

# Optimal Scheduling of In-situ Analysis for Large-scale Scientific Simulations

Preeti Malakar\*, Venkatram Vishwanath\*, Todd Munson\*, Christopher Knight\*, Mark Hereld\*, Sven Leyffer\*, Michael E. Papka\*<sup>†</sup>

\*Argonne National Laboratory

<sup>†</sup>Northern Illinois University

<pmalakar, venkat, tmunson, knightc, hereld, leyffer, papka>@anl.gov

## ABSTRACT

Today's leadership computing facilities have enabled the execution of transformative simulations at unprecedented scales. However, analyzing the huge amount of output from these simulations remains a challenge. Most analyses of this output is performed in post-processing mode at the end of the simulation. The time to read the output for the analysis can be significantly high due to poor I/O bandwidth, which increases the end-to-end simulation-analysis time. Simulation-time analysis can reduce this end-to-end time. In this work, we present the scheduling of in-situ analysis as a numerical optimization problem to maximize the number of on-line analyses subject to resource constraints such as I/O bandwidth, network bandwidth, rate of computation and available memory. We demonstrate the effectiveness of our approach through two application case studies on the IBM Blue Gene/Q system.

## Keywords

simulation, analysis, optimization, scheduling, in-situ

## 1. INTRODUCTION

Computational science simulations have now scaled to millions of cores and are able to better model higher resolution, higher fidelity and more complex physics in numerous domains including cosmology, climate, and geoscience [17, 18, 21, 22]. These simulations are now producing large amounts of data, making analysis and visualization of the simulation output essential for inferring meaningful insight from the simulations. The traditional approach for data processing for analysis and visualization in high-performance computing (HPC) is a post-processing step where the data produced by the simulations is first written out to the storage system, and is then subsequently read from the storage system for analysis. A critical bottleneck facing the analysis of data produced by scientific simulations on supercomputers, and hence scientific discovery, is the performance of the storage system relative to the computing and processing capabilities of these machines. The computing capabilities of supercomputers have increased at least twenty-fold over the past three years to meet the requirements of scientific applications,

while the storage performance has only increased by a factor of two to three times in the same time frame. This trend is expected to continue in future supercomputers. Also, post-processing enables only the analysis of the saved simulation time steps, whose frequency can be low depending on the resource characteristics [7, 36]. This prohibits analysis of the intermittent simulation time steps. Therefore, there has been a steady paradigm shift from traditional post-processing approach towards simulation-time analysis [7, 36, 39].

In HPC, the concept of simulation-time analysis, or processing data at application execution time and in an application's memory for analytics and visualization tasks, is of paramount importance to current and future generations of supercomputers. This overcomes many of the bottlenecks associated with post-processing by enabling analyses of data as the simulation is executing. This also eliminates the requirement to write to and then read from storage and the associated storage overheads. This capability enables scientific simulations to perform various data analyses and visualization tasks needed for scientific discovery and significantly minimizes the data being written to storage. This reduces both the I/O cost and the end-to-end simulation-analysis time, i.e. the time from the start of the simulation to the end of the analysis. Furthermore, simulation-time analysis can be performed at a higher temporal resolution than the simulation output frequency which is often decided based on the I/O overheads [7, 28]. Hence simulation-time analysis not only mitigates the I/O cost but also enables a better understanding of the simulated phenomena and "serendipitous discovery" [6].

There are two execution modes for simulation-time analysis depending on how the analysis is coupled with the simulation. They are (1) tightly coupled (in-situ) and (2) loosely coupled (co-analysis). Tightly coupled analysis, commonly referred to as in-situ analysis, is performed on the same resource and address space as the simulation. In the second case, the simulation and analysis are typically executed on different and dedicated resources and simulation data is transferred over the network to the analysis resources [7].

As science teams are starting to adopt simulation-time analysis in production scientific simulations and transitioning from the post-processing mode, the primary questions and challenges faced by users as they adopt simulation-time analysis in production science codes are:

- What are the different analyses that can be performed during simulation?
- Should the analysis be performed in-situ or in co-analysis mode?
- How often should each analysis be performed?
- How often should the analysis store its output?

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SC '15, November 15-20, 2015, Austin, TX, USA

Copyright 2015 ACM 978-1-4503-3723-6/15/11 ...\$15.00.

<http://dx.doi.org/10.1145/2807591.2807656>

In some cases, the frequency of analysis is empirically determined by the user [7] and this might work well for specific simulation configurations and scales. However, a solution to these questions depends on several factors including the memory, compute and communication profiles of the simulation, as well as the characteristics of each of the analyses over time, and the system resource characteristics such as the available memory, network characteristics and storage bandwidth. For example, if the analysis computation time is small, it may be preferable to perform the analysis in-situ, though this may slightly increase the simulation time. In the case of memory-intensive simulations such as FLASH [13], there may not be sufficient available memory to perform memory-intensive temporal analysis. In the case of compute-intensive simulations such as CESM [20], users may not have many spare cycles for analysis computations at simulation-time. Thus, different simulation and analysis characteristics together with resource constraints need to be considered to answer the above questions.

In this work, we focus on the case of in-situ analysis alternating with the simulation steps at an optimal frequency. Many simulation codes such as the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [33], HACC Cosmology [18], and the multiphysics multiscale FLASH simulation [38] have embedded code for performing analysis at simulation time. It is convenient to analyze the simulation data in-situ because some of the data structures required for analysis are already present in the simulation memory. For example, LAMMPS in-situ analysis code uses the particle masses, coordinates, and velocities stored in the simulation memory to compute different properties, such as the radius of gyration of a group of particles. The analyses use some of the simulation data structures and hence the simulation is blocked during the analysis. However, some of the less compute-intensive in-situ analyses can result in less than 0.1% overhead, since they can be executed at the extreme scale of the simulation. Therefore, it is faster in some cases to analyze in-situ than to transfer the simulation output and auxiliary data structures to remote memory. Furthermore, future architectures for leadership-class systems will introduce NVRAM-based small and fast memory between the compute nodes and the file system [11, 19, 24]. Presence of additional fast memory can provide scientists with the additional memory required for performing memory-intensive in-situ analyses.

We propose optimal scheduling of in-situ analyses based on the resource configurations and application demands. Limited resource availability may restrict the type and frequency of in-situ analysis. We formulate a mixed-integer linear program that recommends the analyses that can be performed in-situ within a specified time threshold and the frequency with which they should be performed. The input to our optimization problem are the time and memory requirements of the simulation and the analysis kernels. We consider different computational requirements of the analyses codes, such as the initial memory allocation and additional memory allocation for analysis. We also take into consideration the time to analyze and the time to output the analysis data. We use a simple interpolation based performance prediction strategy for predicting the execution time and memory of the kernels. We use IBM's hardware performance monitor for measuring computation and communication time per analysis function. We also use kernel functions to estimate the memory usage of the analysis routines. Our formulation also allows the ability to assign weights to the analyses in order to prioritize them. The objective of the linear program is to maximize the number of in-situ analyses and their collective importance. Given these inputs, the linear program outputs a set of feasible in-situ analyses depending on the resource characteristics and the given threshold for analysis computations. The output of

the optimization problem depends on the resource requirement of the analysis codes and the resource consumption of the simulation.

We evaluate the efficacy of our proposed scheduling mechanisms for in-situ analyses with two applications - LAMMPS [33] and FLASH [38]. These simulation codes are routinely used by several scientists on a variety of supercomputing systems. Both simulation codes have the capability to perform in-situ analyses and have several analysis routines embedded within the simulation code. The analysis routines can be optionally invoked at a desired frequency.

The novelty of our work is the ability to guide the application developer or the scientist to decide when to perform and which analyses to perform in-situ depending on the nature of analyses and the system parameters. To summarize, our main contributions are:

- Formulation of optimization problem for scheduling in-situ analyses.
- Recommendation for performing in-situ analyses based on their resource usage and available system resources.
- Performance modeling of in-situ analysis routines.
- Demonstration of in-situ analyses execution with our proposed optimization schedules with two exemplar applications, LAMMPS and FLASH, on a leadership supercomputing system.

## 2. RELATED WORK

### 2.1 Background

Analysis can be performed at the simulation site or an analysis site either during or after the simulation. Simulation site refers to the computing environment where the simulations run and analysis site refers to the computing environment used for the analyses. *Post-processing* refers to the traditional simulation-analysis workflow, where the simulation output is written to storage and the analyses is typically performed after the simulation completes. This involves writing the simulation data to storage and then reading this back from storage for analysis. Post-processing enables the scientists to analyze the entire simulation output at once, which may be necessary for some analyses. It also provides the capability for exploratory data analysis. However, as mentioned earlier, a challenge here is the need to write large amount of data to storage. Simulation-time analysis is a viable solution to this problem. *Co-processing* or *co-analysis*, a simulation-time analysis mode, typically refers to the simulation running on a dedicated resource and the analyses computations running at another dedicated resource. The data required to be analyzed is transferred from the simulation site to the analysis site. *In-situ analysis*, another simulation-time analysis mode, refers to the simulation and analysis being performed on the same resources. In this case, the simulation may periodically invoke the analyses routines at a user-designated frequency. To summarize, following are the different analysis execution modes:

- *post-processing* - after the simulation, at the analysis site
- *co-processing* - during the simulation, at the analysis site
- *in-situ* - during the simulation, at the simulation site

Next, we mention some of the existing work done in these areas.

## 2.2 In-situ analysis

There have been several recent efforts on the infrastructure and efficacy of in-situ analysis and in-transit analysis using staging nodes [2, 7, 39, 43, 44]. In [7], the authors combine in-situ and in-transit analysis for performing three kinds of analyses - descriptive statistics, topological analysis and visualization. They perform some in-situ computations and transfer the intermediate results to staging nodes. In this study, the analysis frequencies are prescribed by application developers. Their in-situ analysis stage transfers reduced data to the staging nodes. The down-sampling rate for visualization is also user-specified. Similarly, the authors in [38, 39] have used GLEAN to interface with the FLASH and PHASTA simulation codes for in-situ analysis and co-analysis. Again, the analysis frequency for this study was selected primarily to demonstrate the efficacy of the capability of performing simulation-time analysis. In our work, we envision a scenario wherein such analysis will be a common occurrence and we propose effective mechanisms for scheduling of in-situ analysis considering the simulation and analysis resource requirements and the system resource constraints.

Dreher et al. [10] presented a framework for designing, deploying and executing in-situ and in-transit analysis. The processing components of an analysis workflow are assembled in a dataflow graph, which is used for deploying the workflow. Zhang et al. [41] perform inter and intra-trajectory analysis in large protein-folding datasets on distributed memory systems. They perform analysis computations locally followed by global reduction operation. While they parallelized their analysis, they perform the analysis on frames of data that was previously generated and stored. Zhang et al. [42] present an in-situ approach for feature tracking through decentralized online clustering. Peterka et al. [32] present in-situ Voronoi tessellation for cosmological simulations. These studies also run their analyses at selected time steps. Our objective in this work, complimentary and synergistic to these efforts, is to select the optimal number of time steps for performing in-situ analyses.

## 2.3 Modeling workflow and performance

Gamell et al. [14] explored the use of NVRAM-based deep memory hierarchies by data-intensive applications. They empirically evaluated the use of NVRAM, SSD and hard disk for I/O and online data analysis. Next, they analytically modeled the workflow execution time, data transfer time and energy cost to evaluate the best choice of memory for a given workflow. In our work, we recommend the optimal set of feasible in-situ analyses. As we move towards deep memory hierarchies, these approaches can be exploited to refine our performance model. In [28, 29], the authors propose an optimization-based adaptive framework for simulation and simultaneous visualization based on the resource constraints like storage space and network bandwidth to decide the best possible parameter values for simulations and remote visualization. In our work, we consider resource constraints like I/O bandwidth and network bandwidth to schedule multiple analyses along with the simulations.

Performance coupling refers to the effect that two simultaneously executing kernels have on each other [15]. Authors in [15, 40] present a performance prediction model for applications using the knowledge of interaction between different kernels of the application. In our case, the analysis and simulation can be considered as two kernels. However, one needs to know the kernel characteristics extensively to apply the above approaches. Since we used the analyses codes embedded within the simulation codes, identifying the coupling between the two is a challenging problem. Hence, we used a model based on linear interpolation, which gives reasonable predictions [25, 27]. Performance coupling can be leveraged to re-

fine our prediction model.

Performance modeling and prediction is an active area of research. There is a rich set of efforts on modeling an application performance. Allan et al. [4] compare tools for predicting application performance on a range of architectures. However, we required modeling of certain regions of an application. Performance profiling through tools like HPCToolkit [3], PAPI [30] and TAU [37] etc. give some insights about the application timings. Intel<sup>®</sup> Trace Analyzer and Collector and Intel<sup>®</sup> VTune Amplifier are only supported on Intel platforms. For our work, we used IBM's Hardware Performance Monitoring tool [16] for profiling. Additionally, we also required estimates of memory used by the analysis routines, for which we used IBM's HPCT profiling tool [9]. As we extend our work to other systems, the above tools will help us in modeling performance on these systems.

## 3. MODELING IN-SITU DATA ANALYSIS

The analyses employed by simulations are driven by the requirements of the scientists. The analyses differ with respect to several characteristics including their memory and compute requirements. These analyses can range from collecting simple descriptive statistics to complex principal component analysis. For certain analyses, it may be desirable to perform the analysis as frequently as possible. However, performing analysis of every simulation time step output is nearly infeasible due to time and memory constraints. At the same time, the simulations have diverse set of requirements and characteristics including memory and computation time. Compute-intensive applications may have smaller tolerance for in-situ analysis, which will result in fewer analysis steps. Memory-intensive simulations may have low available free memory for analysis, which will likely impact memory-intensive analyses. System characteristics also play a key role, viz. slow I/O bandwidth will likely result in fewer analyses outputs. Thus, we need to consider both simulation and analysis resource requirements as well as computing resource characteristics in order to optimally schedule the analyses. We describe the coupling between simulation and in-situ analysis in Section 3.1. Next, in Section 3.2, we model the coupling and present the formulation of the optimization problem for scheduling the in-situ analyses.

### 3.1 Coupling between Simulation and Analysis

In this paper, we focus on in-situ analysis wherein the analysis step occurs after some number of simulation time steps. Figure 1 depicts one such ordering of this coupling illustrating the time steps when simulation ( $S$ ) is executed and analysis ( $A$ ) is executed, and the time steps when simulation output ( $O_S$ ) and analysis output ( $O_A$ ) are written to disk. Analysis may output to disk after some number of analysis steps. The analysis frequency and its output frequency are recommended by our optimization model described in the next section.

$S$  – Simulation,  $O_S$  – Simulation Output  
 $A$  – Analysis,  $O_A$  – Analysis Output  
 $S S S S A S O_S S S S A O_A S S O_S S S A S S S O_S S A O_A S S S$

**Figure 1: Analysis occurs every 4 simulation steps. Analysis outputs every 2 analysis steps. Simulation outputs every 5 simulation steps.**

As mentioned earlier, memory and execution time requirements of the analysis and simulation play an important role in the coupling. There are several other factors that also play a role. For

instance, analyses such as weekly averages in climate simulation have to be performed at a certain specific frequency. Also, the analysis may output its result instantaneously or store in an analysis memory buffer for further computation later followed by output. Certain implementation designs also affect the nature of coupling. For example, the memory required by analysis may be pre-allocated in some applications, while some may require allocation at every analysis step and hence there may be an additional time overhead. The analysis memory may be freed after computation and output or retained. For example, in the molecular dynamics simulation code LAMMPS [33], certain analysis such as the calculation of mean square displacements of molecules require a large pre-allocated memory for subsequent analysis as well as additional memory during the analysis. On the other hand, the FLASH [13] simulation allocates and deallocates the memory required for analysis on-the-fly. The coupling is also influenced by the science objectives. For instance, in certain production runs, a scientist may require a lower execution overhead (threshold) for the total in-situ analyses execution time while they may set a higher threshold during exploratory and debug runs. Additionally, the problem being simulated may require a set of analyses to be performed at simulation time and the various analyses may have different importance. Application developers can select any model of analysis coupling from these wide range of options. We carefully consider and model all such possible scenarios for in-situ analysis and present a comprehensive model of in-situ data analysis in the next section.

### 3.2 Optimal Scheduling of In-situ Analysis

Let us assume that we are given a set of desired analyses  $\mathcal{A}$  to be performed in-situ. We formulate the problem of scheduling these analyses as a mixed-integer linear program [31] with the objective of maximizing the number of in-situ analyses. Each analysis is associated with time and memory requirements. Table 1 describes the input parameters for each analysis  $i \in \mathcal{A}$ . Let  $ft$  and  $fm$  refer to the fixed initialization time and memory allocation required by an analysis at the start of the simulation execution, a one-time cost. Let  $it$  and  $im$  be the execution time and memory required at every simulation time step to facilitate the analysis during the analysis step. A simple example is temporal analysis, where  $it$  can be the time required to copy simulation data from simulation memory to temporary analysis memory so that data is not overwritten and facilitates temporal analysis. In such cases, this additional overhead is incurred at each simulation step to facilitate the analysis at the analysis step. Let  $ct$  be the time required for performing the analysis during the analysis step, and  $cm$  be the associated memory required to facilitate this analysis. A simple example of  $cm$  is the additional memory needed by the analysis to allocate any intermediate memory needed for the analysis computation. Let  $om$  refer to the output memory for the results produced by the analysis after computation and  $ot$  be the time required to write this analysis output to the storage.

Let  $bw$  be the average write bandwidth to storage from the simulation site. Application scientists may provide an upper threshold on time allowed for in-situ analyses.  $cth$  denotes the maximum threshold on analysis time per simulation time step. Let  $Steps$  denote the total number of simulation time steps. A common usage scenario for in-situ analysis is where the application scientist provides an upper bound on the total overhead of performing the in-situ analyses. For example, one could allow a maximum of 10% overhead on the overall simulation. Let  $meth$  refer to maximum memory available for analyses during the entire simulation. Scientists are expected to run multiple analyses in-situ, wherein each analysis is of varying importance to the simulation. We model this

**Table 1: Input parameters for each analysis  $i \in \mathcal{A}$  and available resources.**

Parameter	Parameter description
$ft_i$	Fixed setup time required per analysis
$it_i$	Time required per analysis per simulation time step
$ct_i$	Compute time required per analysis step
$ot_i$	Output time required per output step
$cth$	Threshold (time) per simulation step for analyses
$fm_i$	Fixed memory allocated per analysis
$im_i$	Input memory allocated per analysis per simulation step
$cm_i$	Memory allocated per analysis step
$om_i$	Memory allocated per output step
$meth$	Maximum memory available for analyses
$w_i$	Weight (importance) of each analysis
$itv_i$	Minimum interval between analysis steps
$bw$	Average I/O bandwidth between simulation site and storage

importance by assigning a weight  $w_i$  to each analysis  $i$ ; a higher weight implies more importance. Additionally, a large class of analyses require a minimum interval of  $itv$  steps between consecutive analyses steps. Examples of such analyses include daily mean average temperature in climate simulations.

Our goal is to maximize the following:

- the number of times  $|C_i|$  each analysis  $i$  is performed.
- the total number of different analyses,  $|\mathcal{A}|$ .
- the total importance of analyses performed in-situ. This is denoted by  $w_i \times |C_i|$

The decision variables are the set of feasible in-situ analyses  $\mathcal{A}$ , the set of simulation time steps at which analysis is performed  $\mathcal{C}$ , and the set of simulation time steps at which the analysis output is written  $\mathcal{O}$ .

The objective function of the linear program is given as

$$\text{maximize} \left( |\mathcal{A}| + \sum_{i \in \mathcal{A}} w_i \times |C_i| \right) \quad (1)$$

Next, we describe the three main constraints – the execution times of the various components of the in-situ analyses and their associated memory costs and minimum intervals.

**Time constraint:** The most important criteria for in-situ analysis is the ability to perform analysis within time constraints without significantly increasing the simulation time. The additional time at the simulation site for all feasible analyses should be less than the maximum allowable threshold. The time for in-situ analysis includes different components like the initialization/setup time per analysis ( $ft$ ), time required for analysis per simulation time step ( $it$ ), time required for doing analysis ( $ct$ ), and time required for writing the output of analysis ( $ot$ ). These times depend on the type of analysis and its implementation, and therefore may be zero in some cases. Time constraints are shown in Equations 2 – 4.  $tAnalyze_{i,j}$  is the cumulative time spent on the  $i^{th}$  in-situ analysis from start of simulation to time step  $j$ . The initialization cost (time) is added only once at step 0 for  $tAnalyze$  for each feasible analysis as shown in

Equation 3.

$$\begin{aligned} tAnalyze_{i,j} &= tAnalyze_{i,j-1} + it_i \\ &\quad + ct_i \text{ (if } j \in \mathcal{C}) + ot_i \text{ (if } j \in \mathcal{O}) \\ \forall i \in \mathcal{A}, j \in \{1, \dots, Steps\} \end{aligned} \quad (2)$$

$$tAnalyze_{i,0} = ft_i \quad \forall i \in \mathcal{A} \quad (3)$$

$$\sum_{i \in \mathcal{A}} tAnalyze_{i,Steps} \leq cth \times Steps \quad (4)$$

$it$  is the cost (time) paid every simulation step.  $ct$  is the time required to perform the analysis computation at every analysis step, and  $ot$  is paid whenever analysis output is written to disk.  $ot$  can be substituted by  $\frac{om_i}{bw}$ . Note that it is not required that the time spent on performing analysis at one analysis step is less than the allowed threshold per simulation time step,  $cth$ . However, at the end of the simulation, the sum of the total time spent on all in-situ analyses should be less than the threshold  $cth \times Steps$ . This is shown in inequality 4.

**Memory constraint:** The analysis computations are feasible only when required memory is available at the simulation site. Analysis may allocate a fixed amount of memory at the beginning, and allocate new memory at every simulation time step. In-situ analysis may allocate memory at every analysis step and deallocate memory at every output step. These choices depend on the analysis type and its implementation. The total memory required by an analysis will be the sum of its input and output memory requirements. This sum should be less than the available memory at the simulation site. Here, we consider all the different scenarios through the constraints 5 – 8.  $mStart$  and  $mEnd$  are the amounts of memory used by analysis at the start and end of each simulation step. Equation 5 shows the equation for  $mStart$ . At every step, the variable memory  $im$  may be allocated, hence it is added to  $mStart$ . At every analysis step,  $cm$  may be additionally allocated, and hence it is conditionally added to  $mStart$ . At every output step, analysis may allocate separate buffer  $om$  for output. Therefore,  $om$  is also conditionally added to  $mStart$ .  $mEnd$  is the available memory after the end of each step, which is equal to  $mStart$  at all steps, except for output step. At the end of output step, the additional analysis memory buffers are assumed to be freed, and reset to the initial memory allocation  $fm$  as shown in Equation 6. Memory is constrained by  $mth$ , hence the sum of  $mStart$  for all analyses in all time steps should be less than  $mth$  as shown in Equation 8.

$$\begin{aligned} mStart_{i,j} &= mEnd_{i,j-1} + im_i \\ &\quad + cm_i \text{ (if } j \in \mathcal{C}) + om_i \text{ (if } j \in \mathcal{O}) \\ \forall i \in \mathcal{A}, j \in \{1, \dots, Steps\} \end{aligned} \quad (5)$$

$$mEnd_{i,j} = \begin{cases} fm_i & \text{(if } j \in \mathcal{O}) \\ mStart_{i,j} & \text{(otherwise)} \end{cases} \quad (6)$$

$$\forall i \in \mathcal{A}, j \in \{1, \dots, Steps\} \quad (7)$$

$$\sum_{i \in \mathcal{A}} mStart_{i,j} \leq mth \quad \forall j \in \{1, \dots, Steps\} \quad (8)$$

**Interval constraint:** The minimum interval between analysis steps  $itv_i$  provides an upper bound for the number of analysis steps during  $Steps$  simulation times steps.  $\frac{Steps}{itv_i}$  denotes the maximum number of analysis steps for  $i^{th}$  analysis.

$$|\mathcal{C}_i| \leq \frac{Steps}{itv_i} \quad \forall i \in \mathcal{A} \quad (9)$$

The interval constraint enforcement is implemented similar to the

memory constraints. We keep a running total of the steps without  $i^{th}$  analysis and require this running total to exceed  $itv_i$  before  $i^{th}$  analysis can be performed. The running total is reset after the analysis is performed.

**Solution:** The solution for  $\mathcal{C}$  and  $\mathcal{O}$  are obtained from the above constraints by introducing 0-1 variables for the conditional equations. For example, in equation 5, the third term of the right hand expression  $cm_i$  (if  $j \in \mathcal{C}$ ) is replaced by  $om_i \cdot analysis_{i,j}$  where the 0-1 variable  $analysis_{i,j}$  is 0 when there is no analysis after simulation step  $j$  and 1 when there is an analysis. Similarly, in equation 5, the last term  $om_i$  (if  $j \in \mathcal{O}$ ) is replaced by  $om_i \cdot output_{i,j}$  where the 0-1 variable  $output$  is 0 when there is no output after an analyses performed in step  $j$  and 1 when there is output. Hence, the count of  $analysis$  and  $output$  for each analysis in  $\mathcal{A}$  gives the solution to our problem. The analyses for which the counts are positive form the set  $\mathcal{A}$ . Our model is implemented in the GAMS [8] modeling language and solved with CPLEX 12.6.1.

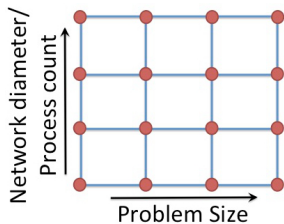
Currently, scientists perform simulation-time analyses at a pre-determined frequency, often found empirically or manually. A formulation like ours provides a systematic way to select an optimal schedule, and can benefit the scientists. Optimally scheduled in-situ analysis is highly impactful, because scientists will immediately see key results with an improved rate of sampling, potentially improving time to disseminate results. Additionally, in-situ analyses periodically outputting results would allow researchers to check behavior of a running simulation and potentially interact with it in real time.

## 4. PERFORMANCE MODELING

As seen in Table 1, our model requires estimates of the time and memory requirements of the various analyses and simulation in order to obtain a solution to the optimization problem. For this, we employ a combination of profiling and interpolation to derive the simulation time and analysis time at various system scales and problem scales. Obtaining an accurate estimate of the analysis routines embedded in the simulation code using an analytical model is a challenging problem. In our study, we perform a careful manual inspection of the analysis routine to understand its computation and communication characteristics and profile them. On our target evaluation system, a BG/Q supercomputer, we use high-resolution kernel timers, IBM's Hardware Performance Monitoring (HPM) and HPCT profiling tool [9, 16] to measure execution time and MPI communication times. HPM allows one to profile regions of code using `HPM_Start()` and `HPM_Stop()` function calls.

To obtain execution times of analysis routines, we first experimentally measure the execution times of few problem sizes on few core counts, and predict the performance for other problem sizes on various other core counts using bilinear interpolation. This is illustrated in Figure 2. The x-variable along the x-axis represents the problem size and the y-axis variable represents the number of processes in the case of compute time estimation. The red circles represent the actual measured execution times. Using this method, we observed less than 6% prediction error for the computation time.

Predicting communication time on modern networks with complex topology is a challenging problem. We use bilinear interpolation for predicting communication time and change our y-variable to the one which is more appropriate for modeling the interconnect. The MPI communications used in most of our current analyses kernels are MPI collective communications such as `MPI_Allreduce`. The maximum number of hops in the collective communication is proportional to the diameter of the network. Hence, in our current study, we use the network diameter as the y-variable for predicting



**Figure 2: Bilinear interpolation.** Red circles represent measured execution times. x-variable represents problem size. y-variable represents the process count and network diameter in case of computation and communication time interpolations respectively.

the communication performance. We observed less than 8% prediction error in communication time estimates. With regards to memory, the implementation of some analysis routines require a fixed amount of memory independent of the problem size. Other analysis routines allocate memory proportional to the problem size. We use bilinear interpolation to determine the memory requirement using the problem size as the x-variable and the process count as the y-variable. In absence of precise analytical model due to lack of complete knowledge of the application, linear interpolation gives a fairly accurate estimate as shown here by the low prediction errors, and in earlier work [27]. Note that we can refine the performance model and leverage the various performance counters and/or models present in different systems.

## 5. EXPERIMENTS AND RESULTS

We describe the experimental setup, the applications and the in-situ analyses used in the evaluation, and present the efficacy of our in-situ analyses scheduling in several typical usage scenarios.

### 5.1 Setup

We conduct our experiments on the IBM Blue Gene/Q *Mira* system at Argonne National Laboratory. *Mira* is a 48-rack machine with Power BQC 1.6 GHz processor cores. Each rack has 2 mid-planes consisting of 512 compute nodes each. Each compute node has 16 GB RAM. *Mira* has peak I/O bandwidth of 240 GB/s to the GPFS file system.

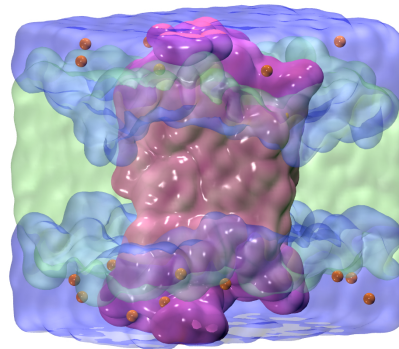
### 5.2 Application Case Studies

We evaluate our optimization-based scheduling of in-situ analyses using two applications. First, we performed our experiments using the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) classical molecular simulation code [23,33]. Two LAMMPS problems were examined in order to best span a large range of conditions explored in molecular simulations of liquids, materials, and biological systems. The first problem investigated is a box of water molecules solvating two types of ions. For this problem, the number of atoms in the system was varied from 16 million to 400 million atoms. Table 2 lists the analyses investigated for this problem. The set comprises of radial distribution functions (RDF), the mean square displacements (MSD) of molecules/ions, and velocity auto-correlation functions. Combined, these physical observables provide key information on understanding the structure and dynamics of liquids and materials [5]. Additionally, their respective algorithms (e.g. accumulating histograms, computing time averages, evaluating correlation functions) are representative of those employed in the calculation of a large class of physical observables (e.g. dielectric constant and shear viscosity).

**Table 2: Analyses for simulation of water and ions in LAMMPS**

Analysis Name	Analysis Description
hydronium rdf (A1)	Compute hydronium-water, hydronium-hydronium, and hydronium-ion RDFs averaged over all molecules
ion rdf (A2)	Compute ion-water and ion-ion RDFs averaged over all molecules
vacf (A3)	Compute velocity auto-correlation function for the water-oxygen, hydronium-oxygen, and ion atoms
msd (A4)	Compute mean squared displacements averaged over all hydronium and ions

The second LAMMPS problem explored in this work is the rhodopsin protein benchmark, which consists of a protein embedded in a membrane and solvated with water and ions [34]. For this problem, we varied the number of atoms in the system from 16 million to 1 billion atoms. The set of analyses investigated for this problem are the radius of gyration for a single protein and 2D histogram of density profiles for the membrane and protein structures (listed in Table 3). These properties provide insight into the distribution of particles within an assembled structure and throughout the system. Just as for the water+ions system, these analyses are commonly employed in studies of aggregate structures and assemblies and are of interest to a large community of researchers [5]. Figure 3 shows a snapshot



**Figure 3: Snapshot of the LAMMPS rhodopsin benchmark (32,000 atoms):** protein (solid purple; center) is embedded in membrane (translucent green; middle) and solvated with water (translucent blue; top and bottom) and ions (orange spheres).

of the base LAMMPS rhodopsin benchmark (32,000 atoms) using VMD [1]. The solid purple structure in the center is the protein. It is embedded in membrane which is shown in translucent green in the middle and solvated with water (shown in translucent blue at the top and bottom) and ions (shown as orange spheres).

The second application used in our evaluation is the FLASH multiphysics multiscale simulation code [13]. FLASH is an adaptive mesh, parallel hydrodynamics code developed to simulate high energy density physics and astrophysical thermonuclear flashes in two or three dimensions, such as Type Ia supernovae, Type I X-ray bursts, and classical novae. It solves the compressible Euler equations on a block-structured adaptive mesh. FLASH provides an Adaptive Mesh Refinement (AMR) grid using a modified version of the PARAMESH package [26] and a Uniform Grid (UG) to store Eulerian data. For this study, we used the Sedov simulation



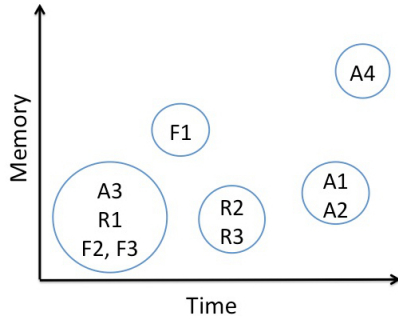
**Table 3: Analyses for the rhodopsin protein LAMMPS benchmark**

Analysis Name	Analysis Description
radius of gyration (R1)	Compute the radius of gyration for a single protein
membrane histogram (R2)	Compute the 2D histogram of the density profiles of all membranes
protein histogram (R3)	Compute the 2D histogram of the density profiles of all proteins

included in the FLASH simulation distribution. Sedov evolves a blast wave from a delta-function initial pressure perturbation [12]. We ran the FLASH simulation for the Sedov problem using three dimensions with  $16^3$  cells per block. Each block consists of 10 mesh variables and we can vary the problem size by adjusting the global number of blocks to simulate larger domain sizes. We perform three different analyses for FLASH, namely

- vorticity (F1)
- L1 error norm for density and pressure (F2)
- L2 error norm for x, y, z velocity variables (F3)

Figure 4 summarizes the relative execution time and memory requirements of the various analyses listed above.

**Figure 4: The relative execution time and memory consumption profiles of the various in-situ analyses.**

## 5.3 Results

In this section we present the results of our experiments with LAMMPS and FLASH on Mira. We used GAMS modeling language for modeling the optimization problem, and solved using IBM CPLEX solver v12.6.1 on a 2.2 GHz Intel CPU. The execution times of the solver for all the problems presented in this paper varied between 0.17 sec – 1.36 sec. The variation in time is due to the difference in problem input parameters.

### 5.3.1 Benefit of in-situ analysis

Table 4 shows the time for MSD calculation (analysis A4 of Table 2) using a serial custom post-processing tool on Intel Core i7 3.4 GHz system and analyzing in-situ on 16384 cores of Mira. The analysis was done for 1000 simulation steps and the simulation was configured to write an output frequency every 100 steps. The first column shows the number of atoms in the system. The second column depicts the time to read the LAMMPS trajectory file when analyzing in post-processing mode. The third and fourth columns show the times to analyze in a post-processing mode and in an in-situ mode respectively. It can be easily seen that the time

to post-process (read+analyze) is significantly higher than doing this in-situ. One of the primary reasons for this is the mitigation of reading cost from the storage. Secondly, in-situ analysis can be performed at scale along with the simulation on powerful compute resources. The overhead for in-situ MSD was less than 5% in both cases. The time to read the output increases with increase in data size (number of atoms). However, significant savings are possible with in-situ analysis. This shows the benefit of in-situ analysis for larger system of atoms.

**Table 4: Analysis time (post-processing and in-situ) for MSD calculation of water+ions simulation (1000 steps) in LAMMPS. In-situ analysis incurs significantly lower cost than post-processing.**

Number of atoms	Time to read (s)	Post-processing time (s)	In-situ analysis time (s)
12,544	23.89	1.03	0.01
100,352	2413.11	17.85	0.03

### 5.3.2 Efficacy of in-situ analysis scheduling

We study the impact of varying the threshold time for analysis, specified as a percentage of the simulation time, on scheduling the various in-situ analyses. This represents a common expected use-case wherein the scientist/user is willing to tolerate execution of the various in-situ analysis within a certain threshold of the simulation time. To study this, we use the LAMMPS water+ion problem with a problem size of 100 million atoms and its associated analyses (see Table 2). Each analysis was assigned the same importance/weight and the minimum interval for each analysis is 100 simulation steps. The experiment is performed on 16384 processes (1024 nodes, 16 ranks per node) of Mira. The total simulation time for 1000 time steps is 646.78 sec. The solution of our optimization model is depicted in Table 5.

The first column shows the threshold allowed for the in-situ analyses as a percentage of the simulation time and the corresponding time. The columns 2 – 5 show the number of times each of the analyses hydronium rdf (A1), ion rdf (A2), vacf (A3) and msd (A4) can be performed within the given threshold during a simulation of 1000 steps. These numbers are the output produced by the optimization solver taking into account the execution time and memory requirement estimates of each analysis. Column 6 depicts the total execution time taken by the in-situ analyses based on the proposed optimization solution when run with the simulation. The last column depicts the ratio(%) of the actual execution time for the in-situ analyses with respect to the threshold limit. The threshold is varied

**Table 5: Threshold (%) and analyses frequencies of 4 analyses for 100 million-atom water+ions simulation in LAMMPS on 16384 cores of Mira.**

Threshold % (time in sec)	A1	A2	A3	A4	Analyses time (sec)	% within threshold
20 (129.35)	10	10	10	4	103.47	80
10 (64.69)	10	10	10	2	52.79	81.6
5 (32.34)	10	10	10	1	27.45	84.87
1 (6.46)	10	10	10	0	2.11	32.66

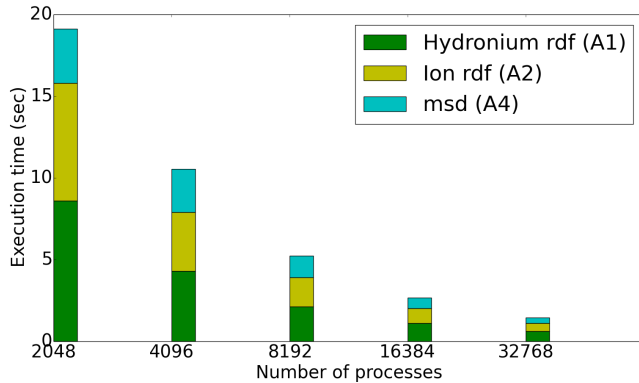
from 20% to 1%. It can be seen that A1, A2, A3 are performed 10 times in 1000 simulation time steps, i.e. once every 100 simulation steps - the maximum allowable frequency. The frequency of A4 decreases as we decrease the threshold, i.e. decrease in the

allowed additional time for in-situ analyses. This is because A4 has both significantly higher analysis execution time and analysis output time as well as requires more memory. Therefore the optimization model recommends fewer number of A4 analysis steps to satisfy the time threshold requirement.

From the last column, we observe that the time taken by the in-situ analyses is always within the specified threshold. As we increase the threshold, the total analyses times are always within 80% of the allowed analyses time. A threshold of 5–20% for this configuration results in higher threshold and hence the solver is able to recommend higher number of analyses that can be performed in-situ. At a threshold of 1%, the solver is unable to accommodate A4 as the execution requirements of A4 exceed that of the threshold and the solver schedules the other three analyses at the maximum frequency. Thus, the linear program recommends optimal parameters (analyses frequencies) for the given application parameters and system parameters. The model is flexible enough to account for a user-specified threshold and propose appropriate optimal schedule for the various in-situ analyses. We expect this use-case to play an important role as scientists start adopting simulation-time analysis.

### 5.3.3 Efficacy of in-situ scheduling for moldable jobs

A new capability of current job schedulers is the ability to accommodate moldable jobs whose running sizes (i.e., number of nodes) are dynamically decided by the scheduler at job allocation time [35]. Moldable jobs are typically strong scaling jobs solving the same problem at different core counts and a primary purpose is to fill the backfill queues of the system to improve the overall utilization of the computing resource. Figure 5 depicts the strong scaling results for the three analyses, A1, A2 and A4, (see Table 2) in the 100 million atom LAMMPS water+ions simulation as we scale from 2048 to 32768 cores on Mira. The simulation time per time step on 2048, 4096, 8192, 16384 and 32768 processes are 4.16, 2.12, 1.08, 0.61 and 0.4 sec respectively. The threshold for the total in-situ analyses time is set to 10% of simulation time. The stacked bars at each core count illustrate the times taken by each analysis in 1000-step simulation. Note that the simulation time decreases with increase in number of processes. This implies a decrease in allowable time for analyses as we have a fixed threshold value. It should be noted that the threshold is a percentage of the simulation time, and as we lower the simulation time, it also lowers the analyses time upper bound.



**Figure 5: Strong scaling for analyses in 100 million atom water+ions LAMMPS simulation on 2048 – 32768 cores of Mira.**

The analyses times shown in the figure are based on the frequencies recommended by the linear program. It takes into account the compute and memory characteristics of A1, A2 and A4 and recom-

mends a frequency of 10 for A1 and A2 on all core counts. However, it recommends decreased frequency of A4 from 10 on 2048 cores to 1 on 32768 cores. The reason is that the MSD analyses (A4) does not scale and takes similar times on all core counts. A1 and A2 scale well at higher number of processes. Therefore, even with decrease in the total allowable analyses time on higher process counts, A1 and A2 can be performed at a high frequency. Thus, our proposed optimization framework effectively helps schedule in-situ analyses for moldable jobs, which are of increasing importance for leadership supercomputers.

### 5.3.4 Efficacy of in-situ scheduling given a total threshold

Another common usage scenario is wherein a scientist/user specifies a maximum total allowable threshold time for the desired set of in-situ analyses. This is distinct from the previous cases where the threshold is a percentage of the simulation time. Specifying a total time for the threshold helps a scientist better plan the run time of their jobs as part of job submission script. Our linear program can recommend solutions based on this maximum time as  $cth$  in Equation 3 of the optimization problem.

Table 6 depicts such a scenario for the rhodopsin protein benchmark in LAMMPS consisting of 1 billion atoms (refer Section 5.2) where the user specifies the maximum allowable in-situ analysis time. The simulation was executed on 32768 cores (2048 nodes, 16 ranks per node) of Mira. Without any in-situ analysis, the simulation takes 5163.03 seconds for 1000 time steps on 32768 cores. The desired set of in-situ analyses R1, R2 and R3 are listed in Table 3. Each analysis was assigned the same importance/weight and the minimum interval for each analysis is 100 simulation steps. The time taken for each analysis followed by an output step was 0.003 sec, 17.193 sec and 17.194 sec for R1, R2 and R3 respectively.

**Table 6: Analysis frequencies, analysis times, and corresponding thresholds for 1 billion atoms rhodopsin simulation (1000 steps) in LAMMPS on 32768 cores of Mira.**

Total threshold (sec)	R1	R2	R3	% within threshold
200	10	4	7	94.59
100	10	2	3	85.99
60	10	1	2	86.01
20	10	1	0	86.11
10	10	0	0	0.3

We vary the threshold from 200 secs to 10 secs, and our optimization solution gives the frequencies for each analysis. This is depicted in columns 2 – 4. We see that as we decrease the maximum allowable time from 200 sec to 10 sec, the total number of analyses decreases from 21 to 10. It is worthwhile to note that since R1 has much lower compute time, the solver recommends a higher frequency for R1. For example, when the maximum allowable time is 100 sec, R1 can be performed 10 times in 1000 simulation time steps, i.e. every 100 steps. However, the recommendation for R2 and R3 are 2 and 3 times respectively during 1000 simulation steps, i.e. every 500 and 330 steps respectively.

The last column depicts the percentage of the allowed threshold time taken by the in-situ analysis executions. We can observe that the analyses times utilize more than 85% of the specified threshold time for almost all cases. At a high threshold of 200 secs, the solver is able to recommend more number of analyses to fully utilize the given threshold time. As the threshold decreases, the solver has lesser options to pack the allotted time. In case of a threshold of



10 secs, a time lower than the minimum time needed for running either R2 or R3, we get an expected solution of 10 steps of R1 - the minimum interval of 100 steps yielding a maximum frequency of 10, and, has a low utilization. In all other cases, we observe a usage of 86% to 94.59% of the allowed threshold time. Thus, the model is flexible enough to account for user requirements on the total time for the various in-situ analysis and yield high utilization. We expect this use case to play an important role for accounting for the total runtime for job scheduling.

### 5.3.5 Effect of output time on analyses time

Table 7 shows the total number of feasible analyses in 1000 simulation steps for 1 billion atoms rhodopsin simulation in LAMMPS on 32768 cores of Mira. The simulation time for 1000 time steps is 5163.03 sec. The simulation produces 91 GB of data per output step and the default output frequency is every 100 steps. The first column shows the total output time, second column shows the user-specified threshold for analyses, and the third column shows the total number of feasible analyses (A1, A2, A3 and A4) recommended by our optimization model. For the default 10 output steps, the total output time (using MPI parallel I/O) is 200.6 sec (first row). Assuming, the user specifies a threshold of 50 sec, the total number of feasible analyses for A1, A2, A3 and A4 is 12. To maximize the benefit of in-situ analyses, the user can decrease the output frequency and hence gain additional time for in-situ analyses. We show the effect of this in second and third rows. If the output frequency is halved, the output time is also halved. Therefore, the user can utilize this additional time in specifying a higher threshold, which results in increase in number of in-situ analyses. It can be observed that by decreasing output frequency, one can increase the number of in-situ analyses. Decrease in output time is

**Table 7: Output times, thresholds, and number of analyses for 1 billion atoms rhodopsin simulation (1000 steps) in LAMMPS on 32768 cores of Mira.**

Output time (sec)	Threshold (sec)	Number of analyses
200.6	50	12
100.3	150.3	18
50.1	200.5	21

also possible by using a higher bandwidth storage like NVRAM. Thus, by selecting a different resource for storing output, one can perform more number of in-situ analyses in the same time. Thus, our optimization approach can be used to understand and predict the effect of different system characteristics on scheduling in-situ analyses.

### 5.3.6 Impact of the importance of analyses on the schedule

Depending on the objective of the scientific simulation, a user may have several analyses to execute in-situ, wherein certain analyses may have higher priority over the others. Also, given a time constraint, one may want to prioritize the set of analyses performed in-situ. We evaluate the efficacy of our optimization model given the importance (weights) of the in-situ analyses. Table 8 depicts the execution times for the three analyses in FLASH - vorticity (F1), L1 error norm (F2) and L2 error norm (F3). The analysis times for one step of F1, F2, and F3 are 3.5 sec, 1.25 sec and 2.3 ms respectively on 16384 cores of Mira. Each simulation time step typically takes 0.87 sec. The table shows the recommendation from the linear program on the frequency of F1, F2 and F3 for a 1000-step

simulation. Importance (weight) of an analysis is denoted by integers and higher weight implies a higher priority of the analysis. The first two rows depict the results when assigning the same im-

**Table 8: Analyses frequencies in FLASH Sedov simulation (1000 steps) on 16384 cores of Mira. Optimization problem recommends different frequencies for difference in importance of the analyses.**

	Vorticity (F1)	L1 error norm (F2)	L2 error norm (F3)
Importance (I1)	1	1	1
Frequency	1	10	10
Importance (I2)	2	1	2
Frequency	5	0	10

portance (I1) to all three analyses. The next two rows depict the result of the optimization when we assign different importance (I2) to the analyses. Let us assume a fixed threshold of 5%, i.e. the user allows a maximum execution time of 43.5 sec for the in-situ analyses during a 1000-step simulation (870 sec). Note that F1 is more compute intensive than F2 and F3. Hence, if we assign the same importance to all three, the proposed solution is to perform F1 once (second row, second column) taking into account the analyses and output times of all three. If the user now prefers F1 and F3 over F2 and reassigns the weights accordingly, the solution of the optimization model yields a higher frequency for F1 and F3 (fourth row). Note that F3 has a higher frequency than F1 because F3 consumes lesser time and memory than F1. This demonstrates the effectiveness of the optimization approach in deciding the different analyses frequencies based on user/application requirements. The approach also provides flexibility to a user to choose the optimal number of analyses based on their importance.

## 6. CONCLUSIONS AND FUTURE WORK

We envision in-situ analysis to be a common execution modality for science campaigns on supercomputing systems. To facilitate this, we tackle the challenge of scheduling the analyses with the simulation. We consider and model the time and memory requirements of the analyses, the importance of the analyses and the system parameters like the computation time, I/O bandwidth, and maximum available memory to decide the optimal frequencies of the in-situ analyses. We use a mixed-integer linear program to solve this problem. Our optimization approach recommends the analyses frequencies depending on a threshold limit on maximum additional time (overhead) for in-situ analyses. This gives ability to the scientist to choose those analyses which are more important and which can be performed within the time and memory constraints. We evaluate our approaches using two configurations of the LAMMPS simulation and one configuration of the FLASH simulation and scale to 32K cores. Our optimization approach yields optimal in-situ schedules while satisfying the various user constraints.

In future, we will extend this work to optimally schedule the analyses computations on different resources. This requires transferring huge data in some cases. It will be also interesting to pipeline visualization along with simulation and analysis.

## Acknowledgments

This research has been funded in part and used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357. This

work was supported in part by the DOE Office of Science, Advanced Scientific Computing Research, under award number 57L38, 57L32, 57K07 and 57K50.

## 7. REFERENCES

- [1] Vmd: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33 – 38, 1996.
- [2] Hasan Abbasi, Matthew Wolf, Greg Eisenhauer, Scott Klasky, Karsten Schwan, and Fang Zheng. DataStager: scalable data staging services for petascale applications. *Cluster Computing*, 13(3):277–290, 2010.
- [3] Laksono Adhianto, Sinchan Banerjee, Mike Fagan, Mark Krentel, Gabriel Marin, John Mellor-Crummy, and Nathan R. Tallent. HPCToolkit: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, 22(6):685–701, 2010.
- [4] R Allan and A Mills. Survey of HPC Performance Modelling and Prediction Tools. *Science and Technology*, 2010.
- [5] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford Science Publications, 1989.
- [6] David A. Bader. *Petascale Computing: Algorithms and Applications*. CRC Press, 2008.
- [7] J.C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, Tong Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, P. Pebay, D. Thompson, Hongfeng Yu, Fan 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.
- [8] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A User's Guide*. The Scientific Press, South San Francisco, California, 1988.
- [9] I-Hsin Chung, Robert Walkup, Hui-Fang Wen, and Hao Yu. MPI Performance Analysis Tools on Blue Gene/L. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2006.
- [10] 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*, pages 277–286, May 2014.
- [11] Brian Van Essen, Roger Pearce, Sasha Ames, and Maya Gokhale. On the Role of NVRAM in Data-intensive Architectures: An Evaluation. *International Parallel and Distributed Processing Symposium*, pages 703–714, 2012.
- [12] FLASH User Guide. <http://flash.uchicago.edu/website/>.
- [13] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. *Astrophysical Journal, Supplement*, 131:273–334, 2000.
- [14] M. Gamell, I. Rodero, M. Parashar, and S. Poole. Exploring energy and performance behaviors of data-intensive scientific workflows on systems with deep memory hierarchies. In *High Performance Computing (HiPC), 2013 20th International Conference on*, pages 226–235, Dec 2013.
- [15] Jonathan Geisler and Valerie Taylor. Performance Coupling: A Methodology for Predicting Application Performance using Kernel Performance. In *In Proceedings of the ninth SIAM conference on parallel processing for scientific computing*, 1999.
- [16] Megan Gilge. *IBM System Blue Gene Solution: Blue Gene/Q Application Development*. IBM Redbooks, June 2013.
- [17] L. Grinberg, V. Morozov, D. Fedosov, J.A. Insley, M.E. Papka, K. Kumaran, and G.E. 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.
- [18] Salman Habib and et al. 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.
- [19] Jiahua He, A. Jagatheesan, S. Gupta, J. Bennett, and A. Snavely. DASH: a Recipe for a Flash-based Data Intensive Supercomputer. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2010.
- [20] James W. Hurrell and et al. The Community Earth System Model: A Framework for Collaborative Research. *Bulletin of the American Meteorological Society*, 94(9):1339–1360, September 2013.
- [21] Tsuyoshi Ichimura, Kohei Fujita, Seizo Tanaka, Muneo Hori, Madgedara Lalith, Yoshihisa Shizawa, and Hiroshi Kobayashi. Physics-based Urban Earthquake Simulation Enhanced by  $10.7 \text{ BlnDOF} \times 30 \text{ K Time-step}$  Unstructured FE Non-linear Seismic Wave Simulation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014.
- [22] Peter Johnsen, Mark Straka, Melvyn Shapiro, Alan Norton, and Thomas 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.
- [23] LAMMPS Molecular Dynamics Simulator. <http://lammps.sandia.gov>.
- [24] Ning Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn. On the role of burst buffers in leadership-class storage systems. In *IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*, April 2012.
- [25] L. Lopes, J. Zilinskas, A. Costan, R.G. Cascella, G. Kecskemeti, E. Jeannot, M. Cannataro, L. Ricci, S. Benkner, S. Petit, et al. *Euro-Par 2014: Parallel Processing Workshops: Euro-Par 2014 International Workshops, Porto, Portugal, August 25-26, 2014, Revised Selected Papers*. Number pt. 2 in Lecture Notes in Computer Science / Theoretical Computer Science and General Issues. Springer International Publishing, 2014.
- [26] Peter MacNeice, Kevin M. Olson, Clark Mobarrry, Rosalinda de Fainchtein, and Charles Packer. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126(3):330–354, 2000.
- [27] Preeti Malakar, Thomas George, Sameer Kumar, Rashmi Mittal, Vijay Natarajan, Yogish Sabharwal, Vaibhav Saxena, and Sathish S. Vadhiyar. A Divide and Conquer Strategy for Scaling Weather Simulations with Multiple Regions of Interest. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012.

- [28] Preeti Malakar, Vijay Natarajan, and Sathish S. Vadhiyar. An Adaptive Framework for Simulation and Online Remote Visualization of Critical Climate Applications in Resource-constrained Environments. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2010.
- [29] Preeti Malakar, Vijay Natarajan, and Sathish S. Vadhiyar. InSt: An Integrated Steering Framework for Critical Weather Applications. In *Proceedings of the International Conference on Computational Science*, volume 4, pages 116 – 125, 2011.
- [30] P. Mucci, S. Browne, C. Deane, and G. Ho. PAPI: A Portable Interface to Hardware Performance Counters. In *Proceedings of Department of Defense HPCMP Users Group Conference*, June 1999.
- [31] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, NY, 1988.
- [32] T. Peterka, J. Kwan, A. Pope, H. Finkel, K. Heitmann, S. Habib, Jingyuan Wang, and G. Zagaris. Meshing the universe: Integrating analysis in cosmological simulations. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 186–195, Nov 2012.
- [33] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1 – 19, 1995.
- [34] LAMMPS Rhodopsin Protein Benchmark. <http://lammps.sandia.gov/bench.html#rhodo>.
- [35] Osman Sarood, Akhil Langer, Abhishek Gupta, and Laxmikant Kale. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 807–818, 2014.
- [36] C. Seshadhri, Ali Pinar, David Thompson, and JanineC. Bennett. Sublinear algorithms for extreme-scale data analysis. In Janine Bennett, Fabien Vivodtzev, and Valerio Pascucci, editors, *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, pages 39–54. Springer Berlin Heidelberg, 2015.
- [37] Sameer S. Shende and Allen D. Malony. The Tau Parallel Performance System. *International Journal of High Performance Computing Applications*, 20(2):287–311, 2006.
- [38] V. Vishwanath, M. Hereld, M. E. Papka, R. Hudson, G. Jordan, and C. Daley. In Situ Data Analytics and I/O Acceleration of FLASH simulations on leadership-class systems with GLEAN. In *Journal of Physics: Conference Series, Proceedings of SciDAC*, 2011.
- [39] V. Vishwanath, M. Hereld, and M.E. Papka. Toward simulation-time data analysis and I/O acceleration on leadership-class systems. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 9–14, Oct 2011.
- [40] Xingfu Wu, V. Taylor, J. Geisler, and R. Stevens. Isocoupling: reusing kernel coupling values to predict the performance of parallel applications. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, April 2004.
- [41] Boyu Zhang, Trilce Estrada, Pietro Cicotti, and Michela 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, IPDPS '14*, pages 221–230, 2014.
- [42] Fan Zhang, S. Lasluisa, Tong Jin, I. Rodero, Hoang Bui, and M. Parashar. In-situ feature-based objects tracking for large-scale scientific simulations. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 736–740, Nov 2012.
- [43] Fang Zheng, H. Abbasi, C. Docan, J. Lofstead, Qing 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] Fang Zheng, Hongbo Zou, G. Eisenhauer, K. Schwan, M. Wolf, J. Dayal, Tuan-Anh Nguyen, Jianting Cao, H. Abbasi, S. Klasky, N. Podhorszki, and Hongfeng Yu. FlexIO: I/O Middleware for Location-Flexible Scientific Data Analytics. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 320–331, May 2013.