Scalable Performance Prediction of Irregular Workloads in Multi-Phase Particle-in-Cell Applications

Sai P. Chenna, Herman Lam, Greg Stitt
Electrical and Computer Engineering
University of Florida
Gainesville, FL USA
{schenna,hlam,gstitt}@ufl.edu

S Balachandar
Mechanical and Aerospace Engineering
University of Florida
Gainesville, FL USA
bala1s@ufl.edu

***Abstract*—The demand for reliable performance prediction of large-scale systems is ever-increasing. With the constant need of application users for faster execution and expectation on system administrators to efficiently allocate system resources, reliable performance prediction frameworks are crucial for identifying scalability bottlenecks which result in suboptimal performance and poor resource utilization. Such challenges in scalable performance prediction are further exacerbated by irregular applications which present dynamic workload fluctuations across processors. In this paper, we propose a novel trace-driven performance prediction framework to reliably predict the performance of a class of irregular applications which employs the Particle-in-Cell (PIC) method. The framework provides multiple advantages in terms of scalability prediction, algorithm evaluation, and performance tuning. To demonstrate scalability prediction, we predicted the performance of CMT-nek, a large-scale scientific application which employs the PIC method, on Quartz (a DOE HPC system) with an average Mean Absolute Percentage Error (MAPE) of 8.42%. For algorithm evaluation, we evaluated the efficiency of two candidate particle mapping algorithms used in CMT-nek. For performance tuning, we performed a parameter study to assess the impact of a key problem parameter in CMT-nek on application performance.**

***Index Terms*—performance modeling, irregular workloads, trace-based simulation, particle-in-cell, load-balancing**

I. Introduction

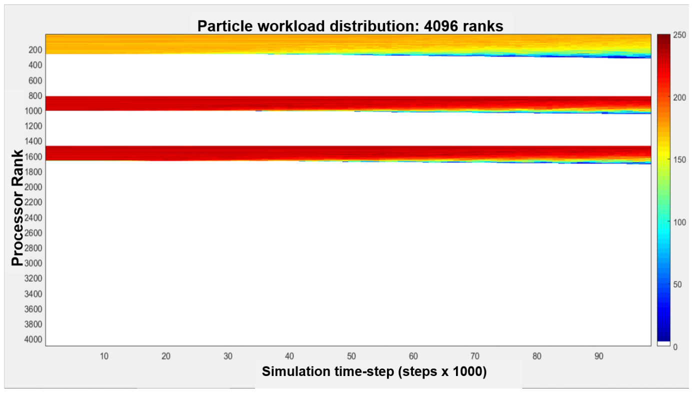
The critical importance of performance prediction for identifying performance bottlenecks of large-scale systems has led to the introduction of a variety of modeling-based prediction methods. While some methods rely on generating analytical performance models [1]–[5] which include system parameters such as processor count, interconnect, etc., other approaches use node or device-level performance models in conjunction with simulators to study system performance [6]–[8].

One key limitation of these existing methods is the assumption of a workload that is statically distributed across the processors. Although effective for representative applications, these methods are not sufficient for increasingly common applications with irregular workloads that change throughout the execution of an application. In order to reliably predict the performance of such irregular applications on large-scale systems, performance prediction must accurately model the dynamic workload on all the system nodes throughout the execution. In this paper, we accomplish this goal by presenting a trace-driven workload prediction approach for one such class of irregular applications which exhibit dynamic workload behavior, namely the Particle-in-Cell (PIC) method, which is widely used in the fields of plasma physics and fluid mechanics. As a case study, we evaluate our approach on CMT-nek, which is a potential exascale Compressible Multiphase Turbulence application that uses the PIC method to simulate particle-laden explosively dispersed turbulent flows.

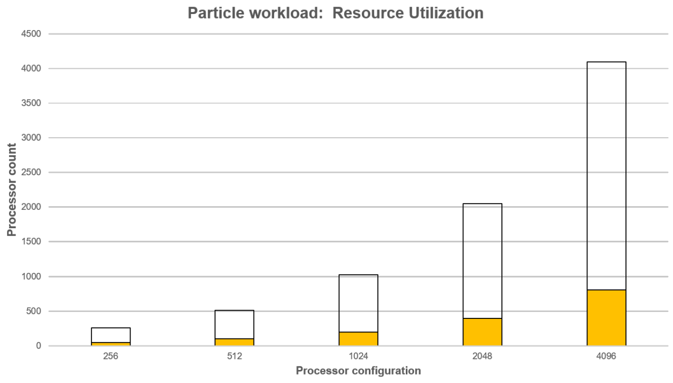
One of the salient features of the PIC method is the constant interaction between the particles and the grid. The computation grid is distributed evenly across the processors. During execution, particles move within the grid, resulting in the dynamically changing workload variation across processors that existing prediction methods do not model. Figure 1(a) illustrates this changing workload distribution using a heat-map of an experiment run using CMT-nek on the Vulcan supercomputer [9], which shows the non-homogeneous distribution of particles across processors during execution. Such non-homogeneous distribution is primarily due to two factors: initial particle distribution, which is specific to the problem being simulated in CMT-nek, and the particle movement during simulation which causes particles to cross processor domains leading to workload fluctuations. This irregular workload distribution leads to the huge load-imbalance shown in Figure 1(b), where 81% of the processors, on average, remained idle with no particle workload during the entire simulation.

To predicting application performance with such irregular workloads, our performance prediction framework captures the dynamic workload on any given number of processors using a single application trace. The framework provides multiple advantages in terms of scalability prediction, algorithm evaluation, and performance tuning. To demonstrate scalability

This work is supported by the U.S. Department of Energy, National Nuclear Security Administration, Advanced Simulation and Computing Program, as a Cooperative Agreement under the Predictive Science Academic Alliance Program, under Contract No. DE-NA0002378. The work was also supported by the National Science Foundation under grant CNS-1718033.



(a)



(b)

Fig. 1: (a) Heatmap plot depicting particle distribution of CMT-nek simulation across 4096 processors on Vulcan. White patches in the plot depict processors with no particles residing in them throughout the simulation. (b) Number of processors with non-zero particles residing in them during the simulation for different processor configurations. On an average, 81% of processors have zero particle workload throughout the execution.

prediction, our performance prediction framework predicted the performance of CMT-nek on Quartz (a DOE HPC system) with an average Mean Absolute Percentage Error (MAPE) of 8.42%. For a given problem case-study involving 599,257 particles and 216,225 elements, we are able to predict the ideal processor count to achieve optimal performance. For algorithm evaluation, we evaluated the impact of two different mapping algorithms, namely element-based and bin-based mapping (discussed in III-B and III-C), on application performance, demonstrating that using the latter approach would reduce the peak particle workload by two orders of magnitude. Finally, to demonstrate performance tuning, we performed a parameter study to quantify and validate the performance impact of a key problem parameter in the CMT-nek simulation.

The rest of the paper is organized as follows. Section II describes our performance prediction framework, elaborating on our dynamic workload generation process. Section III provides a brief background on our application case-study, CMT-nek, and explain the two different particle mapping strategies used in CMT-nek. Section IV presents and evaluates our prediction results. Section V provides an overview of related research on different load-balancing strategies for scalable PIC implementations and different trace-based performance prediction approaches. Section VI concludes our current work and discuss future directions.

II. Methodology

As mentioned above, Particle-in-Cell (PIC) algorithms involve tight interaction between particles and the neighboring grid. In most PIC algorithms, the computation grid is static while the particles keep moving across the grid, owing to the forces acting upon them. The dynamic nature of the particle movement is highly problem-dependent. The initial problem conditions, particle density, and Eulerian forces acting upon the particles, all contribute to particle movement. Such continuous movement of particles may cause workload fluctuations, based on how the particles are distributed across the particles.

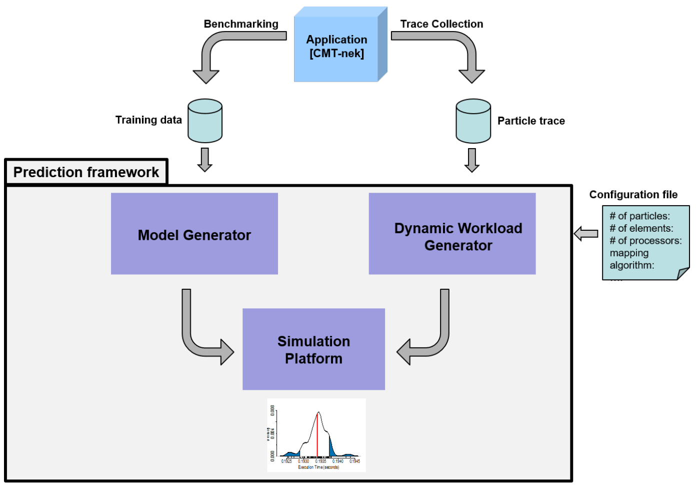


Fig. 2: Performance prediction framework workflow.

Predicting the workload across the processors during the entire simulation is not trivial.

The particle workload on each processor, which is primarily specified in terms of number of particles residing in a processor, is determined by the particle mapping algorithm. The particle mapping algorithm distributes particles in the computation grid across processors in a way to optimize the workload distribution while preserving the particle-grid locality. Most particle mapping algorithms [10]–[12] make use of particle location while distributing the particle workload across the processors. To accurately predict application performance under such irregular parallel workloads, we need to capture the dynamic workload fluctuations across the processors over the course of execution. Using our approach, we calculate the workload on each processor by mimicking the particle mapping strategy onto an application trace. This application trace contains the particle location sampled at pre-defined intervals. We will refer to this application trace as particle trace. By synthetically generating the workload based on this particle trace and the mapping algorithm, we can study the workload distribution on any given number of processors.

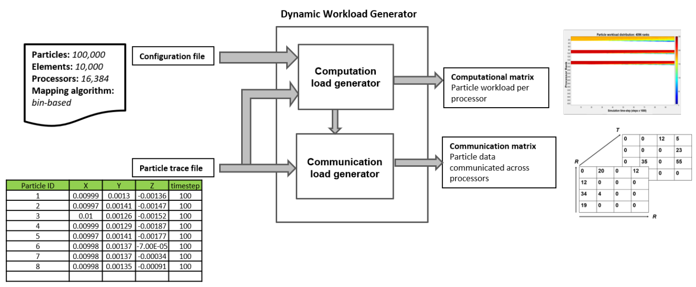
Figure 2 shows the workflow of our proposed performance prediction framework. The key modules of our prediction framework are Dynamic Workload Generator, Model Generator, and Simulation Platform. The Dynamic Workload Generator calculates particle workload based on an input particle trace and mapping algorithm specified in configuration file. The Model Generator generates the performance models for the most expensive kernels in the PIC application based on the training data. Finally, the generated performance models and workload are inputted into a system-level Simulation Platform to predict application performance on a target system.

The details of our work on the Model Generator [13] and a coarse-grained Simulation Platform [7] have been published elsewhere. The main focus of this paper is on the Dynamic Workload Generator. The Dynamic Workload Generator accurately generates the workload on each processor of a target system. It provides multiple advantages. First, generating particle workload using a particle trace is computationally much cheaper than profiling the application on a real system. For example, while running a Hele-Shaw simulation on CMT-nek, as described in Section IV-A, it required less than two minutes to generate the particle workload on target system containing 4176 processors, whereas collecting the same information by running the entire application would take close to 24 hours. Also, as the particle movement is independent of the system configuration, a single application trace is sufficient to predict the workload on any given number of processors. Therefore, by quickly generating the workload on any given number of processors, we can study how the workload scales on larger number of processors, helping us identify any scalability bottlenecks. Using the Dynamic Workload Generator, we were able to predict the workload distribution across different processor configurations and perform detailed performance analysis of CMT-nek application, as shown in Section IV.

A. Dynamic Workload Generator

A particle workload is expressed primarily in terms of the number of actual particles (N_p) and ghost particles (N_{gp}) residing in a processor. Ghost particles are the particles which are not present in the processor domain but their influence is felt on the grid points local to the processor. A particle workload is classified into computation and communication load. Computation load is determined by the total number of real and ghost particles residing in a processor. Communication load is determined by the total number of real and ghost particles crossing processor domain during runtime.

Figure 3 shows the components and the workflow of the Dynamic Workload Generator module. The two key inputs to the Dynamic Workload Generator are the particle trace file and the configuration file. As mentioned above, a particle trace contains the coordinates of particle location at specified intervals. The configuration file contains information of system and application configuration:



Dynamic Workload Generator
Particles: 100,000
Elements: 10,000
Processors: 16,384
Mapping algorithm: bin-based
Configuration file
Particle trace file
Computation load generator
Communication load generator
Computational matrix
Particle workload per processor
Communication matrix
Particle data communicated across processors
T
R
R

Particle ID	X	Y	Z	timestep
1	0.00999	0.0013	-0.00136	100
2	0.00997	0.00141	-0.00147	100
3	0.01	0.00126	-0.00152	100
4	0.00999	0.00129	-0.00187	100
5	0.00997	0.00141	-0.00177	100
6	0.00998	0.00137	-7.00E-05	100
7	0.00998	0.00137	-0.00034	100
8	0.00998	0.00135	-0.00091	100

Fig. 3: Dynamic Workload Generator module.

(i) System configuration: number of processors (R).

(ii) Application configuration: number of particles (N_p), number of spectral elements (N_{el}), grid dimensions (N), particle mapping algorithm used in the application, and other problem parameters affecting performance.

To generate the particle workload, the Dynamic Workload Generator calculates (i) the number of real and ghost particles residing on each processor and (ii) number of real and ghost particles moving across processor domain between consecutive intervals. Let R_p be the residing processor which stores the corresponding particle data. In the Dynamic Workload Generator, we calculate the value of R_p for each particle and update the particle counter for that processor. A separate particle counter is maintained for real and ghost particles.

As shown in Figure 3, the Dynamic Workload Generator has two main components. The Computation Load Generator calculates the residing processor for each particle by mimicking the logic used by the particle mapping algorithm to distribute particles to the processors. For example, in case of element-based mapping for CMT-nek, as discussed in Section III-B, particles are assigned to processor which store the corresponding element. Hence, in the Computation Load Generator, to calculate the workload distribution based on element-based mapping, we first calculate in which element the particle resides. This is calculated based on particle location specified in trace file and PIC domain information specified in the configuration file. Upon calculating the residing element for each particle, we identify the processor which stores the element and update its particle counter. Similarly, for other mapping algorithms, we calculate which processor stores each particle by mimicking the logic used in the corresponding mapping algorithm. As most particle mapping algorithms make use of particle location to distribute particles across processors, the particle trace and configuration file provide sufficient information to generate particle workload for these particle mapping algorithms on any given processor configuration.

The Communication Load Generator calculates the number of particles being communicated across two processors at a given sampling interval. A communication load is calculated by checking if the residing processor (R_p) of a particle is changed between two consecutive intervals. If the value of R_p changes between two consecutive intervals, it means that the particle has moved from one processor to another.

The two outputs of Dynamic Workload Generator are the

Computation matrix (P_{comp}) and the Communication matrix (P_{comm}). The Computation matrix is a two-dimensional array of size $R \times T$ where R is the processor count and T is the total number of samples. The Computation matrix describes the computation load across all the processors throughout the PIC simulation. For example, $P_{comp}[i][j]$ denotes the number of particles residing in processor i at j^{th} interval. Figure 1(a) shows the visual representation of the Computation matrix in form of a heat-map. The Communication matrix is a three dimensional array of size $R \times R \times T$. It quantifies the particle communication across different processors between two consecutive intervals and describes the amount of particle data transfer across processors throughout the execution. For example, $P_{comp}[i][j][k]$ denotes the number of particles moving from processor i to processor j at k^{th} interval. Each particle has a specific amount of data associated with it. By calculating the number of particles moving across the two processors, we can calculate the message size. The Dynamic Workload Generator outputs separate computational and communication matrices for real and ghost particles.

Currently, our Dynamic Workload Generator can generate workload distributions for element-based and bin-based mapping algorithms for CMT-nek. It can be extended to incorporate other particle mapping algorithms. By doing so, it provides a fast and accurate method in identifying the optimal mapping strategy for a given problem specification. By generating the workload distribution across multiple processors for different mapping algorithms, we can evaluate the ideal mapping strategy for optimal performance. In Section IV-C, we evaluate the performance of the two mapping algorithms used to distribute particles in CMT-nek. Using the Dynamic Workload Generator, we predicted and subsequently validated that using bin-based mapping approach reduced the peak particle workload by two orders of magnitude. Furthermore, Resource Utilization (RU), which is determined by the number of processors having at least one or more particles on average during the simulation, increased from 0.68% to 56.13% upon using bin-based mapping.

B. Model Generator

To build performance models, we instrument the source code and benchmark key computation kernels of PIC application for various input parameter combinations. Upon generating the training data from benchmarking, it is fed to Model Generator to build analytical performance models. The framework supports multiple regression methods for model generation. While simple linear regression methods were sufficient to generate single parameter performance models with reasonable accuracy, they failed to generate accurate multi-parameter performance models owing the model complexity. To solve this problem, we developed a fast and accurate multi-parameter modeling approach using symbolic regression [13]. By leveraging genetic programming [14], we were able to automatically discover an underlying multi-parameter model that captures difficult-to-understand behaviors.

The regression methods used to generate performance models for our case-study application CMT-nek are discussed in Section IV-A. The generated analytical performance models are represented in terms of workload parameters, such as number of particles per processor (N_p), number of elements per processor (N_{el}) etc. Our generated CMT-nek performance models had an average MAPE of 8.42%.

C. Simulation Platform

The Simulation Platform takes the dynamic workload and the performance models as inputs along with the target system specification to perform system-level simulations. By feeding the dynamic workload into the Simulation Platform, we simulate actual behavior of the target system by advancing simulation clock of each processor based on their individual workload. The amount by which simulation clock is advanced is calculated by inserting the workload parameter values into performance models. The Simulation Platform performs system-level simulations to predict application performance.

The current version of our system-level simulation platform, BE-SST [7], is a coarse-grained simulator integrated into the parallel discrete-event simulation framework called Structural Simulation Toolkit (SST) from Sandia National Laboratories [6]. BE-SST can be used to perform rapid design-space exploration to reduce a large design space into promising candidates for detailed simulation. In [7], large-scale system-level studies were performed to demonstrate its capabilities on two HPC systems - Vulcan [9] from Lawrence Livermore National Laboratory and Titan [15] from Oak Ridge National Laboratory. Simulation results were validated against testbed measurements up to 128k cores, and blind performance predictions were made up to 1 million cores. However, BE-SST currently does not support trace-based simulations. This capability is being added to BE-SST and will be available in the next version update. When completed, the performance prediction framework will perform trace-driven coarse-grained system-level simulations.

D. Advantages and Limitations

Our prediction framework provides multiple advantages in terms of performance evaluation and optimization. The key advantages of our framework are in (a) scalability prediction, (b) algorithm evaluation, and (c) performance tuning. As mentioned before, accurately generating dynamic workload is crucial in predicting performance of irregular applications on large-scale systems. Large amount of compute hours can be saved by analyzing how the application would scale for large systems without having to re-run multiple times. To perform scalability predictions, we take advantage of the fact that particle movement is independent on the processor count. Hence, a single application trace for a given problem specification is sufficient to predict the application behavior on any number of processors. As we show in Section IV-B, we performed a strong scaling prediction on Hele-Shaw case-study by increasing the processor count from 1044 to 8352 cores. Through the predictions, we evaluated that scaling the processor count beyond 1104 has no impact on particle-solver performance. Apart from evaluating scalability performance, the dynamic workload generated can be used to calculate resource utilization, quantifying load imbalance.

Another advantage of our prediction framework is algorithm evaluation, evaluating different particle mapping strategies for a given simulation problem. While different load-balancing strategies [10]–[12], [16]–[18] have been proposed for PIC algorithms, there is no one single mapping strategy which is optimal for all problem configurations. Our prediction framework provides two benefits in this regard. First, the framework provides a test-bed for quick evaluation of any new mapping strategy without having to spend large amounts of code-development time for a scalable parallel implementation. Second, if multiple particle mapping strategies are readily available, as in the case of CMT-nek, the framework helps pick the optimal mapping strategy suited for a given problem specification by quickly generating the workload distribution for different mapping strategies. This would be largely beneficial while performing large-scale runs (hero-runs), where a single order of performance improvement would save hundreds of compute hours. In Section IV-C, we evaluate the efficiency of two particle mapping algorithms available in CMT-nek on Hele-Shaw simulation case-study.

Finally, our prediction framework helps in optimizing performance through parameter tuning. Certain problem parameters have a significant impact in application performance in a derived way. In CMT-nek, one such parameter is projection filter size. Projection filter size defines the spread of particle influence on the neighboring grid size. Projection filter size has a significant effect on the total number of ghost particles created across processors. Application developers and users would be in a better position to fine tune the parameter value if they can evaluate the performance impact. Our prediction framework facilitates such a parameter study by providing performance cost for the values in the given parameter range. In Section IV-D, we perform a parameter study on projection filter size and evaluate its impact on application performance.

One of the key limitations of our approach is in the trace collection phase. Trace collection is often expensive and at times infeasible. To obtain particle trace, the application needs to be executed once, which may not be ideal, especially while predicting performance of a large-scale simulation. Also, in some cases, a trace file can be expensively large. The size of the trace file is proportional to total numbers of particles in simulation and sampling frequency. In case of large-scale PIC simulations, which usually contain millions of particles and executed for over million time-steps, a trace file would be hundreds of Gigabytes large and would require large amount of compute hours to obtain. One solution for this problem is to generate a synthetic trace from a small-scale run which is relatively inexpensive to collect. However, generating a representative large-scale trace from a small-scale trace is a complex task. We would be exploring this in the future but currently is not in the scope of the paper. Another challenge in trace collection is selecting the sampling frequency. A low sampling frequency would reduce the file size, but would not accurately capture particle movement. High sampling frequency, on the other hand, would capture fine particle movement at the cost of larger file size. In practice, we use application user's intuition to fine tune sampling frequency.

III. CASE-STUDY

A. *CMT-nek*

CMT-nek is a proposed Navier-stokes solver aimed at solving compressible multiphase turbulent flows. It is developed at the DOE PSAAP-II (Predictive Science Academic Alliance Program) Center for Compressible Multi-phase Turbulence (CCMT). CMT-nek, built on top of Nek5000 [19], a highly-scalable computational fuild dynamics (CFD) solver, is primarily used to simulate particle-laden explosively dispersed flows under high pressure and temperature. CMT-nek employs a Particle-In-Cell method for solving particle properties. Zwick et al. [12] provides more details about CMT-nek.

The computational domain of CMT-nek consists of both spectral elements and particles. Each spectral element is further decomposed into grid points. The solver structure of CMT-nek is categorised into two key phases, namely fluid-solver and particle-solver. The fluid-solver phase solves the Euler equations of gas dynamics at each grid-point.

$$\frac{d\boldsymbol{X}}{dt} = \boldsymbol{V} \tag{1}$$

$$M_p \frac{d\boldsymbol{V}}{dt} = \boldsymbol{F}_h + \boldsymbol{F}_c + \boldsymbol{F}_b \tag{2}$$

The particle solver kernel tracks the particle evolution on the computation grid by solving the governing equations of motion and conservation of momentum as described in Equations 1 and 2. The particle solver kernel performs a set of operations at each iteration, commonly known as PIC Solver loop as shown below:

1. Interpolation (Grid-Particle interaction): Calculate the Eulerian fluid forces acting upon particles by interpolating the fluid properties from neighboring grid points.

2. Equation solver: Upon calculating the fluid forces on particles, solve the governing equation of conservation of momentum, specified in 2 to calculate particle velocity.

3. Particle pusher: Upon calculating new particle velocity, advance the particle position to the next time-step based on 1.

4. Projection (Particle-Grid interaction): Project the newly calculated Lagrangian particle properties onto the neighboring Eulerian grid-points.

In the case of the CMT-nek particle solver kernel, collision force is also accounted in addition to fluid forces to solve Equation 2. These additional forces are calculated based on particle-particle collisions.

The CMT-nek computational workload can be classified into fluid workload and particle workload. Fluid workload is primarily represented in terms of the number of elements per processor (N_{el}) and grid resolution within an element (N). Particle workload is primarily represented in terms of the

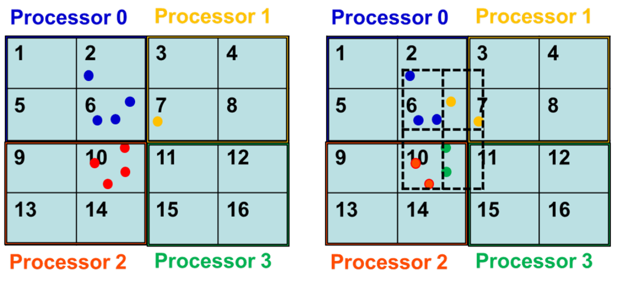


Fig. 4: Particle decomposition of a two-dimensional computational grid containing 16 spectral elements and 9 particles using (a) Element-based mapping and (b) Bin-based mapping.

number of real (N_p) and ghost (N_{gp}) particles residing in a processor. Different decomposition strategies are available to distribute spectral elements and particles onto the processors. CMT-nek uses recursive-bisection algorithm [20] to distribute spectral elements onto the processor in a way to minimize grid-data exchange across processors. Most particle mapping strategies use spatial locality to map particles to the processors owing to the tight particle-grid interaction. Efficiency of different particle mapping strategies relies on how effectively particles are distributed across processors in order to ensure close to uniform workload distribution throughout the application run time. In the following subsections, we discuss two particle mapping strategies used in CMT-nek. In IV-C, we evaluate the efficiency of these mapping strategies on a Hele-Shaw case study containing 216,225 spectral elements and 599,257 particles running on Quartz.

B. *Element-based mapping*

One simple and intuitive way to map particles to processors is to ensure both the particle and the spectral element in which it is residing are stored in same processor. By ensuring such particle-grid locality, all the fluid-particle interactions are computed locally per processor, thereby minimizing inter-processor communication. Such element-based mapping is the de facto standard for most PIC algorithms. However, as the particles start moving across the grid, the element in which the particle is residing changes, which may result in particles crossing processor domain. In such cases, data associated with the particle has to be transferred to new processor containing the corresponding element. Also, if initial particle distribution is non-homogeneous, a valid scenario in most explosive turbulent simulations simulated in CMT-nek, processors experience irregular workloads right from the beginning.

Figure 4(a) illustrates an example of element-based mapping on a two-dimensional computational grid containing 16 spectral elements and 9 particles. The computational grid is decomposed into four processors. As shown in the left side of the figure, element-based mapping algorithm maps the particles to processors based on residing element. While this provides an advantage of localizing particle-grid computations, it suffers from load imbalance as shown in figure 4(a). Here, processor 0 and processor 2 have four particles each where processor 1 has only 1 particle and processor 3 has none.

C. *Bin-based mapping*

In order to solve the load-imbalance problem, CMT-nek developers designed a bin-based mapping algorithm. In this approach, particle-grid locality is decoupled and the particle domain is partitioned into multiple bins which are uniformly distributed across processors. Bin-based mapping ensures close to optimal particle distribution. However, as particle-grid locality is not preserved, additional work is needed to perform particle-grid computation. This involves transferring associated grid data between the processors. Also, as the particles move continuously, particle domain expands and shrinks. Hence, particle domain partition and grid-data transfer has to be performed at every iteration. Zwick et al. [12] provides a detailed description of the bin-based mapping algorithm.

Figure 4(b) shows how particles are mapped based on bin-based algorithm on a two-dimensional grid mentioned above. Initially, a particle domain is identified by generating a particle boundary, as shown with dotted lines. Later the domain is partitioned using recursive planar cut algorithm. Recursive bin partition terminates when either the bin size reaches a threshold value or the number of bins are equal to the number of processors. Each bin is then assigned to a processor. As shown in the figure, using a bin-based mapping approach, processor 0 receives three particles whereas rest of the processors receive 2 particles each, showing a much better workload distribution. In Section IV-C, we evaluate and predict the efficiency of these two approaches.

IV. PERFORMANCE PREDICTIONS

A. *Experimental Setup*

We evaluate our performance prediction framework on Hele-Shaw simulation case-study. In Hele-Shaw simulation, a large number of particles initially are packed at the bottom of a three dimensional cylinder specified by the computational domain. Beneath the particle bed, there is a high pressurized gas separated by a diaphragm. At the beginning of simulation, the diaphragm bursts, releasing a high pressurized gas into the particle bed, causing a shock wave. Consequently, the particles are displaced due to the shock loading. Koneru et al. [21] provides more details on the Hele-Shaw simulation.

To generate CMT-nek performance models, we instrumented the application and benchmarked for multiple parameter combinations to collect the training data. To generate single-parameter performance models, we used linear regression. Multi-parameter performance models are generated using symbolic regression. We collected particle trace by sampling particle location for every 100 iterations. The trace is obtained by running CMT-nek on 1024 processors. We generated workload for element-based and bin-based algorithms. We used Quartz as the target system for benchmarking and trace collection. Quartz [22] is a supercomputer from Lawrence Livermore National Laboratory(LLNL). Quartz contains 3018

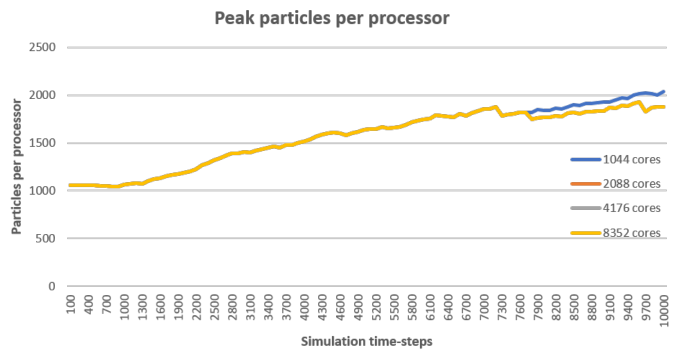


Fig. 5: Maximum number of particles per processor for different processor configurations.

Intel Xeon E5 nodes connected through Intel OmniPath with a peak performance of 3,200 TFLOP/s.

In subsequent sections, we demonstrate how our performance prediction framework can be used for scalability prediction, algorithm evaluation, and performance tuning.

B. Scalability Prediction

Figure 5 shows the prediction of the maximum number of particles in a processor during the entire simulation for different processor configurations, as outputted from our Dynamic Workflow Generator. The processor with largest workload represents the critical path in the system execution. It is interesting to note that the workload does not scale while increasing the processor count. For the first 7800 iterations, the peak particle workload per processor remains same while running the application on 1044, 2088, 4176, and 8352 processors. Upon closer inspection, we found out that it is due to bin size threshold. As mentioned in Section III-C, in bin-based mapping approach, particle domain is recursively decomposed into bins until it reaches either a threshold bin-size or the number of bins equal the processor count. In our problem case-study, the threshold bin-size is reached at a maximum of 1012 bins during the first 7800 iterations. Hence the bins are distributed to at most 1012 processors. As a result, increasing the processor count would not improve workload distribution. Using our Dynamic Workflow Generator, we were able to predict the workload distribution on different processor counts without actually having to physically run the code. Such predictions would be highly useful in identifying an optimal processor count for a given problem configuration, thereby optimizing resource allocation. Note that we also have validated our predictions shown in Figure 5 by comparing the output of our Dynamic Workload Generator with actual workload, obtained by running the Hele-Shaw simulation case-study on 1044, 2088, 4176 and 8352 processors.

Figure 5 also shows that after the first 7800 iterations, there is a dip in the peak processor workload when using more than 1044 processors. In Hele-Shaw simulation, particles move from the bottom of cylinder due to shock loading, and a result, particle boundary expands during the course of execution. As

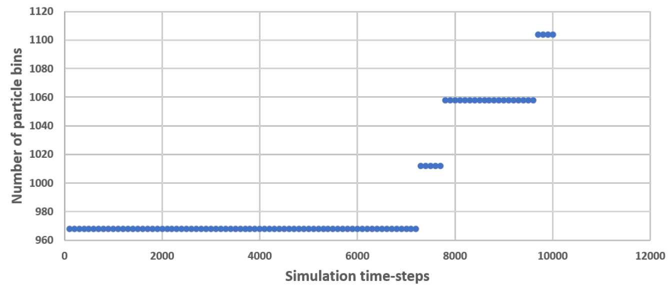


Fig. 6: Particle bins generated during the application run. Number of particle bins increase during the simulation as the particle boundary expands.

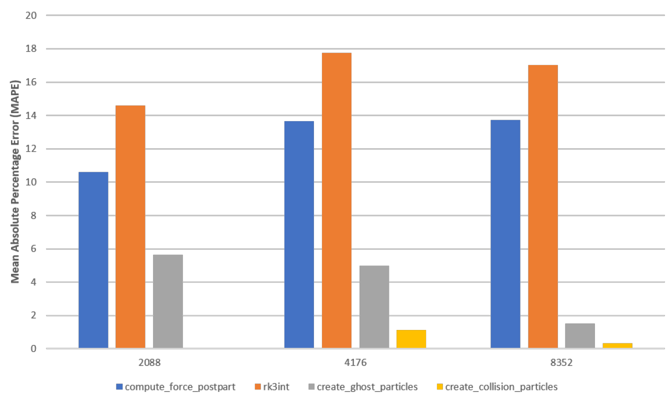


Fig. 7: Mean Absolute Percentage Error (MAPE) of key kernels of CMT-nek for different processor configurations

the particle boundary becomes bigger, it can be decomposed into further bins before reaching the threshold bin-size.

Figure 6 shows the maximum number of particle bins generated before reaching threshold bin size. Using our Dynamic Workflow Generator, we have relaxed the processor count limitation, thereby calculating the maximum number of bins that can be generated irrespective of system configuration. This provides an upper limit on the processor count to obtain optimal workload distribution for bin-based mapping algorithm. As shown in the figure, the maximum number of bins generated throughout the simulation run is 1104. Hence, optimal processor count for given problem configuration is 1104. As a result, you see a dip in peak particle workload in figure 5, when increasing the processor count from 1044 to 2088. However, subsequent increase in processor count has no effect on workload distribution due to bin-size threshold.

Currently our coarse-grained simulation platform, BE-SST does not support trace-based simulations. Trace-based simulation capability is currently being added to the simulator and will be available in the next version. In order to demonstrate the accuracy of our performance models, we developed a python script which takes the generated performance models and the output of workload generator as inputs, and predicts the kernel performance across all processors during the entire execution. Figure 7 shows our prediction accuracy of key

CMT-nek kernels on different processor counts. On average, our performance models had an 8.42% MAPE error with a peak error of 17.7%. Figure 7 is a representative result of what it would be if we use a system-level simulator to perform end-to-end performance prediction.

Note that element workload, i.e., the number of spectral elements per processor (N_{el}), is scaled uniformly. Hence, even though there is no change in the particle workload when increasing the processors from 2088 to 4176 and 8352 processors, element workload per processor is halved every time the processor count is doubled. As the irregular workload is generated due to particles, we would be considering the impact of particle workload alone in this paper.

C. Algorithm evaluation

One of the key advantages of our prediction framework is to evaluate the efficiency of different mapping algorithms without having to run the application. Figure 8 shows peak particle workload distribution of Hele-Shaw simulation on two different particle mapping strategies. Bin-based mapping algorithm has a better workload distribution, in fact a couple of orders reduction in peak particle workload compared to element-based mapping. As mentioned before, in element-based mapping, particles are assigned to processor based on the residing element. In Hele-Shaw simulation, large number of the particles initially reside in a relatively smaller area in the computational domain. Hence, only the processors assigned to these elements has the majority of particles as shown in the figure. As the processor count is increased, the elements containing the majority of particles are distributed to other processors resulting in reduced peak particle workload. However, due to this tight particle-element coupling, a small number of processors share the entire particle workload. Figure 9 shows the percentage of processors containing at least one particle per processor during the entire simulation. Bin-based mapping algorithm results in 584 processors sharing the particle workload out of 1044 processors, resulting in 56% processor utilization. In comparison, only 4 processors in element-based approach share the entire particle workload, resulting in 0.68% of processor utilization. In general, bin-based mapping algorithm does a better job in mapping the particle workload onto the processors.

D. Performance tuning

Certain parameters have a significant impact on application performance in a derived way. In CMT-nek, one such parameter is projection filter size. Projection filter size denotes the spread of particle influence on the neighboring grid. It determines the total number of ghost particles per processor. Projection filter is also used as the threshold bin-size for particle bin generation. Figure 10(a) shows the maximum number of particle bins generated for different values of projection filter. Smaller values of projection filter ensures lower threshold, which results in generating more bins. On the other hand, higher filter values results in more ghost particles per processor as the particle influence is spread much further. Figure 10(b) shows the execution time of one of CMT-nek kernels, *create_ghost_particles*, which generates ghost particles at each processor. As expected, there is a significant increase in the execution time for larger filter sizes. By predicting the impact of application performance with respect to various projection filter sizes, the application developers would be in a better position to evaluate the trade-off between simulation accuracy and application performance.

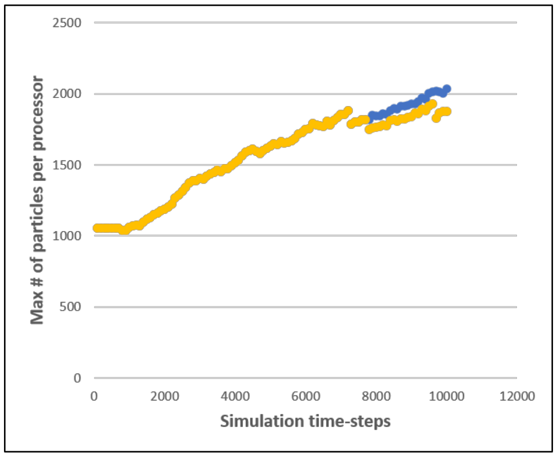
Note that varying the projection filter-size may effect the particle movement. In most mapping algorithms, such changes in particle movement are minor and does not significantly affect processor workload.

V. RELATED RESEARCH

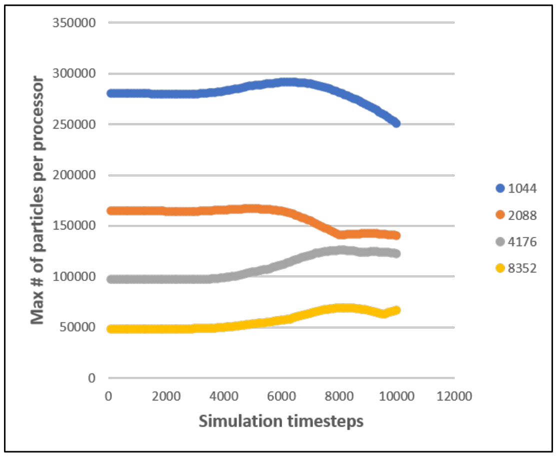
Trace-driven simulation approaches are widely used for system-level predictions when the application behavior cannot be comprehensibly described using analytical models. In this section, we briefly discuss different trace-based simulation approaches for performance prediction of large-scale systems. We also discuss about different load-balancing algorithms proposed to optimize workload distribution in PIC applications.

Scalable workload distribution in PIC applications is a complex task. While grid-based decomposition results in computational load-imbalance due to moving particles, particle-based decomposition causes communication and memory overheads in order to store and synchronize redundant grid data across processors. Many particle-based decomposition strategies have been proposed for scalable workload decomposition of PIC applications on large-scale systems. Zwick et al. [12] proposes a particle decomposition based on binning. Particle bins are constructed by recursive planar cutting of particle domain at each iteration. Zhai et al. [11] presents a load-balanced partitioning strategy where the computational load is calculated based on the number of grid points and particles residing on each processor. Particle-grid locality is maintained and the elements are distributed in a way to ensure all processors share similar computational load. Re-partitioning occurs once a processor exceeds a threshold workload. Liao et al. [10] uses two different partition strategies for grid and particle distribution. For particle distribution, the author assigns a unique global number to the particles based on hilbert ordering of spectral elements. Upon generating the indices, particles are distributed based on the increasing order of global number to preserve the particle-grid locality, while optimizing the workload distribution. Our work does not propose a new particle mapping algorithm but rather provides a framework to quickly generate workload distribution for various particle mapping algorithms using a low-cost implementation.

Trace-based simulations are primarily used when application behavior cannot be completely captured in analytical models. In such scenarios, a trace is collected, either from an application run or through synthetic execution, in order to accurately capture the application performance on large-scale systems [23], [24]. Carothers et al. [23] generates scalable synthetic workloads from real applications. The workload is fed into a trace-driven simulation environment, CODES, along



(a)



(b)

Fig. 8: Peak particle workload distribution of Hele-Shaw simulation on different processor configurations using (a) bin-based and (b) element-based mapping algorithms.

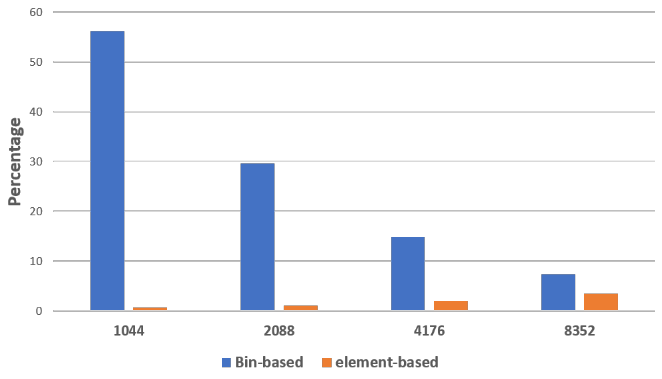


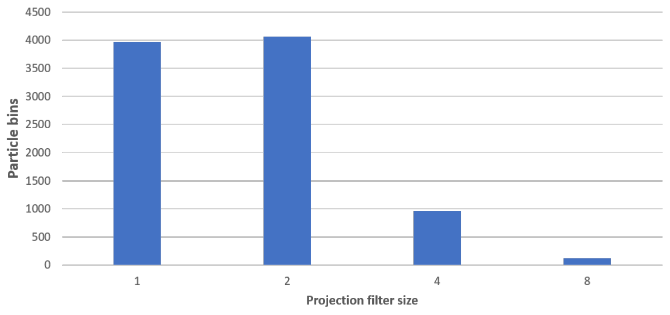
Fig. 9: Processor utilization depicting the percentage of processors containing non-zero particle workload

with the application description specified in an analytical performance modeling language [25]. Our approach is similar in that both use a workload generator and analytical performance models to perform accurate large-scale performance predictions. However, our workload generator is customized to PIC workloads. Many synthetic workload generation approaches have been proposed to calculate the application workload on large-scale HPC systems [26]–[28]. However, most of these approaches calculate the communication load by detecting and subsequently scaling up the communication pattern on target systems. These approaches assumes static distribution of computational workload across processors while scaling synthetic workload, which is inaccurate for irregular applications such as PIC. To the best of our knowledge, our proposed Dynamic Workload Generator is the first in generating accurate workload distribution of PIC applications on large-scale systems.

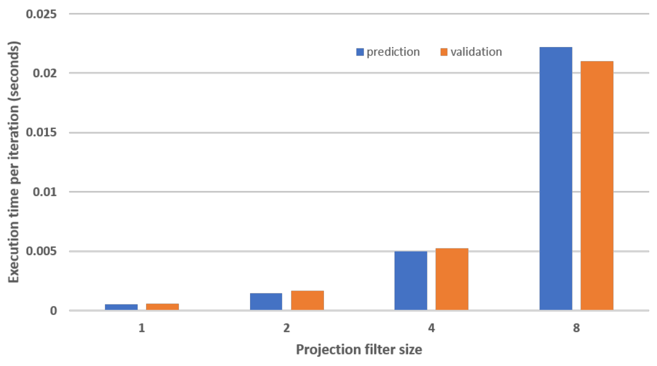
VI. CONCLUSIONS

Scalable performance prediction of large-scale applications on HPC systems is crucial to accelerate the application time to solution and to optimize the system resources. In order to predict the performance, it is crucial to capture the application workload and behavior in terms of workload distribution and performance models, respectively. Irregular applications, such as those which employs the PIC method, are notorious for their dynamic workload distribution across the processors due to constantly moving particles. In this paper, we propose a novel trace-driven performance prediction framework to reliably predict the application performance of PIC applications. The framework provides multiple advantages in terms of scalability prediction, algorithm evaluation, and performance tuning. For scalability prediction, our performance prediction framework demonstrated an average MAPE of 8.42% while predicting CMT-nek performance on Quartz. For a given problem case-study involving 599,257 particles and 216,225 elements, we are able to predict the ideal processor count to achieve optimal performance. For algorithm evaluation, we evaluated the impact of two different mapping algorithms, namely element-based and bin-based mapping, on application performance, demonstrating that using the latter approach would reduce the peak particle workload by two orders of magnitude. Finally, to demonstrate performance tuning, we performed a parameter study to quantify and validate the performance impact of a key problem parameter in the CMT-nek.

Going forward, we are in the process of adding trace-based simulation capability to our system-level simulation platform, BE-SST. This will allow us to demonstrate end-to-end capabilities of the performance prediction platform for evaluating applications with dynamic workloads. To overcome the challenge of expensive trace collection, we are working on incorporating trace extrapolation feature to our Dynamic Workload Generator to generate representative high-scale particle trace from a low-fidelity execution. This will help in reducing the trace collection cost, especially for large-scale application runs which usually involve a billion particles and millions of spectral elements. Finally, we will be adding addi-



(a)



(b)

Fig. 10: (a) Maximum number of particle bins generated for different project filter sizes. Total number of bins provides an upper limit on particle distribution (b) Execution time of *create_ghost_particles* kernel for different projection filter sizes.

tional particle mapping algorithms to our Dynamic Workload Generator so application users can evaluate and select an optimal mapping strategy for their application needs.

References

[1] A. Calotoiu *et al.*, "Fast multi-parameter performance modeling," in *2016 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2016, pp. 172–181.

[2] E. Ipek, B. R. De Supinski, M. Schulz, and S. A. McKee, "An approach to performance prediction for parallel applications," in *European Conference on Parallel Processing*. Springer, 2005, pp. 196–205.

[3] B. C. Lee *et al.*, "Methods of inference and learning for performance modeling of parallel applications," in *Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM, 2007, pp. 249–258.

[4] G. Bauer, S. Gottlieb, and T. Hoefler, "Performance modeling and comparative analysis of the milc lattice qcd application su3_rmd," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*. IEEE, 2012, pp. 652–659.

[5] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. De Supinski, and M. Schulz, "A regression-based approach to scalability prediction," in *Proceedings of the 22nd annual international conference on Supercomputing*. ACM, 2008, pp. 368–377.

[6] A. F. Rodrigues *et al.*, "The structural simulation toolkit," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 4, pp. 37–42, 2011.

[7] A. Ramaswamy, N. Kumar, A. Neelakantan, H. Lam, and G. Stitt, "Scalable behavioral emulation of extreme-scale systems using structural simulation toolkit," in *Proceedings of the 47th International Conference on Parallel Processing*. ACM, 2018, p. 17.

[8] C. D. Carothers, D. Bauer, and S. Pearce, "Ross: A high-performance, low-memory, modular time warp system," *Journal of Parallel and Distributed Computing*, vol. 62, no. 11, pp. 1648–1669, 2002.

[9] Vulcan. https://computation.llnl.gov/computers/vulcan.

[10] W.-k. Liao, C.-w. Ou, and S. Ranka, "Dynamic alignment and distribution of irregularly coupled data arrays for scalable parallelization of particle-in-cell problems," in *Proceedings of International Conference on Parallel Processing*. IEEE, 1996, pp. 57–61.

[11] K. Zhai, T. Banerjee, D. Zwick, J. Hackl, and S. Ranka, "Dynamic load balancing for compressible multiphase turbulence," in *Proceedings of the 2018 International Conference on Supercomputing*, 2018, pp. 318–327.

[12] D. Zwick and S. Balachandar, "A scalable euler–lagrange approach for multiphase flow simulation on spectral elements," *The International Journal of High Performance Computing Applications*, vol. 34, no. 3, pp. 316–339, 2020.

[13] S. P. Chenna *et al.*, "Multi-parameter performance modeling using symbolic regression," in *2019 International Conference on High Performance Computing Simulation (HPCS)*, 2019, pp. 312–321.

[14] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA, USA: MIT Press, 1994.

[15] Titan. https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/.

[16] H. Nakashima *et al.*, "Ohhelp: a scalable domain-decomposing dynamic load balancing for particle-in-cell simulations," in *Proceedings of the 23rd international conference on Supercomputing*, 2009, pp. 90–99.

[17] F. Wolfheimer, E. Gjonaj, and T. Weiland, "A parallel 3d particle-in-cell code with dynamic load balancing," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 558, no. 1, pp. 202–204, 2006.

[18] D. B. Seidel *et al.*, "Dynamic load-balancing for a parallel electromagnetic particle-in-cell code," in *PPPS-2001 Pulsed Power Plasma Science 2001. 28th IEEE International Conference on Plasma Science and 13th IEEE International Pulsed Power Conference. Digest of Papers (Cat. No. 01CH37251)*, vol. 2. IEEE, 2001, pp. 1000–1003.

[19] P. F. Fischer, J. W. Lottes, and S. G. Kerkemeier. (2008) Nek5000. http://nek5000.mcs.anl.gov.

[20] S.-H. Hsieh *et al.*, "Recursive spectral algorithms for automatic domain partitioning in parallel finite element analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 121, no. 1-4, pp. 137–162, 1995.

[21] R. B. Koneru *et al.*, "A numerical study of particle jetting in a dense particle bed driven by an air-blast," *Physics of Fluids*, vol. 32, no. 9, p. 093301, 2020.

[22] Quartz. https://hpc.llnl.gov/hardware/platforms/Quartz.

[23] C. D. Carothers *et al.*, "Durango: Scalable synthetic workload generation for extreme-scale application performance modeling and simulation," in *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 2017, pp. 97–108.

[24] C. Minkenberg and G. Rodriguez, "Trace-driven co-simulation of high-performance computing systems using omnet++," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, 2009, pp. 1–8.

[25] K. L. Spafford and J. S. Vetter, "Aspen: a domain specific language for performance modeling," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 84.

[26] J. Chen and R. M. Clapp, "Astro: Auto-generation of synthetic traces using scaling pattern recognition for mpi workloads," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, pp. 2159–2171, 2017.

[27] X. Wu and F. Mueller, "Scalaextrap: Trace-based communication extrapolation for spmd programs," *ACM SIGPLAN Notices*, vol. 46, no. 8, pp. 113–122, 2011.

[28] L. Carrington, M. A. Laurenzano, and A. Tiwari, "Inferring large-scale computation behavior via trace extrapolation," in *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*. IEEE, 2013, pp. 1667–1674.