) [88] to measure the split I/O times, such as aggregation and wait times. The maximum wait times for writes on 16384 ranks (second data point in Figure 9(a)) using default and our approach were 11.41×10^9 and 3.47×10^9 clock ticks respectively, the corresponding averages were 6.67×10^9 and 2.72×10^9 clock ticks respectively. Our mapping leads to reduction in wait times by a factor of 3, thus reducing the overall write/read times.



(a) Write/read times of 512 GB, 1 TB, 2 TB shared files on Intel KNL (weak scaling).



(b) Write/read times of 192 and 480 GB shared files on Intel Haswell.

Fig. 9: I/O improvements with our mappings on Intel KNL and Intel Haswell.

I/O results on Mira are not presented here[1]. We ran the same benchmark on Cori Haswell partition on 12 and 30 nodes (32 ranks/node). Figure 9(b) shows the write and read times for 192 GB and 480 GB shared files using the default and our NoC-aware approach. The plots show the mean and standard deviation from 20 runs. We used the NUMA-aware mapping heuristic (§V-B3) to place the aggregators, i.e. we consider the two sockets per node and ring interconnect to place the aggregators. We achieved 18% and 11% reduction in write times on 384 and 960 ranks, respectively. The corresponding reduction in read times were 26% and 42%. We observed a reduction of 42% and 60% in the average and maximum wait times (from Cray MPI timers) for writing 480 GB file from 960 ranks. Thus, rank placement based on the NoC topology is important to reduce data movement times in collective I/O.

[1](1) Mira has MPIv2, which does not allow fine-grained control of aggregator placements and (2) BG/Q has 16 cores/node and default/best ratio of aggregator:compute is 1:16, therefore placing the aggregator on any of the 16 cores would give similar results due to crossbar interconnect.

VIII. Conclusions and Future Work

We demonstrated effective techniques to scale space-shared co-analysis of large-scale molecular dynamics simulations. We presented node interconnect topology-aware mappings for improved placement of simulation and analysis ranks to reduce congestion and improve data movement. Space-shared co-analysis outperforms time-shared mode of analysis (e.g. 20% improvement on 4096 Theta cores) as it is able to concurrently execute simulation and analysis on sub-partitions of allocated nodes. Our topology-aware placement of analysis ranks (contiguous, striped, folded and NUMA-aware) improves performance in comparison to the default mapping due to reduced data transfer times. The folded mapping performs better than others on Theta (Intel KNL), which has a 2D mesh interconnect. On Intel Haswell (Cori), the NUMA-aware mapping, which fully exploits the ring interconnect, reduces the execution times by 30%. This also yields a more energy-efficient mapping (32% less than the default mapping) due to reduced data movement in the NoC. On BG/Q (Mira), which uses a crossbar chip interconnect, the striped mapping achieves 10–20% improvement in comparison to the default mapping.

We demonstrated the efficacy of our placement schemes for an I/O benchmark. For MPI collective I/O, we reduce read and write times by 20–30% on Cori and Theta by optimizing the placement of aggregators using our approaches. We achieved 33% improvement in another application – coupled execution of long and short-range forces using NUMA-aware mapping on Cori. Our mixed integer linear program formulation identified optimal 3D decompositions for both the simulation and analysis. This resulted in 22% lower execution times for the LAMMPS co-analysis workflow on all three supercomputing systems. Additionally, we improved the performance of this workflow by significantly improving the overlap between both executions and reducing the inherent synchronizations between them. This achieved a 7% reduction in execution times. This implies an improvement in the time-to-solution and computational cost for long-running MD simulations – common workloads on large scale supercomputing systems. With these optimizations, we are able to perform several in situ analyses at a desirable high analysis frequency at scale yielding greater insights.

In future, we will extend this to other applications such as cosmology [20] and to in situ analysis modes such as those where simulation and analysis are executed as separate jobs.

Acknowledgments

This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research under award numbers 57L32, 57L11, 57K50 and 5080500 and by the Exascale Computing Project (17-SC-20-SC). This work was funded in part and used resources of the Argonne Leadership Computing Facility, a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

References

[1] L. Grinberg, V. Morozov, D. Fedosov, J. Insley, M. Papka, K. Kumaran, and G. Karniadakis, "A new computational paradigm in multiscale simulations: Application to brain blood flow," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2011.

[2] S. Habib, V. Morozov, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, T. Peterka, J. Insley, D. Daniel, P. Fasel, N. Frontiere, and Z. Lukic, "The Universe at Extreme Scale: Multi-petaflop Sky Simulation on the BG/Q," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012.

[3] P. Johnsen, M. Straka, M. Shapiro, A. Norton, and T. Galarneau, "Petascale WRF Simulation of Hurricane Sandy Deployment of NCSA's Cray XE6 Blue Waters," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013.

[4] T. Ichimura, K. Fujita, S. Tanaka, M. Hori, M. Lalith, Y. Shizawa, and H. Kobayashi, "Physics-based Urban Earthquake Simulation Enhanced by 10.7 BlnDOF $\times$ 30 K Time-step Unstructured FE Non-linear Seismic Wave Simulation," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014.

[5] A. Randles, E. W. Draeger, T. Oppelstrup, L. Krauss, and J. A. Gunnels, "Massively Parallel Models of the Human Circulatory System," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15, 2015.

[6] Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee, "Top Ten Exascale Research Challenges," US Department of Energy, Office of Science, Tech. Rep., 2014.

[7] J. Jeffers, J. Reinders, and A. Sodani, *Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition*. Elsevier Science, 2016.

[8] F. Chen, M. Flatken, A. Basermann, A. Gerndt, J. Hetheringthon, T. Krger, G. Matura, and R. W. Nash, "Enabling In Situ Pre- and Post-processing for Exascale Hemodynamic Simulations - A Co-design Study with the Sparse Geometry Lattice-Boltzmann Code HemeLB," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, Nov 2012, pp. 662–668.

[9] P. OLeary, J. Ahrens, S. Jourdain, S. Wittenburg, D. H. Rogers, and M. Petersen, "Cinema image-based in situ analysis and visualization of MPAS-ocean simulations," *Parallel Computing*, vol. 55, pp. 43–48, 2016, visualization and Data Analytics for Scientific Discovery.

[10] A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock, and E. W. Bethel, "In Situ Methods, Infrastructures, and Applications on High Performance Computing Platforms," *Computer Graphics Forum*, vol. 35, no. 3, pp. 577–597, 2016.

[11] C. Seshadhri, A. Pinar, D. Thompson, and J. Bennett, "Sublinear Algorithms for Extreme-Scale Data Analysis," in *Topological and Statistical Methods for Complex Data*, ser. Mathematics and Visualization, J. Bennett, F. Vivodtzev, and V. Pascucci, Eds. Springer Berlin Heidelberg, 2015, pp. 39–54.

[12] J. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, P. Pebay, D. Thompson, H. Yu, F. Zhang, and J. Chen, "Combining in-situ and in-transit processing to enable extreme-scale scientific analysis," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012.

[13] Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee, "Synergistic Challenges in Data-Intensive Science and Exascale Computing," US Department of Energy, Office of Science, Tech. Rep., 2013.

[14] X. Ma, M. Winslett, J. Lee, and S. Yu, "Improving MPI-IO output performance with active buffering plus threads," in *Proceedings International Parallel and Distributed Processing Symposium*, April 2003.

[15] K. L. Ma, "In Situ Visualization at Extreme Scale: Challenges and Opportunities," *IEEE Computer Graphics and Applications*, vol. 29, no. 6, pp. 14–19, Nov 2009.

[16] J. Kress, S. Klasky, N. Podhorszki, J. Choi, H. Childs, and D. Pugmire, "Loosely Coupled In Situ Visualization: A Perspective on Why It's Here to Stay," in *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, ser. ISAV2015, 2015.

[17] Y. Wang, G. Agrawal, T. Bicer, and W. Jiang, "Smart: A MapReduce-like Framework for In-situ Scientific Analytics," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15, 2015.

[18] M. Dorier, R. Sisneros, T. Peterka, G. Antoniu, and D. Semeraro, "Damaris/Viz: A nonintrusive, adaptable and user-friendly in situ visualization framework," in *2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)*, Oct 2013.

[19] P. Malakar, V. Vishwanath, C. Knight, T. Munson, and M. E. Papka, "Optimal execution of co-analysis for large-scale molecular dynamics simulations," in *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016, pp. 702–715.

[20] B. Friesen, A. Almgren, Z. Lukić, G. Weber, D. Morozov, V. Beckner, and M. Day, "In situ and in-transit analysis of cosmological simulations," *Computational Astrophysics and Cosmology*, vol. 3, no. 1, 2016.

[21] C. Sewell, K. Heitmann, H. Finkel, G. Zagaris, S. T. Parete-Koon, P. K. Fasel, A. Pope, N. Frontiere, L.-t. Lo, B. Messer, S. Habib, and J. Ahrens, "Large-scale Compute-intensive Analysis via a Combined In-situ and Co-scheduling Workflow Approach," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15, 2015.

[22] M. Dreher and B. Raffin, "A Flexible Framework for Asynchronous in Situ and in Transit Analytics for Scientific Simulations," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, May 2014, pp. 277–286.

[23] "Top 500 Supercomputing Sites," http://www.top500.org.

[24] "Argonne Leadership Computing Facility's Supercomputer Theta," http://www.alcf.anl.gov/theta.

[25] "NERSC, Lawrence Berkeley National Laboratory's Supercomputer Cori," http://www.nersc.gov/users/computational-systems/cori.

[26] A. Varghese, B. Edwards, G. Mitra, and A. P. Rendell, "Programming the Adapteva Epiphany 64-core network-on-chip coprocessor," *The International Journal of High Performance Computing Applications*, vol. 31, no. 4, pp. 285–302, 2017.

[27] M. Li, S. S. Vazhkudai, A. R. Butt, F. Meng, X. Ma, Y. Kim, C. Engelmann, and G. Shipman, "Functional Partitioning to Optimize End-to-End Performance on Many-core Architectures," in *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2010, pp. 1–12.

[28] A. Sodani, R. Gramunt, J. Corbal, H. S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. C. Liu, "Knights Landing: Second-Generation Intel Xeon Phi Product," *IEEE Micro*, vol. 36, no. 2, pp. 34–46, 2016.

[29] G. Aupy, A. Benoit, B. Goglin, L. Pottier, and Y. Robert, "Co-scheduling HPC workloads on cache-partitioned CMP platforms," Inria, Research Report RR-9154, Feb 2018.

[30] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C. C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook, "TILE64 - Processor: A 64-Core SoC with Mesh Interconnect," in *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, Feb 2008, pp. 88–598.

[31] I. E. Papazian, S. Kottapalli, J. Baxter, J. Chamberlain, G. Vedaraman, and B. Morris, "Ivy Bridge Server: A Converged Design," *IEEE Micro*, vol. 35, no. 2, pp. 16–25, 2015.

[32] R. Graham, E. Lawler, J. Lenstra, and A. Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey," in *Discrete Optimization II*, ser. Annals of Discrete Mathematics, P. Hammer, E. Johnson, and B. Korte, Eds. Elsevier, 1979, vol. 5, pp. 287 – 326.

[33] Y. Peng, C. Knight, P. Blood, L. Crosby, and G. A. Voth, "Extending Parallel Scalability of LAMMPS and Multiscale Reactive Molecular Simulations," in *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the Campus and Beyond*, ser. XSEDE '12, 2012, pp. 37:1–37:7.

[34] J. W. Hurrell, M. M. Holland, P. R. Gent, S. Ghan, J. E. Kay, P. J. Kushner, J.-F. Lamarque, W. G. Large, D. Lawrence, K. Lindsay, W. H. Lipscomb, M. C. Long, N. Mahowald, D. R. Marsh, R. B. Neale, P. Rasch, S. Vavrus, M. Vertenstein, D. Bader, W. D. Collins, J. J. Hack, J. Kiehl, and S. Marshall, "The Community Earth System Model: A Framework for Collaborative Research," *Bulletin of the American Meteorological Society*, vol. 94, no. 9, pp. 1339–1360, September 2013.

[35] B. Zhang, T. Estrada, P. Cicotti, and M. Taufer, "Enabling In-Situ Data Analysis for Large Protein-Folding Trajectory Datasets," in *Proceedings*

of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium, ser. IPDPS '14, 2014, pp. 221–230.

[36] T. Peterka, J. Kwan, A. Pope, H. Finkel, K. Heitmann, S. Habib, J. Wang, and G. Zagaris, "Meshing the Universe: Integrating Analysis in Cosmological Simulations," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, Nov 2012, pp. 186–195.

[37] P. Malakar, V. Vishwanath, T. Munson, C. Knight, M. Hereld, S. Leyffer, and M. E. Papka, "Optimal scheduling of in-situ analysis for large-scale scientific simulations," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2015.

[38] J. Ahrens, S. Jourdain, P. OLeary, J. Patchett, D. H. Rogers, and M. Petersen, "An Image-Based Approach to Extreme Scale in Situ Visualization and Analysis," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2014, pp. 424–434.

[39] C. Docan, M. Parashar, and S. Klasky, "DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10, 2010, pp. 25–36.

[40] Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J. Y. Choi, S. Klasky, R. Tchoua, J. Lofstead, R. Oldfield, M. Parashar, N. Samatova, K. Schwan, A. Shoshani, M. Wolf, K. Wu, and W. Yu, "Hello adios: the challenges and lessons of developing leadership class i/o frameworks," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 7, pp. 1453–1473, 2014. [Online]. Available: http://dx.doi.org/10.1002/cpe.3125

[41] F. Zheng, H. Zou, G. Eisenhauer, K. Schwan, M. Wolf, J. Dayal, T.-A. Nguyen, J. Cao, H. Abbasi, S. Klasky, N. Podhorszki, and H. Yu, "FlexIO: I/O Middleware for Location-Flexible Scientific Data Analytics," in *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, May 2013, pp. 320–331.

[42] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng, "DataStager: scalable data staging services for petascale applications," *Cluster Computing*, vol. 13, no. 3, pp. 277–290, 2010.

[43] F. Zheng, H. Abbasi, C. Docan, J. Lofstead, Q. Liu, S. Klasky, M. Parashar, N. Podhorszki, K. Schwan, and M. Wolf, "PreDatA – preparatory data analytics on peta-scale machines," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, April 2010.

[44] U. Ayachit, A. Bauer, B. Geveci, P. O'Leary, K. Moreland, N. Fabian, and J. Mauldin, "ParaView Catalyst: Enabling In Situ Data Analysis and Visualization," in *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, ser. ISAV2015, 2015.

[45] B. Whitlock, J. M. Favre, and J. S. Meredith, "Parallel in Situ Coupling of Simulation with a Fully Featured Visualization System," in *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization*, ser. EGPGV '11, 2011, pp. 101–109.

[46] U. Ayachit, A. Bauer, E. P. N. Duque, G. Eisenhauer, N. Ferrier, J. Gu, K. E. Jansen, B. Loring, Z. Lukic, S. Menon, D. Morozov, P. O'Leary, R. Ranjan, M. Rasquin, C. P. Stone, V. Vishwanath, G. H. Weber, B. Whitlock, M. Wolf, K. J. Wu, and E. W. Bethel, "Performance Analysis, Design Considerations, and Applications of Extreme-Scale In Situ Infrastructures," in *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016, pp. 921–932.

[47] M. Dorier, G. Antoniu, F. Cappello, M. Snir, and L. Orf, "Damaris: How to Efficiently Leverage Multicore Parallelism to Achieve Scalable, Jitter-free I/O," in *2012 IEEE International Conference on Cluster Computing*, Sept 2012, pp. 155–163.

[48] A. Bhatele, T. Gamblin, S. H. Langer, P.-T. Bremer, E. W. Draeger, B. Hamann, K. E. Isaacs, A. G. Landge, J. A. Levine, V. Pascucci *et al.*, "Mapping applications with collectives over sub-communicators on torus networks," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*. IEEE, 2012, pp. 1–11.

[49] G. Mercier and E. Jeannot, "Improving MPI Applications Performance on Multicore Clusters with Rank Reordering," in *Proceedings of the 18th European MPI Users' Group Conference on Recent Advances in the Message Passing Interface*, ser. EuroMPI'11, 2011, pp. 39–49.

[50] G. Michelogiannakis, K. Z. Ibrahim, J. Shalf, J. J. Wilke, S. Knight, and J. P. Kenny, "APHiD: Hierarchical Task Placement to Enable a Tapered Fat Tree Topology for Lower Power and Cost in HPC Networks," in *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2017, pp. 228–237.

[51] T. Hoefler and M. Snir, "Generic topology mapping strategies for large-scale parallel architectures," in *Proceedings of the International Conference on Supercomputing*, ser. ICS '11, 2011, pp. 75–84.

[52] T. Hoefler, E. Jeannot, and G. Mercier, "An Overview of Topology Mapping Algorithms and Techniques in High-Performance Computing," in *High Performance Computing on Complex Environments*. Wiley, 2014, pp. 75–94.

[53] A. Bhatele, G. R. Gupta, L. V. Kale, and I. H. Chung, "Automated mapping of regular communication graphs on mesh interconnects," in *2010 International Conference on High Performance Computing*, Dec 2010.

[54] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*. Oxford Science Publications, 1989.

[55] L. DeRose, B. Homer, D. Johnson, S. Kaufmann, and H. Poxon, "Cray performance analysis tools," in *Tools for High Performance Computing*, M. Resch, R. Keller, V. Himmler, B. Krammer, and A. Schulz, Eds. Springer Berlin Heidelberg, 2008.

[56] S. Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1 – 19, 1995.

[57] S. Ito, K. Goto, and K. Ono, "Automatically optimized core mapping to subdomains of domain decomposition method on multicore parallel environments," *Computers & Fluids*, vol. 80, pp. 88 – 93, 2013.

[58] "LAMMPS Documentation," http://lammps.sandia.gov/doc/Manual.html.

[59] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-chip," *ACM Comput. Surv.*, vol. 38, no. 1, 2006.

[60] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, Aug 2005.

[61] R. Haring, M. Ohmacht, T. Fox, M. Gschwind, D. Satterfield, K. Sugavanam, P. Coteus, P. Heidelberger, M. Blumrich, R. Wisniewski, A. Gara, G.-T. Chiu, P. Boyle, N. Chist, and C. Kim, "The IBM Blue Gene/Q Compute Chip," *Micro, IEEE*, vol. 32, no. 2, pp. 48–60, 2012.

[62] D. Molka, D. Hackenberg, and R. Schöne, "Main Memory and Cache Performance of Intel Sandy Bridge and AMD Bulldozer," in *Proceedings of the Workshop on Memory Systems Performance and Correctness*, ser. MSPC '14, 2014.

[63] S. Jahagirdar, V. George, I. Sodhi, and R. Wells, "Power management of the third generation Intel core micro architecture formerly codenamed Ivy Bridge," in *2012 IEEE Hot Chips 24 Symposium (HCS)*, 2012, pp. 1–49.

[64] S. Saini, R. Hood, J. Chang, and J. Baron, "Performance Evaluation of an Intel Haswell-and Ivy Bridge-Based Supercomputer Using Scientific and Engineering Applications," in *2016 IEEE 18th International Conference on High Performance Computing and Communications*, 2016, pp. 1196–1203.

[65] J. Hofmann, G. Hager, G. Wellein, and D. Fey, "An analysis of core- and chip-level architectural features in four generations of intel server processors," in *High Performance Computing*, J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes, Eds. Springer International Publishing, 2017, pp. 294–314.

[66] "Top GREEN 500 Supercomputing Sites," https://www.top500.org/green500/list/2017/11/.

[67] G. J. Colin de Verdière, "Computing element evolution towards exascale and its impact on legacy simulation codes," *The European Physical Journal A*, vol. 51, no. 12, p. 163, Dec 2015.

[68] J. Balkind, M. McKeown, Y. Fu, T. Nguyen, Y. Zhou, A. Lavrov, M. Shahrad, A. Fuchs, S. Payne, X. Liang, M. Matl, and D. Wentzlaff, "OpenPiton: An Open Source Manycore Research Framework," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '16, 2016, pp. 217–232.

[69] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. C. Miao, J. F. B. III, and A. Agarwal, "On-Chip Interconnection Architecture of the Tile Processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, Sept 2007.

[70] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, Feb 2007, pp. 98–589.

[71] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer, "Analyzing network health and congestion in dragonfly-based supercomputers," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016, pp. 93–102.

[72] W. M. Brown, P. Wang, S. J. Plimpton, and A. N. Tharrington, "Implementing molecular dynamics on hybrid high performance computers short range forces," *Computer Physics Communications*, vol. 182, no. 4, pp. 898 – 911, 2011.

[73] "MPI IO Benchmark," https://github.com/sc18auxdata/iobenchmark.

[74] "Argonne Leadership Computing Facility's Supercomputer Mira," http://www.alcf.anl.gov/mira.

[75] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, 2nd ed. Duxbury Press, 2003.

[76] "Mapfiles," https://github.com/sc18auxdata/mapping.

[77] M. Pagel, H. Pritchard, K. McMahon, and A. Hilleary, "Performance and Functional Improvements in MPT Software for the Cray XT System," in *In Proceedings of the Cray User Group Conference (CUG)*, 2007.

[78] G. Lakner, I.-H. Chung, G. Cong, S. Fadden, N. Goracke, D. Klepacki, J. Lien, C. Pospiech, S. R. Seelam, and H.-F. Wen, *IBM System Blue Gene Solution: Performance Analysis Tools*. IBM Redbooks, 2008.

[79] X. Yang, J. Jenkins, M. Mubarak, R. B. Ross, and Z. Lan, "Watch out for the Bully!: Job Interference Study on Dragonfly Network," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16, 2016.

[80] S. Chunduri, K. Harms, S. Parker, V. Morozov, S. Oshin, N. Cherukuri, and K. Kumaran, "Run-to-run Variability on Xeon Phi Based Cray XC Systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17, 2017.

[81] T. Groves, Y. Gu, and N. J. Wright, "Understanding Performance Variability on the Aries Dragonfly Network," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2017, pp. 809–813.

[82] J. M. Brandt, E. Froese, A. C. Gentile, L. Kaplan, B. A. Allan, and E. J. Walsh, "Network Performance Counter Monitoring and Analysis on the Cray XC Platform," in *In Proceedings of the Cray User Group Conference (CUG)*, 2016.

[83] Cray, "Aries Hardware Counters," http://docs.cray.com/PDF/Aries_Hardware_Counters_S-0045-30.pdf, 2015, Cray Technical Documentation.

[84] R. Thakur, W. Gropp, and E. Lusk, "Data sieving and collective I/O in ROMIO," in *Frontiers of Massively Parallel Computation, 1999. Frontiers '99. The Seventh Symposium on the*, Feb 1999, pp. 182–189.

[85] P. Schwan, "Lustre: Building a File System for 1,000-node Clusters," in *Proceedings of the 2003 Linux Symposium*, 2003.

[86] G. M. Shipman, D. A. Dillow, S. Oral, F. Wang, D. Fuller, J. Hill, and Z. Zhang, "Lessons Learned in Deploying the World's Largest Scale Lustre File System," in *In Proceedings of the Cray User Group Conference (CUG)*, 2010.

[87] M. Chaarawi and E. Gabriel, "Automatically Selecting the Number of Aggregators for Collective I/O Operations," in *2011 IEEE International Conference on Cluster Computing*, Sept 2011, pp. 428–437.

[88] M. Moore, P. Farrell, and B. Cernohous, "Lustre Lockahead: Early Experience and Performance using Optimized Locking," in *In Proceedings of the Cray User Group Conference (CUG)*, 2017.

Appendix

This work focused on reducing the overall execution time by better process mapping, given a particular ratio of simulation and analysis tasks. The experimental results presented in this paper use empirically determined ratio of simulation to analysis tasks. Note that our mapping schemes do not depend on the particular ratio used in an experiment. Here, we provide an extensive evaluation to show the premise of ratio selection. This ratio was obtained from experiments conducted with various ratios, as shown below for the different systems, mappings and multiple atom counts. We provide results from a subset of all possible configurations (which is combinatorial).

Experiments on Mira

Table V shows the execution times using various process ratios of simulation:analysis on Mira Blue Gene/Q. The runs were repeated only 5 times, as there was negligible variance in the runtimes. Rows 1 – 4 show the results for node counts 128 – 1024, with 16 ranks per node. Column 2 shows the atom counts for each row. Columns 3 – 5 shows the results for ratios 1 (8 cores/node for simulation 8 cores/node for analysis), 3 (12 cores/node for simulation 4 cores/node for analysis), and 7 (14 cores/node for simulation 2 cores/node for analysis) using the striped mapping. Ratio of 3:1 results in the lowest runtime in all cases. There are 16 cores per node of Mira. Hence a ratio of 3:1, which leaves 4 cores for analysis per node, provides a good balance between too few and too many cores/node for analysis and therefore gives good results. Therefore, this ratio was selected for the experiments conducted on Mira in this paper.

TABLE V: Execution times for simulation+analysis on various core counts on Mira for different atom counts using three different ratios for simulation:analysis tasks – 1, 3 and 7. Ratio of 3:1 (column 4) performs the best.

#Nodes	#Atoms	Execution Time (seconds)		
		r=1	r=3	r=7
128	0.34M	31.5	31.2	33.1
256	0.80M	44.8	40.3	53.9
512	2.09M	59.9	52.2	58.3
1024	4.30M	68.9	64.5	67.3

Experiments on Cori

Figures 10 and 11 depict the execution times of simulation and analysis of 1.5 and 4 million atoms on various node counts on the Cori Haswell system (32 cores per node) using three different ratios – 1 (16 cores/node for simulation and 16 cores/node for analysis), 3 (24 cores/node for simulation and 8 cores/node for analysis) and 7(28 cores/node for simulation and 4 cores/node for analysis). A ratio of greater than 7 gives worse performance. Therefore we selected ratios of simulation:analysis processes as 1, 3 and 7 as the range of our experiments for empirically determining the best ratio. The graphs are plotted from 15 runs for each configuration and the results are shown for the NUMA-aware mapping. We note that the ratio of 3 is the best among these three ratios for both atom counts and on all node counts. Therefore we used ratio of 3:1 for experiments on Cori in this work. However, our mappings are not tied to this ratio. One of the reasons that this ratio performs better than the others could be attributed to the computation and communication characteristics of simulation and analysis used for the experiments. In this work, we do not investigate how to determine the optimal ratio of simulation to analysis tasks.

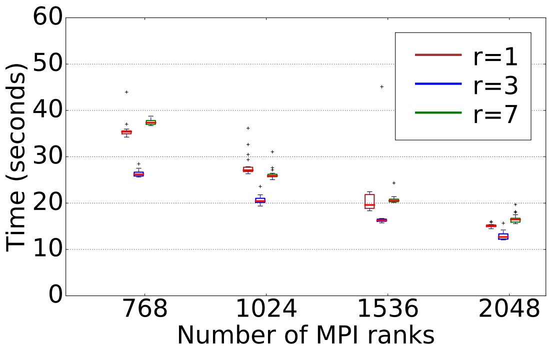


Fig. 10: Execution times of simulation and analysis of 1.5M atoms on 24, 32, 48 and 64 nodes (32 ranks per node) of Cori Haswell. Simulation:analysis ratios of 1, 3 and 7 are shown here. Ratio of 3 performs the best on average from 15 runs.

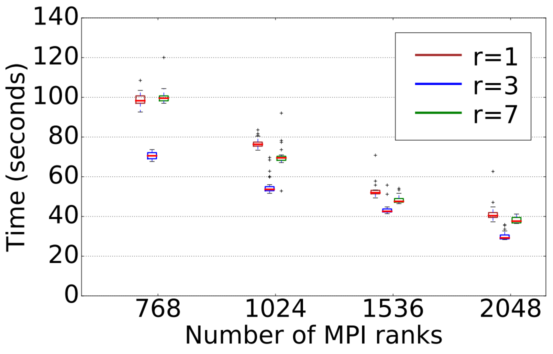


Fig. 11: Execution times of simulation and analysis of 4M atoms on 24, 32, 48 and 64 nodes (32 ranks per node) of Cori Haswell. Simulation:analysis ratios of 1, 3 and 7 are shown here. Ratio of 3 performs the best on average from 15 runs.

Experiments on Theta

Figure 12 shows the execution times of simulation and analysis of 4 million atoms using four different ratios – 1, 3, 7 and 15. Results are shown for six configurations (2 different node counts, and three different mappings for each node count). Intel Knights Landing system, Theta, has 64 cores per node. A ratio greater than 15 gives worse performance. Therefore we selected ratios of simulation:analysis processes as 1, 3, 7 and 15 as the range of our empirical evaluations. The graphs are plotted from 15 runs of each configuration and the results are shown for the default, contiguous and striped mappings on 64 and 128 nodes of Theta (64 ranks per node). We note that ratio of 7:1 is the best among these four, for each of the 6 configurations. Therefore we used this ratio for our experiments on Theta in this paper. Note that our mappings do not depend

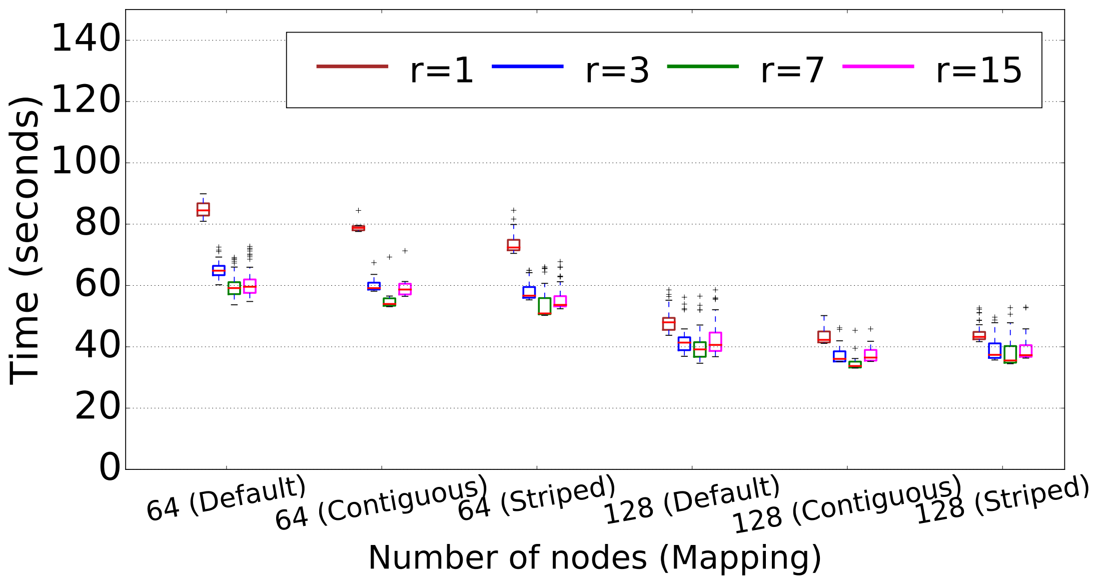


Fig. 12: Execution times of simulation and analysis of 4M atoms on 64 and 128 nodes (64 ranks per node) of Theta KNL system. Simulation:analysis ratios of 1, 3, 7 and 15 are shown here. Ratio of 7 performs the best on average from 15 runs.

on a specific ratio. Given a specific ratio, we presented various NOC-aware mapping schemes that benefit coupled codes, as shown in the results section of the paper.