Performance Optimality or Reproducibility: That is the Question

Tapasya Patki, Jayaraman J. Thiagarajan
patki1,jayaramanthi1@llnl.gov
Lawrence Livermore National Laboratory

Alexis Ayala
ayalaa2@wwu.edu
Western Washington University

Tanzima Z. Islam
tanzima@txstate.edu
Texas State University

Abstract

The era of extremely heterogeneous supercomputing brings with itself the devil of increased performance variation and reduced reproducibility. There is a lack of understanding in the HPC community on how the simultaneous consideration of network traffic, power limits, concurrency tuning, and interference from other jobs impacts application performance.

In this paper, we design a methodology that allows both HPC users and system administrators to understand the trade-off space between optimal and reproducible performance. We present a first-of-its-kind dataset that simultaneously varies multiple system- and user-level parameters on a production cluster, and introduce a new metric, called the *desirability score*, which enables comparison across different system configurations. We develop a novel, model-agnostic machine learning methodology based on the graph signal theory for comparing the influence of parameters on application predictability, and using a new visualization technique, make practical suggestions for best practices for multi-objective HPC environments.

CCS Concepts

• **Computing methodologies** → **Parallel computing methodologies, Machine learning, Model development and analysis**; • **General and reference** → Performance.

Keywords

Performance reproducibility, machine learning, graph signal analysis, visualization

ACM Reference Format:

Tapasya Patki, Jayaraman J. Thiagarajan, Alexis Ayala, and Tanzima Z. Islam. 2019. Performance Optimality or Reproducibility: That is the Question. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '19), November 17–22, 2019, Denver, CO, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3295500.3356217

1 Introduction

The path toward exascale supercomputing has presented the HPC community with several new requirements that make performance optimization challenging. These include, but are not limited to, adhering to system power budgets, minimizing network interference, ensuring resiliency, and managing several heterogeneous devices along with large amounts of data and co-scheduled components. Simultaneously managing shared resources such as cores, memory, power, network and I/O while meeting such specified system requirements makes these environments *multi-objective*, often introducing conflicting optimization goals (for example, high system throughput versus high energy efficiency). Such conflicting optimization goals can result in increased performance variability, adversely affecting user experience.

Performance variability, which we define as the difference between execution times across repeated runs of an application in the same execution environment, is a common occurrence in HPC systems. For example, it has been shown that network contention and inter-job interference can lead to about 25% slower messaging rate in large-scale physics applications despite a fixed physical node mapping [8]. Similarly, other experiments have indicated that using techniques such as power capping can cause over 64% variation in per-rank execution time at scale, introducing load imbalance in an otherwise load-balanced application such as DGEMM [29]. Such run-to-run variability in application codes is expected to worsen when we operate future systems with more than one optimization objective, such as simultaneous management of power-limits, network interference, and temperature hotspots; or, when heterogeneous components such as GPUs, FPGAs, or burst buffers are involved. Notions of optimal performance in such multi-objective environments can be unclear, which complicates the understanding of application behavior further. Traditionally, optimal performance is defined in terms of the lowest execution time achieved or the maximum number of floating point operations possible. With high degrees of performance variation, however, such a definition may not necessarily convey the desirability of selected system settings in a multi-objective environment.

There exists a gap in understanding the *influence* of different user- and system-level parameters on application behavior that consider both fast execution time and low variability. While the impact of a single objective (such as power constraints or network traffic optimization) on application execution times has been studied, no research exists when it comes to understanding application behavior under two or more such objectives—which is the goal of this paper. We first present a unique, first-of-its-kind dataset that varies several power, network bandwidth, task placement, and concurrency parameters *simultaneously* across five benchmarks on a 324-node production system at Lawrence Livermore National Laboratory. With the help of this dataset, we conduct a detailed analysis of the influence of various parameters as well as the desirability of system configurations. Such a dataset and associated research is crucial for developing advanced adaptive resource management policies for minimizing variation in future systems, and in turn, for maximizing job throughput.

SC '19, November 17–22, 2019, Denver, CO, USA

ACM ISBN 978-1-4503-6229-0/19/11…$15.00
https://doi.org/10.1145/3295500.3356217

We define *performance reproducibility* in terms of minimal run-to-run variation in execution times of applications. While an ideal scenario is one where there is no variation in execution times, we note that such an ideal scenario is not practically feasible. Note that we do not consider numerical or bitwise reproducibility in this work. For performance optimality, we consider the *lowest* average per-task time, which allows us to compare performance both within the ranks of an application, as well as across various network, power, placement, and concurrency parameters.

The paper is organized in two parts. First, we define a new metric called *desirability score*, which allows us to quantify the trade-off space between performance optimality and reproducibility for system configurations. Second, we present a novel machine learning technique for analyzing the impact of changing user- and system-level parameters on performance predictability. This technique, inspired by graph signal analysis, is model-agnostic, and can be used to understand how reliable the output of *any* predictive model applied to our dataset is. By deriving an *influence score*, we identify which parameters are more significant than others in multi-objective environments. We present these influence scores using an *influence path diagram*, a novel visualization technique that we developed to facilitate comparison of feature importance.

Identifying parameters that impact performance variability can be posed as a feature selection process in machine learning, and most existing literature takes an exhaustive approach of exploring the entire space for each parameter to determine their impact [8]. Such approaches are not scalable. Furthermore, the accuracy of existing techniques for feature selection, such as recursive feature elimination, depends heavily on a-priori knowledge of the underlying data model, which can be difficult to obtain. We thus advocate the use of graph signal analysis because it does not require specific modeling assumptions unlike traditional multivariate analysis, making it more robust to noise in the data. In summary, the contributions of this paper are:

- A unique dataset, collected on a production system by changing the number of nodes, number of cores per node, network quality of service levels, power caps, and placements of application ranks across five benchmarks of interest in the presence of an interfering application. Such a dataset will serve as an example of data collection for future systems and enable reproducibility research in the community.
- Definition of the *desirability score*, a quantitative metric that enables comparison of different system configurations, including results on constrained configurations (such as network-limited or power-limited ones). This metric provides insights about expected variation in a given execution environment, and quantitatively explains the trade-off space between performance optimality and reproducibility.
- A novel, model-agnostic machine learning approach based on graph signal analysis that allows for comparison of the *influences* of various system- and user-level parameters on the predictability of application behavior; and a novel visualization technique based on path diagrams for the same.
- Best practice suggestions to achieve high reproducibility along with high performance for both HPC users and system administrators operating in multi-objective environments. To the best of our knowledge, this work is the first to provide a mechanism for co-designing an environment that is capable of addressing both optimality and reproducibility.

Table 1: Applications of Interest

Name	Description	Size and Characteristics	Scaling
LU	Lower-Upper Gauss-Seidel	Class D, Compute/comm.-bound	Strong
MG	Multi-Grid NAS	Class D, Memory-bound	Strong
FT	3D Fast Fourier Transform	Class D, Comm.-bound	Strong
Kripke	3D Neutron-transport	`nprocs: 8-16`, `zones: 128, 256` Compute-bound	Weak
CoMD	Classical Molecular Dynamics	`i,j,k: 8-16`, `x,y,z: 240-480`, Memory/comm.-bound	Weak

The rest of the paper is organized as follows: Section 2 presents a real-world motivating example that led to this research and drives the need for performance variability research in modern HPC environments. Section 3 presents a high-level view of our approach along with description of the dataset and the *desirability score*. Section 4 presents the details of the machine learning approach that computes the *influence score* of each of the parameters, and Section 5 describes our analysis, the system design questions they answer, and the observations in detail. Section 7 concludes the paper with idea about how our observations can be used for future work.

2 Real-world Example

This section illustrates the conundrum of performance optimality versus reproducibility, and motivates the need for the machine learning model as well as the visualization approach explained later in this paper. A large part of this research began as a result of a real problem that was encountered on the Catalyst cluster at Lawrence Livermore National Laboratory, which took over six months to resolve. Catalyst is a `150` TeraFLOPS, 324-node production capacity cluster. Each node in this cluster has two 12-core Intel Xeon CPUs and two HCA network adapters. It is supported by an InfiniBand QDR network with a two-level fat-tree topology, and has a layout of a total of `18` switches, with `18` nodes per switch.

In early 2017, some users that were conducting experiments on the Catalyst cluster observed severe run-to-run performance variability with up to `2-5x` slowdowns in their applications' execution time with the same execution environment. This led to user applications being killed prematurely by the resource manager as they exceeded their allocated time limits. Such concerns reported by users motivated the system administrators to collect data to determine the source(s) of such variation. Initial observations pointed to network traffic and inter-job interference issues, and this was verified with several `mpiGraph` [49] tests that allow for contention visualization. A few network switches were observed to be slower than expected,

Table 2: Infiniband Service Level Descriptions

Level	Communication Usage	Priority and Bandwidth
0	Application-level/MPI	87% of high-priority lane
1	Lustre	13% of high-priority lane
2	Not used	71% of low-priority lane
3	Not used	29% of low-priority lane

but upon further testing, hardware issues were ruled out. This initial analysis could not pin point the exact cause of observed variation. With the initial information from system administrators and after discussions with fellow researchers, we set out to conduct a more rigorous exploration of network-level variation in order to determine techniques to mitigate it. Our approach was to explore user-level knobs for network quality of service by analyzing InfiniBand Service Levels using *control jobs*. We defined a *control job* as an application whose performance (execution time) and reproducibility (repeatable execution times) we were interested in. In order to allow for emulation of real network traffic, we used a communication-bound interfering application in the background. Running a communication-bound interfering application assumes the worst-case scenario in a shared cluster environment – a scenario where multiple user applications are executing on the shared cluster and are all communicating and contending for network resources. This assumption enables us to collect data in a manner that allows studying the performance of the application of interest (that is, the control job) in a realistically emulated and worst-case shared cluster environment. We describe our selected applications, experimental setup and findings from this motivational study in the subsections below.

2.1 Representative Applications

We pick five representative applications as *control jobs*: LU, FT and MG from NAS Parallel Benchmark suite [4], and the Kripke and CoMD proxy applications [38, 53]. The details of these applications are presented in Table 1. We generate inter-job interference by using the OSU `MPI_Alltoall` benchmark [1] using 32KB messages that generates bursty communication at random intervals. All benchmarks are MPI-based, compiled with Intel compiler 16.0.3 and MVAPICH 2.2. Each representative application ran for at least 45 seconds (we increased iteration counts) and was repeated thrice to ensure reliable timing data. Note that strongly-scaled applications are ones where there is a fixed *total problem size* and the time to solution depends on the number of nodes used. On the other hand, weakly-scaled applications have a fixed *problem size per processor*, and using more nodes on such applications allows users to solve larger total problem sizes. We also pick two task placement algorithms: `packed` and `spread`. `Packed` places all application ranks as close together as possible on the network, so that the fewest number of switches are used from the fat tree. `Spread` distributes the ranks in a round-robin manner across the switches. The default is a `random` assignment of application ranks to physical nodes.

2.2 Network Quality of Service

In our cluster, four network quality of service levels associated with four virtual lanes are available as per the OpenSM configuration. These are detailed in Table 2. Two of these service levels are not used by default. The default setting is for all users to run their applications at the service level of `0`, and to have the Lustre file system [37] traffic use the service level of `1`. For our experiments, our goal was to first understand if users can utilize different virtual lanes and use service level knobs to reduce interference, mitigate variation and improve reproducibility. Thus, for controlling the network QoS, we use the `IPATH_SL` environment variable to select the virtual lanes [17] to send application traffic through a different lane than the default setting. Setting `IPATH_SL` to `0` corresponds to the application of interest having the highest bandwidth, and setting it to `3` corresponds to the lowest bandwidth.

2.3 Determining Sources of Network Variation

The first row of Figure 1 shows the initial data we collected. Our allocation comprised of 256 nodes of the 324-node cluster. Due to restrictions outside of our control, the remainder of the cluster (68 nodes) was left idle during our experiments. In this experiment, we ran 2048 application ranks across 86 nodes using all 24 cores on each node (except on the last node) in a 256-node allocation. The remaining nodes in the allocation (170 nodes) were running the interfering `MPI_Alltoall` benchmark to emulate a shared cluster. By default, both network adapters on the node were used.

Figure 1 compares the impact of using both the network adapters as opposed to picking one adapter on application performance (normalized execution time for each benchmark is shown on Y-axis). The X-axis shows the service level combinations used for the application followed by the interference—more bandwidth is allocated to the application (and less to the interfering application) as we progress from left to right on the X-axis in each of the graphs. As mentioned previously, the default on our system is `<0,0>`. For each configuration, we show multiple repeated runs with both `packed` and `spread` placements.

Our main counter-intuitive observation here is the significant amount of variation that we see when both adapters are used, which is the default on most HPC clusters. Even though using both adapters can lead to better (or, *optimal*) performance (it is roughly 6% faster on average than using a single adapter when looking at the minimum execution time in each set), it can severely impact reproducibility for our particular hardware with its default routing algorithm. Notice the scale of the Y-axis, we observed over 5x run-to-run variation in some cases, such as that of the FT benchmark.

It took over six months of active debugging to isolate this issue of adapters and routing. Setting `IPATH_UNIT` selects one of the two adapters, and once that option was used, we achieved significantly better reproducibility as shown in rows 2 and 3 of the figure. In this particular (yet limited) example, we had an issue that could be addressed through a static fix, and the sources of variation were not as complex. Future systems with power, network, and concurrency objectives are expected to be much more challenging with dynamic variation, and several opportunities for adaptive management exist. We believe users and system administrators should be able to request reproducible performance if needed. Additionally, a quantitative mechanism to compare the trade-off space between optimality and reproducibility should be developed. This led us to the realization that large scale collection and analysis data in a complex environment, involving network, power, and concurrency tuning

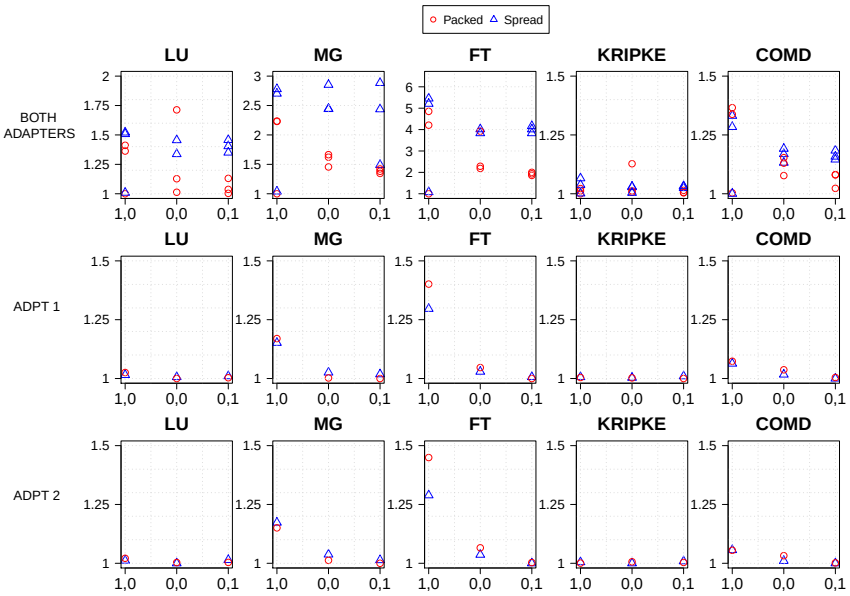


Figure 1: Impact of `IPATH_UNIT` on application execution times with 2048 tasks. Y-axis denotes normalized runtime, X-axis denotes service level combinations. Going from left to right on X-axis gives the application more bandwidth. The first row represents use of both network adapters, and the second and the third row represent the cases when the first or the second adapter is chosen. Packed and spread represent the application rank layout on the network.

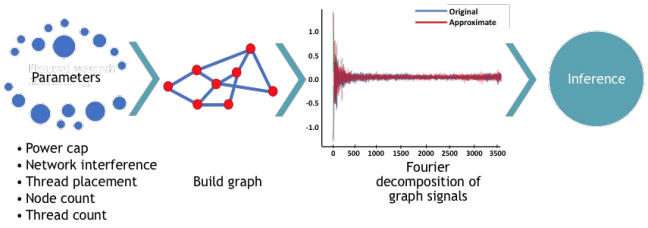


Figure 2: Overview of our approach

is critical for future resource management and adaptive systems. This led to us building the dataset and the graph signal processing based scoring mechanism that we present in the rest of this paper. Section 3 discusses how we approach this problem and design a workflow to ponder about such trade-offs.

3 Workflow

We now describe the overall methodology for identifying the impact of user- and system-level parameters on the average execution time of applications, both from optimality and reproducibility perspectives. In our work, we consider three system-level parameters based on network and power resources, and two user-level scaling parameters. We vary these parameters simultaneously to understand application behavior, identify desirable system configurations, and quantify parameter influence and importance. We follow a workflow of *data collection, analysis, and visualization*, as shown in Figure 2. We address the following key research questions:

- Which system-level and user-level parameters impact application behavior and what are their relative importance?
- Which system-level and user-level parameters improve or worsen performance predictability of applications?
- Given a system-level configuration, how much performance variation can a user expect across applications?
- Which system-level configurations are preferred compared to others?

Similar to Section 2, we vary tuneable parameters and collect the execution times of five different *control applications* with an interfering application running in the background. We view this problem as that of *influence analysis*, where we look at how greatly adding a parameter to an already selected set of parameters impacts the performance predictability of the applications. Additionally, we derive a new metric as a function of mean and variance that allows us to explore the optimality and reproducibility trade-offs and gauge the desirability of certain system settings. Note that we are not considering a certain modeling approach here, instead, we're analyzing the general predictability of a dataset. The goal of this work is to identify how reliable the output of *any* predictive model would be. The following subsections describe this in detail.

3.1 Data Collection

We first collect a one-of-its-kind dataset that looks at performance of applications at various scales, network bandwidth levels, placement styles, and power limits. As discussed previously, we distinguish between two types of configuration parameters in this paper—system- and user-level. For system-level, we consider task placement across switches (L), bandwidth-level (N), and power cap for the control job (P). For user-level parameters, we vary the number of node count (C) and the number of core or thread count (T).

Table 3: Description of configurations, optimality and reproducibility scores that will be used throughout the paper.

Config. Parameters	**System-level**	**Placement (L)**	`packed,spread,random`
		Bandwidth (N)	`1(low)-5(high)`
		Power cap (P)	`64W, 80W, 115W`
	Application-level	**Task count**	`512,1024,2048,4096`
		Core count	`24,20,16`
Optimality score		Minimum average per-task execution time	The lower, the better
Desirability score [Equation 1 in Section 3.2]		$\exp^{-mean \times variance}$	The higher, the better

Table 3 describes configuration parameters that we varied to collect data on the Catalyst system described in Section 2. The five representative applications and the interfering application used were described in Section 2. In addition to looking at task placement and InfiniBand service levels, we also collect data under power limits with varying levels of concurrency (by varying both number of nodes and cores per node). The former enables us to understand scenarios in future clusters where power may be constrained, and the latter allows us to look at varied memory intensity. Using all cores on a node could mean limited memory bandwidth for an application that is memory-intensive, and often running fewer cores per node in such scenarios can offer significant caching and memory bandwidth improvements [51, 63].

For power-capping, we use Intel's RAPL technology [18, 30] which is supported through `libmsr` [57, 65] and `msr-safe` in user-space on the Catalyst system. On our production clusters with supported Intel architectures, the `msr-safe` kernel module is deployed as part of the operating system. This kernel module allows for whitelisting of privileged RAPL registers as well as access to specific bits of RAPL registers through group permissions. This enables our users to both measure and control power from user space with libraries such as `libmsr`. Currently, several capacity clusters at other similar HPC sites with Intel architectures use `msr-safe` and expose such knobs in user space. Equivalent modules for IBM systems (Power8 and Power9) are available through the `OPAL` firmware [13] and can be set up in user space.

We vary the CPU power range between 64 W and 115 W (per socket, each node has two sockets). Here, 115 W corresponds to the scenario of peak or full power, 64 W corresponds to a scenario where power is extremely constrained, and 80 W corresponds to a scenario of medium power. We vary the number of tasks (MPI ranks) to be 512 tasks, 1024 tasks, 2048 tasks, and 4096 tasks. For each task count, we vary node-level concurrency by varying the number of cores per node from 16 through 24, in increments of 4. This in turn changes the total number of nodes that the application uses, and allows us to capture scaling and memory behavior in our dataset. Because there are 24 cores per node in the used system, the last node in the allocation for a fixed number of tasks may have fewer ranks. Note that this does not impact our performance numbers or analysis. For InfiniBand service levels, we use five combinations as opposed to the three described in Section 2. We also set `IPATH_UNIT` to 1, so that we don't include the known routing issue in our cluster. The five service level combinations we use are `<2,0>, <1,0>, <0,0>, <0,1>, <0,2>`, where the first value denotes the service level of the representative application, and the second value denotes the service level of the interference. The default is `<0,0>`. The remaining four configurations should ideally be interference-free (Lustre traffic was disabled during our experiments), because we send the application's communication traffic with a different service level than that of the interference. Having a lower service level for the application represents lower bandwidth for the application. For the remainder of the paper, we denote these five service level combinations as bandwidth levels 1 to 5, where 1 corresponds to low bandwidth and 5 corresponds to high bandwidth for the application.

3.1.1 Scale of Data Collection Similar to the setup in Section 2, we collect our data on 256 of the 324 nodes in the cluster during an allocated dedicated time. Given the exhaustive nature of these experiments, we collected up to three repeated runs on the `random` placement, and only one data point each for the `packed` and `spread` placements. Additionally, because of limited time on the cluster, we collected a full dataset for 512 and 1024 tasks, but a smaller number of configurations at 2048 and 4096 tasks. We also had some experiments that failed due to scheduler issues outside our control. As a result, we collected 874 data points for 512 tasks, 900 data points of 1024 tasks, 654 data points for 2048 tasks, and 300 data points for 4096 tasks spanning the above configurations. The fastest configuration (that is, highest power cap, maximum bandwidth, and highest task count) is tuned to execute for at least 15 seconds. Note that the slowest configuration for the same application (that is, lowest power cap of 64 W, minimum bandwidth, and the lowest task count) may execute about 8x slower than the fastest for strongly scaled applications, and about 4x slower for the weakly scaled applications (due to lower power cap and lower bandwidth). Running these 2728 experiments took a little under 5 days of dedicated and uninterrupted execution time on our 324-node shared cluster, including time for setup, job launch overheads, and general debugging at scale. Because Lustre was unmounted, we also had high job turnaround times (despite relatively short application duration) because we were writing out several files.

3.1.2 Challenges in Data Collection Collecting such data at scale with all the different network, power and placement parameters was non-trivial, making this a unique dataset for analysis of multi-objective application performance. A key challenge when collecting such data is the need for *dedicated time* on the shared cluster. This is because even though some of these knobs are available in user space, tuning these power, network quality of service, and placement knobs impacts other regular users on the cluster and can drastically degrade performance of their applications. Running such experiments repeatedly can also cause temperature hotspots

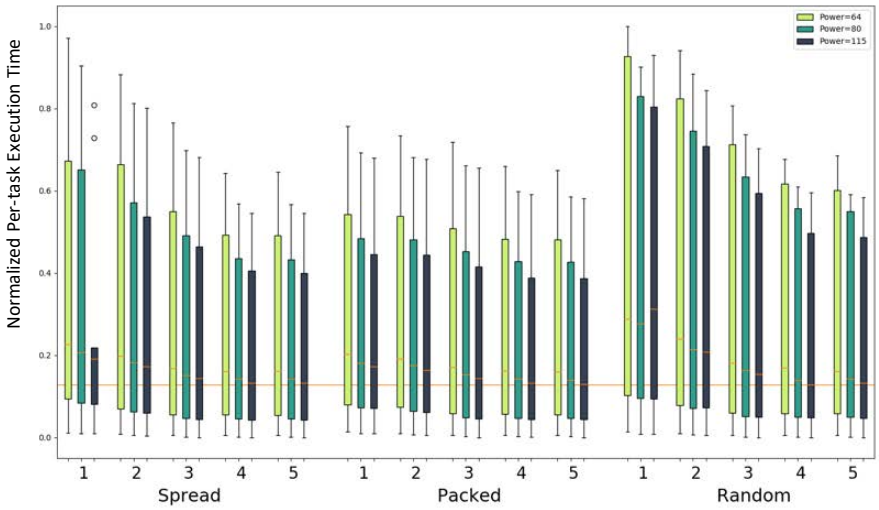


Figure 3: This graph is generated for the FT application with varying number of task placements, service levels, and power caps. The Y-axis has been normalized by the range as well as the number of tasks. The red horizontal line corresponds to the *minimum average* (normalized) per-task execution time.

or thrashing of the native resource manager, resulting in unintended failures for other users. Typical requests for dedicated times on shared clusters do not exceed 12 hours and are granted once per week. In order to enable data collection for over 5 days, system administrators often request that such dedicated time requests be spread across multiple days and multiple time slots. However, spreading such dedicated times over multiple months (which would have been the case for our research) can introduce noise and inconsistency due to upgrades in low-level system software or due to temperature changes in the machine room, etc. Additionally, to allow for a clean execution environment, the Lustre file system traffic which is typically sent through Service Level 1 needs to be entirely disabled, calling for system administrator support. We had a wait time of a few months before our 5-day long dedicated time request was granted in 2017, which made collecting such a dataset challenging and non-trivial, and also making it difficult to re-run failed experiments. We note that availability of such performance data is critical for other performance analysis tools as well as for co-design of future hardware, and making such a dataset available is a unique contribution of this paper. This realization is also the motivation for the proposed machine learning approach for identifying a small subset of the configurations that can be explored to achieve both low execution time and low variation guarantee.

3.2 Analysis and Visualization

Although the first part of our paper is motivated by collecting a large dataset, we strongly believe that such an exhaustive set of data cannot be collected by all users for every application they want to run on an HPC system. Hence, the next part of our paper proposes ways so that we can gain application-independent insights about which of the parameters impact the performance of applications (both the average runtime as well as its reproducibility), so that under a given constraint, a user can identify the minimum number of configurations to explore for an application to estimate an average time which will be reproducible. With this goal, we propose a novel graph signal analysis based machine learning approach to conduct influence analysis on the various parameters in our dataset. By modeling our dataset as a neighborhood graph and relying on Graph Fourier Transform, we view a set of coefficients that allow us to identify the degree of variation in our data and understand relative influences. We use this model-agnostic analysis to derive the *influence score*, which allows us to determine which parameters or sets of parameters are more important for performance tuning than others during the development of predictive models in multi-objective environments. Based on the influence score, we analyze different system-level and application-level parameters and their impact using a novel visualization approach, which we refer to as the *influence path diagram*. We also analyze constrained configurations, such as the power-limited or topology-aware configurations in detail. Here, we present a general system-level configuration that works well for our cluster – this is a counter-intuitive configuration that gives low bandwidth to the application, uses the `spread` placement, and uses maximum power per socket of 115 W. For this analysis, we define a *desirability score*, which allows us to balance optimality and reproducibility. Such an analysis can be conducted on any other cluster (with similar or other multi-objective data such as temperature hotspots or communication requirements, if necessary) to attain similar insights on application performance, desirable configurations, and parameter influence. The strength of our approach is that regardless of the number of tunable parameters, our approach provides a set of easily interpretable suggestions about which system configurations are worthy of exploration out of the inordinate number of possibilities. We present the details of these scores in Section 4. We believe such an approach can lead to development of better mitigation strategies and adaptive system software such as variation-aware resource managers.

Figure 3 presents a traditional analysis of the average and the spread of per-task execution times for the application FT. The red horizontal line corresponds to the *optimality score* for an application,

as it represents the *lowest* average per-task execution time this application, or any similar application, can achieve. Our optimality goal is to achieve a per-task execution time as close to the red line as possible. We consider the average or mean value in order to accommodate for issues such as noise and jitter, and we consider the *lowest* across all configurations to capture the notion of speed of calculation. Note that we can always consider the median or the minimal values, we picked the average as that allows us to set a reasonable user expectation of optimality (based on how job durations are requested).

From this figure, we can observe that configuration <location = `spread`, bandwidth level = 1, power cap = 115W> does not result in the optimal performance for FT, however results in the tightest possible bound on the variability of the execution times across several runs. On the other hand, configuration such as <location = random, service level = 1, power cap = 115> produces both performance that is neither optimal nor reproducible (high variance). Given that we already have this data available, we can make another observation that, those configurations that result in both low average per-task execution times, and low variance across runs, are the desirable ones. Based on this observation, we propose a new metric called *"desirability score"* that helps us compare different configurations based on a score as opposed to looking at so many box plots side-by-side. Equation 1 explains how desirability score is computed based on the mean and variance execution times for each application. We use the desirability score metric to compare across configurations. The higher the desirability score, the better the configuration is in achieving both low average per-task execution time as well as low variation across several runs.

$$desirability\_score = e^{-mean \times variance} \quad (1)$$

3.2.1 Generalization of Results This work develops a methodology that system administrators and users can utilize to understand the trade-off space on performance optimality versus performance reproducibility. It is important to note that every HPC system is different in terms of the user-level and system-level knobs that are exposed. Typically, site-level policies as well as the underlying microarchitecture determine the specific knobs that are made available for tuning. This work specifically contributes a general approach for systematically exploring the trade-off space between optimality and reproducibility. The observations made in this paper will apply to the 324-node cluster described in this paper as well as other HPC systems with similar knobs and microarchitectures. The results also explain how to interpret the results from our methodology when applied to a new system. Similar analysis needs to be performed on other HPC clusters that have different user-level and system-level knobs. The presented methodology for influence analysis and visualization using influence path diagrams will still lead to the identification of the knobs to tune.

4 Methodology

Our influence analysis method builds upon the existing theoretical framework of graph signal analysis (GSA) [62]. The underlying premise of our approach is in modeling the observed data as a neighborhood graph, wherein the nodes correspond to runs from different configurations of system parameters and the edges encode crucial information to perform information propagation and data interpolation. This modeling assumption is driven by the intuition that similar configurations (as measured directly in the parameter space) have the tendency to produce similar performance. In contrast to conventional graph embedding approaches, GSA enables the study of a function defined at the nodes of a graph with respect to the structure encoded by the graph. This is akin to performing Fourier analysis of functions defined in the Euclidean space, to study their spectral characteristics. In our setup, the function, also known as a graph signal, corresponds to the actual performance metric (that is, execution time) observed for each of the configurations. We first begin by describing the fundamental tools in graph signal analysis, and subsequently present our algorithm.

4.1 Graph Signal Analysis

Definitions: Formally, an undirected weighted graph is represented by the triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ denotes the set of nodes with cardinality $|\mathcal{V}| = N$, $\mathcal{E}$ denotes the set of edges, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is an adjacency matrix that specifies the weights on the edges, where $\mathbf{A}_{i,j}$ corresponds to the edge weight between nodes v_i and v_j. Let $\mathcal{N}_i = \{j | \mathbf{A}_{i,j} \neq 0\}$ define the neighborhood of node v_i, i.e. the set of nodes v_j that have incident edges to it. The normalized graph Laplacian, $\mathbf{L}$, is then constructed as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is the degree matrix with diagonal entries $\mathbf{D}_{ii} = \sum_{j \in \mathcal{N}_i} \mathbf{A}_{i,j}$ indicating how strongly connected a node is to the rest of the graph, and $\mathbf{I}$ denotes the identity matrix.

Given the graph $\mathcal{G}$, we define a graph signal $\mathbf{s}$, a numerical function indexed by $\mathcal{V}$, as follows: $\mathbf{s} = [s_1, s_2 \cdots s_N]^T; \forall s_i \in \mathbb{R}$. For example, an image can be represented as pixels defined on a 2–D regular lattice graph, and in this case, the pixel values form the graph signal. Following [62], we define the *graph shift* operator, akin to the *time-shift* operator in classical signal processing. With the graph shift operation, the signal s_i indexed by the node v_i can be transformed as a weighted linear combination of the signal values at the neighboring nodes:

$$\tilde{s}_i = \sum_{j=1}^{N} \mathbf{\Lambda}_{i,j} s_j \implies \tilde{\mathbf{s}} = \mathbf{A}\mathbf{s}. \quad (2)$$

Here, $\mathbf{\Lambda}$ parameterizes the dependencies for the local neighborhood. Consequently, the graph shift can be defined directly using the adjacency matrix $\mathbf{A}$, the transition matrix $\mathbf{D}^{-1}\mathbf{A}$, or the normalized graph Laplacian $\mathbf{L}$. In our implementation, we use the graph Laplacian to define the graph shift.

4.1.1 Graph Fourier Transform: Performing spectral decomposition of a signal space $\mathcal{S}$ is at the core of GSA, to generalize the notion of signal analysis from Euclidean spaces to arbitrary graphs. In general, spectral decomposition of a signal space corresponds to identifying subspaces that are invariant to the choice of filtering, i.e. the filtered version of a signal from subspace $\mathcal{S}_k$ still lies in that subspace. The set of generalized eigenvectors of the graph Laplacian, $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{U} \in \mathbb{R}^{N \times N}$, is referred to as the graph Fourier basis. Consequently, decomposition of a signal $\mathbf{s} \in \mathcal{S}$ corresponds to computing its expansion in the graph Fourier basis: $\mathbf{s} = \mathbf{U}\hat{\mathbf{s}}$, where the expansion coefficients can be computed as $\hat{\mathbf{s}} = \mathbf{U}^{-1}\mathbf{s}$. This process is popularly referred as the Graph Fourier Transform (GFT), and the collection of coefficients $\hat{\mathbf{s}}$ can be viewed as the *spectrum* [15]. Note that, the ordered set of eigenvalues loosely represent frequencies of signal variation, with λ_1 to λ_N representing the smallest to largest frequencies. In other words, larger signal

variations between closely connected neighbors correspond to high frequencies, while smooth variations correspond to low frequencies. In this context, the graph filtering using a graph shift operator corresponds to a simple *low-pass* filter. In other words, such a filtering operation will retain only signal characteristics that vary smoothly across close neighborhoods, in other words, are more predictable.

4.2 Algorithm

Identifying parameters that have a strong influence on the optimality or reproducibility of performance characteristics is at the core of our analysis. From a machine learning (ML) standpoint, this is referred to as feature selection. Broadly, *predictability* and *sensitivity* are two commonly used heuristics for selecting features. While the former measures how well a feature supports the overall prediction, the latter measures how much the prediction is bound to change when a feature is perturbed. In practice, either of these metrics are estimated based on an ML approach trained on the data – this inherently involves understanding bias-variance trade-off and the analysis results can change significantly based on the model choice. To circumvent this challenge, we propose to employ model-agnostic feature selection based on GSA. In the rest of this section, we will define a novel *influence* score based on graph spectral analysis, and describe the idea of *influence path diagrams*, that can provide a comprehensive understanding of different design choices on optimality.

4.2.1 Influence Score: We propose a general metric for qualitatively evaluating the influence of any subset of design features in predicting performance. For a given set of input features for N cases, we first construct a $k-$nearest neighbor graph $\mathcal{G}$ using the Gower distance between those features, which can support the use of different data types for each of the features. Subsequently, we compute the GFT of the performance metric $\mathbf{s}$, and measure the magnitude of the resulting spectrum. In the context of graph signal analysis, a predictable signal is characterized by the smoothness property with respect to the neighborhoods. In other words, we expect a graph spectrum to be dominated by low frequency content when the signal is predictable in the domain considered. Similarly, a response function that is highly uncorrelated with the predictor variables manifests as a spectrum with majority of its energy concentrated at higher frequencies. Note that, frequencies here refer to the eigen bases corresponding to the smallest eigen values. We now define the influence score for the chosen subset of features as $\mathcal{I} = \exp(-\hat{s}_m)$, where $\hat{s}_m$ denotes the average of the Fourier coefficients $\hat{\mathbf{s}}$ from the expansion $\mathbf{U}^{-1}\mathbf{s}$. Smaller values for $\mathcal{I}$ implies that the spectrum is skewed heavily towards the low frequencies.

4.2.2 Influence Path Diagram: To understand the complex interactions between design variables in high-dimensional spaces, we propose to leverage the influence score from GSA and produce a summary layout of influences of different subsets of variables. Referred to as an *influence path diagram* (IPD), this is a directed graph layout where each node represents different subsets of features (drawn as circles). For example, if our design space was comprised of two variables (x_1, x_2), the IPD will contain three nodes, namely $(x1)$, $(x2)$ and $(x1, x2)$. Each node in an IPD, n_i contains a directed edge to another node n_j if and only if the feature sets at nodes n_i and n_j differ by only one feature. For example, in the case of a design space with 3 features x_1, x_2, x_3, the node containing (x_1) will have an edge to the nodes corresponding to the subsets $(x1, x_2)$ and $(x1, x_3)$ respectively, but not to the node with (x_1, x_2, x_3). We compute the influence score at each node of the IPD using only the feature subset included at the node and indicate that score as the fill color of the nodes. Note that, darker the color, higher the influence score of those features. The edges encode how much the influence changes by the inclusion of one additional feature. The change is measured as $(\mathcal{I}(n_j) - \mathcal{I}(n_i))/\mathcal{I}(n_j)$. A blue edge indicates a positive change in the influence, while a red edge indicates a negative change. While, dark shaded edges correspond to significant changes, dotted lines indicate trivial changes in influence. As we show in our empirical studies, the proposed IPD provides crucial insights into complex design spaces for performance optimization and by entirely dispensing the need for explicit model design, this approach is highly flexible, when compared to existing machine learning pipelines.

Configuration	ID	Parameters (L, N, P)
Traditional	1	<random, 3, 115W>
Power-limited	2	<random, 3, 64W>
Network-limited	3	<random, 1, 115W>
Topology-aware	4	<packed, 5, 115W>
Topology-aware	5	<spread, 5, 115W>
Proposed	6	<spread, 1, 115W>

Table 4: Description of configurations.

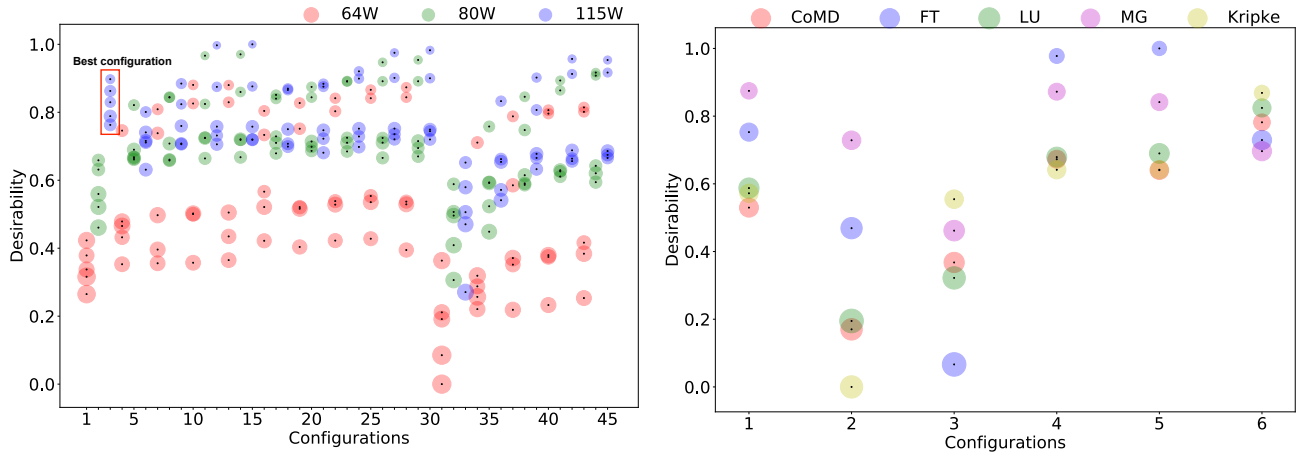
5 Results

We divide this section into three parts. First, we present detailed analysis of the collected data to understand the optimality and reproducibility trade-off across configurations. Next, we analyze selected constrained configurations and their impact on performance variability. Finally, we address the question which user- and system-level parameters are more influential from the point of view of predictability of the average execution time using our novel visualization. This analysis guides a user to find the smallest set of parameters that need to be tuned for any application to achieve performance predictability.

5.1 Optimality versus Reproducibility

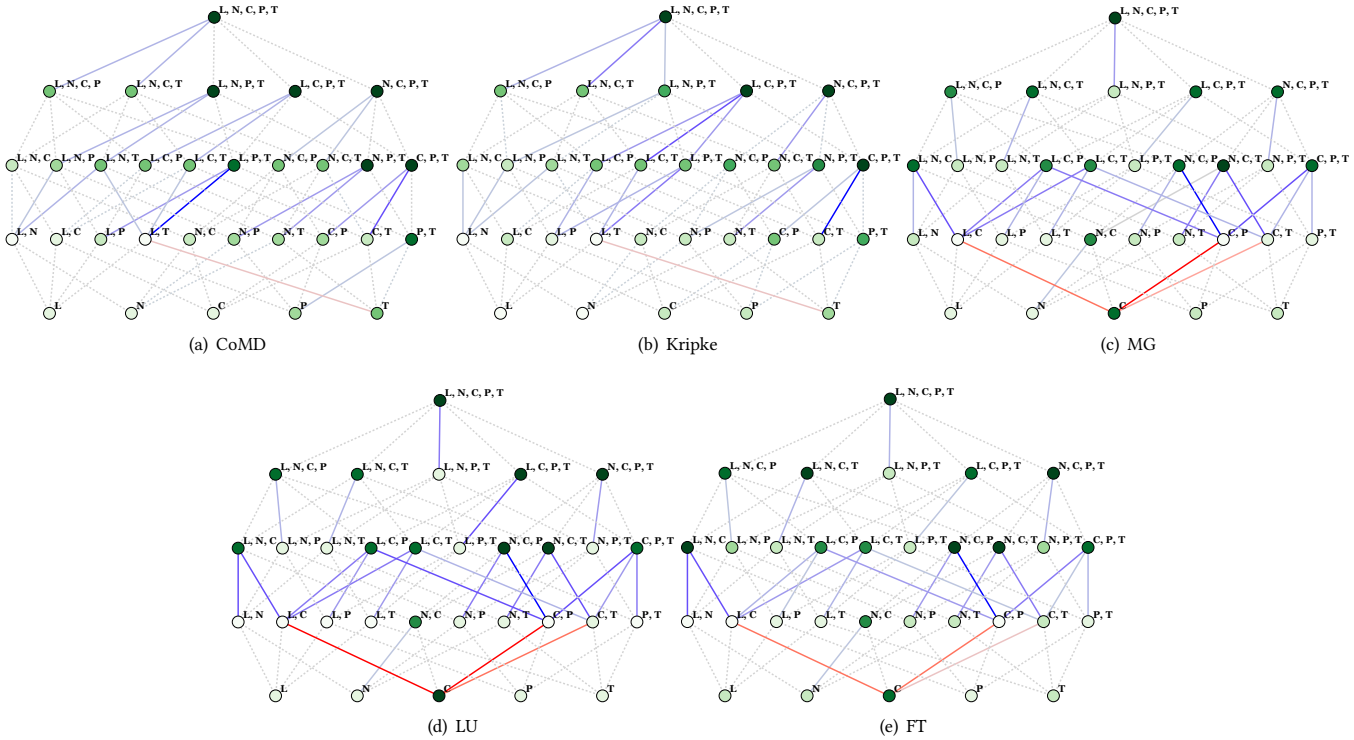
Here, we explored all possible system configurations described previously in Table 3 for all five applications using the desirability score, as shown in Figure 4a. Along X-axis, we show a hierarchical grouping of rank placement, the bandwidth level, and the power cap, which amount to a total of 45 possible configurations for our cluster. For each such configuration, the Y-axis shows the desirability score for the five benchmarks. We do not annotate the benchmarks in this graph so as to get a general understanding of the configuration space. We compare all configurations to identify conditions that lead to a balance in performance optimality and reproducibility using the desirability score. Note that execution times were divided by the task count to obtain a per-task execution time.

The X-axis shows all configurations, and the Y-axis shows the desirability scores (Equation 1) normalized with respect to the minimum and the maximum values. The higher the score, it is more likely that a configuration will finish quicker and the performance will be reproducible. The size of the bubbles indicate variance across the per-task execution times, hence the larger the circle, more the



(a) All configurations (b) Constrained configurations

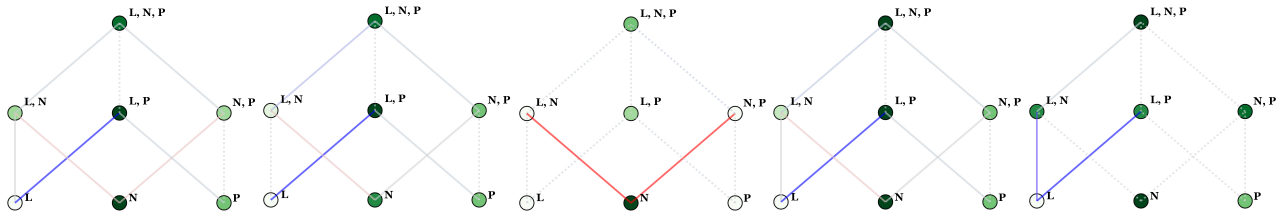
Figure 4: (a) The impact of various configurations on the average per-task execution times across applications from run-to-run. (b) Comparison of the six constrained configurations described in Table 4. X-axis shows the configurations, and Y-axis shows the desirability score for each application scaled with respect to the minimum and the maximum range of the scores.



(a) CoMD (b) Kripke (c) MG

(d) LU (e) FT

Figure 5: Impact of both system- and user-level parameters on performance reproducibility. Darker circle indicates higher influence. Red line indicates loss of influence, blue solid lines indicate improved influence, dotted lines indicate nominal change in influence, and solid line indicates significant improvement. Here C indicates the number of nodes, T indicates the number of cores, L is placement, N is bandwidth level, and P indicates the power cap.



(a) CoMD (b) Kripke (c) MG (d) LU (e) FT

Figure 6: Impact of system-level parameters only.

variance. The take-away point of this figure is that the random configurations result in both sub-optimal runs across applications (low score) as well as more variability (larger circles). While packing tasks onto the same router results in the most optimal execution for some applications, tasks spread across routers and executed under 115W offers an environment where applications can finish quickly with certain guarantee of repeatability. Hence, to execute a new application in this HPC environment, if one would have to choose a configuration that offers the best trade-off between optimal and reproducible performance (high score, and all applications as close to each other as possible), it would be <spread, 1, 115W>.

5.2 Understanding Constrained Configurations

The objective of this experiment is to compare several constrained environments that HPC applications are likely to encounter, and their impact on the desirability score (Equation 1). Table 4 describes the five different constrained configurations typical HPC systems may be running with. From Figure 4b, we observe that power-limited configurations (e.g., configuration 2) impact performance variability the most across all applications, and topology-aware configurations (4, 5) result in optimal performance across all applications (that is, the average execution times of all applications is the lowest, but variance is high). Our proposed configuration (number 6) offers a reasonable balance across optimality and reproducibility. Because there are several parameters that are different across these configurations, and some constraints (such as power) may be unavoidable, it is not immediately evident which parameters are more important to tune. In order to infer the importance of different features and sets of features, we run the next experiment.

5.3 Influence Analysis

In this experiment, shown in Figures 5 and 6, we leverage the novel influence score computation scheme we proposed earlier to communicate importance of tuning certain parameters to achieve better reproducibility of the average mean.

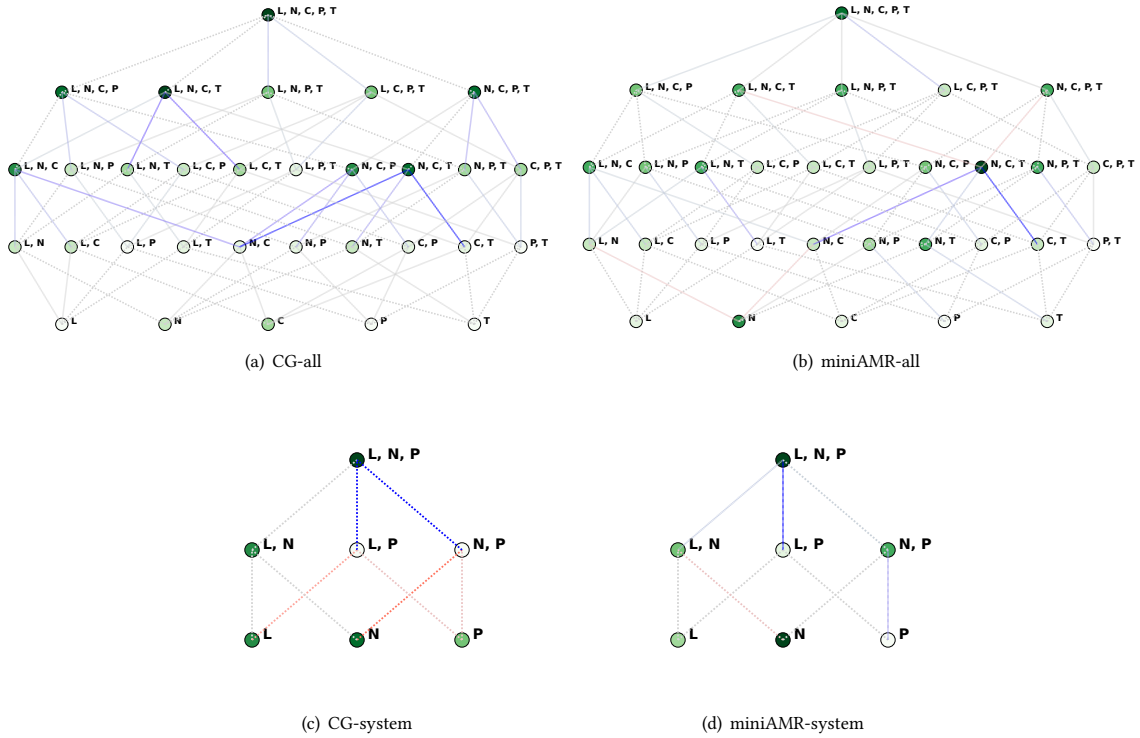
Starting from the bottom layer, we fix the specified variables at each step, and calculate the influence score based on the novel methodology presented in Section 4. A darker circle means a combination of parameters is more important than others in improving predictability, thus producing repeatable observations. The graph should be read from bottom up, different types of lines mean different influence values. Red and blue lines, respectively, indicate loss and increase of influence. Dotted lines and solid lines, respectively, indicate nominal and significant change in influence. Here L is placement, N is bandwidth level, C indicates the number of nodes, P is the power cap, and T is the number of cores (or threads).

The objective of this experiment is two fold. First, we want to identify which parameters are influential for strongly-scaled and weakly-scaled applications. Second, we want to understand which system-level parameters impact the reproducibility of the average performance for applications. The goal is for a user to not have to run exhaustive number of configurations to identify the best possible setting for their application or system. Once a user identifies a subset of the parameters by our approach to be influential, they can only focus on tuning those knobs to achieve the most repeatable optimal average runtime.

5.3.1 ***Strong-scaling versus weak-scaling*** From Figure 5, we can observe that the number of nodes is the most important (dark circle) parameter that impact the performance for strongly-scaled applications (MG, LU, and FT) on the current system, while the pair of power-cap and the number of cores is important for weakly-scaled applications (Kripke, and CoMD). In other words, this indicates that once an application is run with the above mentioned parameters set, users can expect higher predictability in execution time from run to run on the current system. We also observe that, after the node count, for the strongly-scaled applications, the bandwidth level is the most important parameter that impacts reproducibility of the execution times. One may have to tune the bandwidth level parameter (for example, with `IPATH_SL`) for greater reproducibility, which may be a system-level parameter and hence harder to tune. On the other hand, specifying the power cap and core/thread count (P and T) for weakly-scaled applications would greatly improve the reproducibility of these observations.

We also observe that, for strongly-scaled applications, the node count and the bandwidth levels (C and N) provided to the application are the two most influential parameters to tune. There are total 5 parameters with a total of 540 combinations for evaluation (4 node counts, 3 thread/core counts, 3 power caps, 3 location/placement algorithms, and 5 bandwidth levels). Figure 5 suggests that a user can only evaluate only 20 values (4 node counts, 5 bandwidth levels) while leaving the other parameters to the default value to estimate an average execution time which will be reproducible from run to run for a new application. This observation reduces the total number of configurations to evaluate for a new application with strong scaling up to 97% on the current HPC system.

5.3.2 ***Impact of System-level Parameters Alone*** In this experiment, we normalized the execution times by the total number of tasks (Figure 6). The rationale is that we want to observe the impact of these system-level parameters on the average execution times of these applications and not worry about possible work imbalance



(a) CG-all (b) miniAMR-all

(c) CG-system (d) miniAMR-system

Figure 7: Validation of our observations with two new applications on the described system. Here C indicates the number of nodes, T indicates the number of cores, L is placement, N is bandwidth level, and P indicates the power cap.

across the tasks. A darker circle indicates that setting specific values for a set of parameters result in greater reproducibility in per-task execution times from run to run. From Figure 5, we can observe that execution times across all application and all parameters tend to be stable once the bandwidth level (N) is determined. After that, the next parameter that impacts the reproducibility of applications is the power cap (P). Note that not all applications are impacted by the CPU power cap (for example, MG is memory bound and is not as impacted). We also observe that, after adjusting the bandwidth level and power, the task placement only improves the score nominally.

5.4 Discussion and Validation

We observe that even though the traditional configuration allows equal bandwidth to all the running applications, it is not ideal for high-priority applications of interest. Through our novel influence path diagrams, we observe that the reproducibility of applications is impacted the most by the bandwidth provided, or by a pair of system parameters such as rank placement (location) and the power-cap when considered together. For certain HPC systems, controlling the quality of service parameter can be difficult. In that case, our experiments show that specifying the task placement and the power-cap could be the way to achieve reproducibility across the applications of interest in the presence of an interference in the system. We can also identify certain common combinations of system-level parameters that have higher impact on the reproducibility that hold true across applications (e.g. bandwidth level), which is interesting.

For validating our observations, we collected additional data for two more applications in August 2019: CG [3] and miniAMR [74] on the Catalyst cluster. Note that this data is being collected on the system after two years of the original data collection process of 2017. As a result, various maintenance cycles, compiler, and operating system upgrades have been conducted between the initial data collection and the newly collected data. Additionally, the routing related issue that we detected earlier has been resolved. Figure 7 presents the influence path diagram results for these two new applications. For CG, we used class D. This application calculates the conjugate gradient, and has characteristics that include irregular memory access and communication. MiniAMR applies a stencil calculation on a unit cube computational domain, and is primarily communication bound. For miniAMR, the `npx,npy,nyz` options were dependent on the number of tasks that we were running. We set the number of time-steps to 200 and the stages per time step to 100. Total blocks were determined by the application itself through weak scaling as 2048K, 4128K, and 8208K for the 1024-, 2048- and 4096-task experiments respectively. Even after two years, for the two new applications, we observe that the bandwidth level parameter (N) is still the most influential of all knobs. Another observation that can be made is that tuning the task placement (L) parameter next can be beneficial for both applications. Tuning power cap alone for miniAMR or with task placement for CG does not contribute new information. This is because neither of these applications is compute-bound, and they see minimal impact based on power.

We are able to claim that for the Catalyst cluster, in order to achieve the most reproducible average execution time that is also the fastest, a user with a new set of applications would have to only primarily tune the bandwidth level provided to the application. If that is not possible for the user, then they can achieve a desirable level of predictability in average execution time by only tuning the power cap and the task placement specifications in most cases. The outcome of the machine learning model can be integrated into a performance auto-tuner to explore the search space in a smart manner. Our observations will apply to other applications on the HPC system described in this paper. We also expect our observations to apply to other systems with the same knobs and the same micro-architecture. Since most HPC systems collect similar data during acceptance test, system administrators can apply our model to their collected dataset to identify the impact of those knobs. Once the users and administrators understand the influence of their user- and system-level parameters on optimality and reproducibility for a particular HPC system, both parties can solely focus on tuning only the essential ones. Based on our validation, we also note that regular updates to such a dataset are necessary after major system upgrades for determining configurations that lead to high desirability scores.

6 Related Work

Existing research has examined performance variation resulting from network interference [8, 33, 35, 75], I/O congestion [2, 26] and from power constrained scenarios that bring out chip-level manufacturing differences [12, 19, 29, 34, 42, 56, 61, 71, 72]. A large body of work has explored solutions for the aforementioned variation through adaptive resource management, including power-aware schedulers [20–22, 25, 27, 36, 40, 52, 59, 60, 63, 67, 76–78], and network-aware or I/O-aware schedulers [33, 41, 54, 55, 68, 79]. Additionally, existing research has also explored intelligent task mapping and job placements to improve communication performance [6, 7, 11, 28, 48, 58]. Predicting user estimates of the runtime of their jobs so as to ensure that they not killed prematurely due to underestimation has also been studied from the point of view of noise and variation [23, 66]. Furthermore, several machine learning techniques to predict application performance and detect anomalies for both power and network related variations have been explored [5, 9, 16, 32, 43, 44, 70, 73], and the visualization community has come up with novel mechanisms to analyze network traffic and performance data [10, 24, 31, 39, 45, 69]. However, all the studies listed above have focused on a single objective such as network traffic, or a single constraint such as power. There is little research that explores the *simultaneous impact* of network, power, and concurrency in real systems [50]. This limited research is either focused on power and IO management [14, 64] or on temperature-awareness for communication-intensive workloads [46, 47]. Our work is the first-of-its-kind study to support a truly multi-objective environment and to quantitatively evaluate various configurations on a real system while understanding the influence scores of different parameters. This research is a necessary precursor to developing adaptive system software that involves management of different heterogeneous hardware components and diverse resources such as power, network and I/O.

7 Conclusion

Limited research exists on understanding how various sources of heterogeneity in computational environments, such as network traffic, power limits, task placements, and interference from other jobs, impact both the average execution times of applications and their predictability from run to run. While a low average execution time has always been a quantity that has been sought after, only recently the need for performance reproducibility has been recognized. There is a vast gap in understanding the trade-off between performance optimality and reproducibility. To bridge that gap, this paper takes two steps. First, based on measurements on a production HPC system, this paper presents a metric called *desirability score* that can be used to compare various system configurations quantitatively. Second, we propose a novel machine learning technique based on graph signal analysis for determining the influence of parameters on performance predictability. The second part of the contribution is unique and is being proposed in order to reduce the inordinate number of combinations of both user- and system-level parameters that can be tuned in an HPC environment, while some combinations can be hard to attain under certain constraint.

To the best of our knowledge, this paper is the first to provide both a thorough analysis of a multi-objective dataset from the optimality and reproducibility trade-off standpoint. Future work involves collecting data across various other production clusters and gaining a deeper understanding of parameter influence in multi-objective environments and to develop finer-grained approaches for the desirability score.

Acknowledgments

Authors would like to thank Dr. Ayse Coskun and Emre Ates for the technical discussions and initial data collection in the early stages of this work. We would also like to thank Livermore Computing, specifically Ryan Day, Greg Tomaschke, Adam Moody, Don Frederick, Christopher Springmeyer and Lance Weems. They helped with administering multiple dedicated time requests on the Catalyst cluster with a total span of over 12 total days for our experiments. We are grateful for the support we received from them with regards to understanding OpenSM configurations, debugging network routing issues, unmounting Lustre filesytem, and minimizing node downtimes. This work would not have been possible without their constant support. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-788156).

References

[1] 2016. OSU Benchmarks. http://mvapich.cse.ohio-state.edu/benchmarks/. (2016).
[2] Ana Gainaru Ana, Guillaume Aupy, Anne Benoit, Franck Cappello, Yves Robert, and Marc Snir. 2015. Scheduling the I/O of HPC applications under congestion. In IEEE International Parallel and Distributed Processing Symposium (IPDPS).
[3] David H. Bailey. 2006. NASA Advanced Supercomputing Division, NAS Parallel Benchmark Suite v3.3. (2006).
[4] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. [n. d.]. The NAS Parallel Benchmarks. In Proceedings of the 1991 ACM/IEEE Conference on Supercomputing (Supercomputing '91).
[5] Bradley J. Barnes, Barry Rountree, David K. Lowenthal, Jaxk Reeves, Bronis de Supinski, and Martin Schulz. 2008. A Regression-based Approach to Scalability Prediction. In Proceedings of the 22nd Annual International Conference on Supercomputing. 368–377.

[6] Abhinav Bhatele. 2010. Automating Topology Aware Mapping for Supercomputers. In PhD Thesis, Dept. of Computer Science, University of Illinois. http://hdl.handle.net/2142/16578.

[7] Abhinav Bhatele, Todd Gamblin, Steven H. Langer, Peer-Timo Bremer, Erik W. Draeger, Bernd Hamann, Katherine E. Isaacs, Aaditya G. Landge, Joshua A. Levine, Valerio Pascucci, Martin Schulz, and Charles H. Still. 2012. Mapping Applications with Collectives over Sub-communicators on Torus Networks. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12).

[8] Abhinav Bhatele, Kathryn Mohror, Steven H. Langer, and Katherine E. Isaacs. 2013. There Goes the Neighborhood: Performance Degradation Due to Nearby Jobs. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13).

[9] A. Bhatele, A. R. Titus, J. J. Thiagarajan, N. Jain, T. Gamblin, P. T. Bremer, M. Schulz, and L. V. Kale. 2015. Identifying the Culprits Behind Network Congestion. In Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International.

[10] H. Bhatia, N. Jain, A. Bhatele, Y. Livnat, J. Domke, V. Pascucci, and P.-T. Bremer. 2018. Interactive Investigation of Traffic Congestion on Fat-Tree Networks Using TreeScope. Computer Graphics Forum 37, 3 (2018), 561–572. https://doi.org/10.1111/cgf.13442

[11] S.H. Bokhari. 1981. On the Mapping Problem. Computers, IEEE Transactions on C-30, 3 (March 1981), 207–214.

[12] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. 2003. Parameter Variations and Impact on Circuits and Microarchitecture. In Proceedings of the 40th annual Design Automation Conference. 338–342.

[13] M. Broyles, C. Cain, T. Rosedahl, and G. Silva. 2015. IBM EnergyScale for POWER8 Processor-Based Systems. In IBM Whitepaper.

[14] R. R. Chandrasekar, A. Venkatesh, K. Hamidouche, and D. K. Panda. 2015. Power-Check: An Energy-Efficient Checkpointing Framework for HPC Clusters. In 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.

[15] Siheng Chen, Rohan Varma, Aliaksei Sandryhaila, and Jelena Kovačević. 2015. Discrete signal processing on graphs: Sampling theory. IEEE transactions on signal processing 63, 24 (2015), 6510–6523.

[16] Ryan Cochran, Can Hankendi, Ayse K Coskun, and Sherief Reda. 2011. Pack & Cap: adaptive DVFS and thread packing under power caps. In Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture. ACM, 175–185.

[17] Diego Crupnicoff, Sujal Das, and Eitan Zahavi. 2005. Deploying Quality of Service and Congestion Control in InfiniBand-based Data Center Networks. Technical Report. Mellanox Technologies.

[18] Howard David, Eugene Gorbatov, Ulf Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: Memory Power Estimation and Capping. In Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design (ISLPED '10). 189–194.

[19] S. Dighe, S.R. Vangal, P. Aseron, S. Kumar, T. Jacob, K.A. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, V.K. De, and S. Borkar. 2011. Within-Die Variation-Aware Dynamic-Voltage-Frequency-Scaling With Optimal Core Allocation and Thread Hopping for the 80-Core TeraFLOPS Processor. Solid-State Circuits, IEEE Journal of 46, 1 (Jan 2011), 184–193.

[20] Maja Etinski, Julita Corbalan, Jesus Labarta, and Mateo Valero. 2010. Optimizing Job Performance Under a Given Power Constraint in HPC Centers. In Green Computing Conference. 257–267.

[21] Maja Etinski, Julita Corbalan, Jesus Labarta, and Mateo Valero. 2011. Linear Programming Based Parallel Job Scheduling for Power Constrained Systems. In International Conference on High Performance Computing and Simulation. 72–80.

[22] Maja Etinski, Julita Corbalan, Jesus Labarta, and Mateo Valero. 2012. Parallel Job Scheduling for Power Constrained HPC Systems. *Parallel Comput.* 38, 12 (Dec. 2012), 615–630.

[23] Y. Fan, P. Rich, W. E. Allcock, M. E. Papka, and Z. Lan. 2017. Trade-Off Between Prediction Accuracy and Underestimation Rate in Job Runtime Estimates. In 2017 IEEE International Conference on Cluster Computing (CLUSTER). 530–540. https://doi.org/10.1109/CLUSTER.2017.11

[24] T. Fujiwara, P. Malakar, K. Reda, V. Vishwanath, M. E. Papka, and K. Ma. 2017. A Visual Analytics System for Optimizing Communications in Massively Parallel Applications. In 2017 IEEE Conference on Visual Analytics Science and Technology (VAST). 59–70.

[25] Yiannis Georgiou, Thomas Cadeau, David Glesser, Danny Auble, Morris Jette, and Matthieu Hautreux. 2014. Energy Accounting and Control with SLURM Resource and Job Management System. In Distributed Computing and Networking. Lecture Notes in Computer Science, Vol. 8314. Springer Berlin Heidelberg, 96–118.

[26] Luís Fabrício Góes, Pedro Guerra, Bruno Coutinho, Leonardo Rocha, Wagner Meira, Renato Ferreira, Dorgival Guedes, and Walfredo Cirne. 2005. AnthillSched: A Scheduling Strategy for Irregular and Iterative I/O-Intensive Parallel Jobs. In Job Scheduling Strategies for Parallel Processing: 11th International Workshop, JSSPP 2005.

[27] I. Goiri, Kien Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini. 2011. GreenSlot: Scheduling Energy Consumption in Green Datacenters. In High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for. 1–11.

[28] T. Hoefler and M. Snir. 2011. Generic Topology Mapping Strategies for Large-scale Parallel Architectures. In Proceedings of the 2011 ACM International Conference on Supercomputing (ICS'11). ACM, 75–85.

[29] Yuichi Inadomi, Tapasya Patki, Koji Inoue, Mutsumi Aoyagi, Barry Rountree, Martin Schulz, David Lowenthal, Yasutaka Wada, Keiichiro Fukazawa, Masatsugu Ueda, Masaaki Kondo, and Ikuo Miyoshi. 2015. Analyzing and Mitigating the Impact of Manufacturing Variability in Power-constrained Supercomputing. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15).

[30] Intel. 2011. Intel-64 and IA-32 Architectures Software Developer's Manual, Volumes 3A and 3B: System Programming Guide. (2011).

[31] Katherine E. Isaacs, Alfredo Giménez, Ilir Jusufi, Todd Gamblin, Abhinav Bhatele, Martin Schulz, Bernd Hamann, and Timo Bremer. 2014. State of the Art of Performance Visualization. In EuroVis.

[32] Nikhil Jain, Abhinav Bhatele, Louis H. Howell, David Böhme, Ian Karlin, Edgar A. León, Misbah Mubarak, Noah Wolfe, Todd Gamblin, and Matthew L. Leininger. 2017. Predicting the Performance Impact of Different Fat-tree Configurations. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17). ACM, New York, NY, USA, Article 50, 13 pages. https://doi.org/10.1145/3126908.3126967

[33] Nikhil Jain, Abhinav Bhatele, Xiang Ni, Todd Gamblin, and Laxmikant V. Kale. 2017. Partitioning Low-diameter Networks to Eliminate Inter-job Interference. In Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS '17 (to appear)). IEEE Computer Society. LLNL-CONF-.

[34] Sudhakar Jilla. 2013. Minimizing The Effects of Manufacturing Variation During Physcial Layout. Chip Design Magazine (2013). http://chipdesignmag.com/display.php?articleId=2437.

[35] A. Jokanovic, J. C. Sancho, G. Rodriguez, A. Lucero, C. Minkenberg, and J. Labarta. 2015. Quiet Neighborhoods: Key to Protect Job Performance Predictability. In 2015 IEEE International Parallel and Distributed Processing Symposium. 449–459. https://doi.org/10.1109/IPDPS.2015.87

[36] Kyong Hoon Kim, R Buyya, and Jong Kim. 2007. Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters. In Cluster Computing and the Grid, 2007. CCGRID 2007. 541–548.

[37] R. Kent Koeninger. 2003. The Ultra-Scalable HPTC Lustre Filesystem. Cluster World (2003).

[38] A. J. Kunen, T. S. Bailey, and P. N. Brown. [n. d.]. KRIPKE - A Massively Parallel Transport Mini-App. In American Nuclear Society M&C 2015.

[39] Aaditya G Landge, Joshua A Levine, Abhinav Bhatele, Katherine E Isaacs, Todd Gamblin, Martin Schulz, Steve H Langer, P-T Bremer, and Valerio Pascucci. 2012. Visualizing network traffic to understand the performance of massively parallel simulations. Visualization and Computer Graphics, IEEE Transactions on 18, 12 (2012), 2467–2476.

[40] Barry Lawson and Evgenia Smirni. 2005. Power-aware Resource Allocation in High-end Systems via Online Simulation. In International onference on Supercomputing. 229–238.

[41] Kangkang Li, Maciej Malawski, and Jarek Nabrzyski. 2017. Topology-aware Job Allocation in 3D Torus-based HPC Systems with Hard Job Priority Constraints. Procedia Computer Science 108 (2017), 515 – 524. https://doi.org/10.1016/j.procs.2017.05.016 International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

[42] Xiaoyao Liang and David Brooks. 2006. Mitigating the Impact of Process Variations on Processor Register Files and Execution Units. In International Symposium on Microarchitecture. 504–514.

[43] Aniruddha Marathe, Rushil Anirudh, Nikhil Jain, Abhinav Bhatele, Jayaraman Thiagarajan, Bhavya Kailkhura, Jae-Seung Yeom, Barry Rountree, and Todd Gamblin. 2017. Performance Modeling Under Resource Constraints Using Deep Transfer Learning. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17). ACM, New York, NY, USA, Article 31, 12 pages. https://doi.org/10.1145/3126908.3126969

[44] Aleksander Maricq, Dmitry Duplyakin, Ivo Jimenez, Carlos Maltzahn, Ryan Stutsman, and Robert Ricci. 2018. Taming Performance Variability. Berkeley, CA, USA. http://dl.acm.org/citation.cfm?id=3291168.3291198

[45] C. M. McCarthy, K. E. Isaacs, A. Bhatele, P. Bremer, and B. Hamann. 2014. Visualizing the Five-dimensional Torus Network of the IBM Blue Gene/Q. In 2014 First Workshop on Visual Performance Analysis. 24–27. https://doi.org/10.1109/VPA.2014.10

[46] Jie Meng, Eduard Llamosí, Fulya Kaplan, Chulian Zhang, Jiayi Sheng, Martin Herbordt, Gunar Schirner, and Ayse K Coskun. 2016. Communication and cooling aware job allocation in data centers for communication-intensive workloads. *J. Parallel and Distrib. Comput.* 96 (2016), 181–193.

[47] Jie Meng, Samuel McCauley, Fulya Kaplan, Vitus J. Leung, and Ayse K. Coskun. 2015. Simulation and optimization of {HPC} job allocation for jointly reducing communication and cooling costs. Sustainable Computing: Informatics and Systems 6 (2015), 48 – 57. Special Issue on Selected Papers from 2013 International Green Computing Conference (IGCC).

[48] G. Michelogiannakis, K. Z. Ibrahim, J. Shalf, J. J. Wilke, S. Knight, and J. P. Kenny. 2017. APHiD: Hierarchical Task Placement to Enable a Tapered Fat Tree Topology for Lower Power and Cost in HPC Networks. In 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). 228–237. https://doi.org/10.1109/CCGRID.2017.33

[49] Adam Moody. 2009. Contention-Free Routing for Shift-based Communication in MPI Applications on Large-scale InfiniBand Clusters. LLNL-TR-418522, Lawrence Livermore National Laboratory, Livermore, CA (October 2009).

[50] T. Patki, E. Ates, A. Coskun, and J. Thiagarajan. 2018. Understanding Simultaneous Impact of Network QoS and Power on HPC Application Performance. In Computational Reproducibility at Exascale (CRE'18), Supercomputing Workshop 2018.

[51] Tapasya Patki, David K. Lowenthal, Barry Rountree, Martin Schulz, and Bronis R. de Supinski. 2013. Exploring Hardware Overprovisioning in Power-constrained, High Performance Computing. In International Conference on Supercomputing.

[52] Tapasya Patki, Anjana Sasidharan, Matthias Maiterth, David Lowenthal, Barry Rountree, Martin Schulz, and Bronis de Supinski. 2015. Practical Resource Management in Power-Constrained, High Performance Computing. In High Performance Parallel and Distributed Computing (HPDC).

[53] Olga Pearce, Hadia Ahmed, Rasmus W. Larsen, Peter Pirkelbauer, and David F. Richards. 2017. Exploring dynamic load imbalance solutions with the CoMD proxy application. Future Generation Computer Systems (2017). http://www.sciencedirect.com/science/article/pii/S0167739X17300560

[54] Samuel D. Pollard, Nikhil Jain, Stephen Herbein, and Abhinav Bhatele. 2018. Evaluation of an Interference-free Node Allocation Policy on Fat-tree Clusters. In Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '18). IEEE Press, Piscataway, NJ, USA, Article 26, 13 pages. http://dl.acm.org/citation.cfm?id=3291656.3291691

[55] R. Rajachandrasekar, J. Jaswani, H. Subramoni, and D. K. Panda. 2012. Minimizing Network Contention in InfiniBand Clusters with a QoS-Aware Data-Staging Framework. In 2012 IEEE International Conference on Cluster Computing.

[56] Barry Rountree, Dong H. Ahn, Bronis R. de Supinski, David K. Lowenthal, and Martin Schulz. 2012. Beyond DVFS: A First Look at Performance under a Hardware-Enforced Power Bound. In IPDPS Workshops (HPPAC). IEEE Computer Society, 947–953.

[57] Barry Rountree and Stephanie Labasan. [n. d.]. Libmsr. https://github.com/LLNL/libmsr. ([n. d.]).

[58] P. Sadayappan and F. Ercal. 1987. Nearest-Neighbor Mapping of Finite Element Graphs onto Processor Meshes. Computers, IEEE Transactions on C-36, 12 (Dec 1987), 1408–1424.

[59] R. Sakamoto, T. Cao, M. Kondo, K. Inoue, M. Ueda, T. Patki, D. Ellsworth, B. Rountree, and M. Schulz. 2017. Production Hardware Overprovisioning: Real-World Performance Optimization Using an Extensible Power-Aware Resource Management Framework. In 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS). 957–966. https://doi.org/10.1109/IPDPS.2017.107

[60] R. Sakamoto, T. Patki, T. Cao, M. Kondo, K. Inoue, M. Ueda, D. Ellsworth, B. Rountree, and M. Schulz. 2018. Analyzing Resource Trade-offs in Hardware Overprovisioned Supercomputers. In 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS). 526–535. https://doi.org/10.1109/IPDPS.2018.00062

[61] Samie B. Samaan. 2004. The Impact of Device Parameter Variations on the Frequency and Performance of VLSI Chips. In Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on. 343–346.

[62] Aliaksei Sandryhaila and José MF Moura. 2013. Discrete signal processing on graphs. IEEE transactions on signal processing 61, 7 (2013), 1644–1656.

[63] Osman Sarood, Akhil Langer, Abhishek Gupta, and Laxmikant V. Kale. 2014. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. In Supercomputing.

[64] Lee Savoie, David K Lowenthal, Bronis R De Supinski, Tanzima Islam, Kathryn Mohror, Barry Rountree, and Martin Schulz. 2016. I/O Aware Power Shifting. In Proceedings - 2016 IEEE 30th International Parallel and Distributed Processing Symposium, IPDPS 2016. Institute of Electrical and Electronics Engineers Inc., United States, 740–749. https://doi.org/10.1109/IPDPS.2016.15

[65] Kathleen Shoga, Barry Rountree, and Martin Schulz. 2014. Whitelisting MSRs with msr-safe. Third Workshop on Extreme-Scale Programming Tools, held with SC 14 (November 2014).

[66] Wei Tang, N. Desai, D. Buettner, and Zhiling Lan. 2010. Analyzing and Adjusting User Runtime Estimates to Improve Job Scheduling on the Blue Gene/P. In Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on. 1–11.

[67] R. Teodorescu and J. Torrellas. 2008. Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors. In Computer Architecture, 2008. ISCA '08. 35th International Symposium on. 363–374.

[68] Sagar Thapaliya, Purushotham Bangalore, Jay Lofstead, Kathryn Mohror, and Adam Moody. 2014. IO-Cop: Managing Concurrent Accesses to Shared Parallel File System. In International Conference on Parallel Processing Workshops (ICCPW).

[69] L. Theisen, A. Shah, and F. Wolf. 2014. Down to Earth - How to Visualize Traffic on High-dimensional Torus Networks. In 2014 First Workshop on Visual Performance Analysis. 17–23. https://doi.org/10.1109/VPA.2014.6

[70] J. J. Thiagarajan, R. Anirudh, B. Kailkhura, N. Jain, T. Islam, A. Bhatele, J. Yeom, and T. Gamblin. 2018. PADDLE: Performance Analysis Using a Data-Driven Learning Environment. In 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS). 784–793. https://doi.org/10.1109/IPDPS.2018.00088

[71] Ehsan Totoni, Akhil Langer, Josep Torrellas, and Laxmikant Kale. 2015. Scheduling for HPC Systems with Process Variation Heterogeneity. (January 2015).

[72] James W. Tschanz, James T. Kao, Siva G. Narendra, Raj Nair, Dmitri A. Antoniadis, Anantha P. Chandrakasan, and Vivek De. 2002. Adaptive Body Bias for Reducing Impacts of Die-to-die and Within-die Parameter Variations on Microprocessor Frequency and Leakage. Solid-State Circuits, IEEE Journal of 37, 11 (Nov 2002), 1396–1402.

[73] Ozan Tuncer, Emre Ates, Yijia Zhang, Ata Turk, Jim Brandt, Vitus Leung, Manuel Egele, and Ayse K. Coskun. 2017. Diagnosing Performance Variations in HPC Applications using Machine Learning. International Supercomputing Conference in High Performance Computing (ISC-HPC) (June 2017).

[74] C. T. Vaughan and R. F. Barrett. 2015. Enabling Tractable Exploration of the Performance of Adaptive Mesh Refinement. In 2015 IEEE International Conference on Cluster Computing. 746–752. https://doi.org/10.1109/CLUSTER.2015.129

[75] X. Yang, J. Jenkins, M. Mubarak, R. B. Ross, and Z. Lan. 2016. Watch Out for the Bully! Job Interference Study on Dragonfly Network. In SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 750–760. https://doi.org/10.1109/SC.2016.63

[76] Xu Yang, Zhou Zhou, Sean Wallace, Zhiling Lan, Wei Tang, Susan Coghlan, and Michael E. Papka. 2013. Integrating Dynamic Pricing of Electricity into Energy Aware Scheduling for HPC Systems. In International Conference for High Performance Computing, Networking, Storage and Analysis. 17–22.

[77] Ziming Zhang, Michael Lang, Scott Pakin, and Song Fu. 2014. Trapped Capacity: Scheduling under a Power Cap to Maximize Machine-room Throughput. In Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing. IEEE Press, 41–50.

[78] Zhou Zhou, Zhiling Lan, Wei Tang, and Narayan Desai. 2014. Reducing Energy Costs for IBM Blue Gene/P via Power-Aware Job Scheduling. In Job Scheduling Strategies for Parallel Processing. Springer Berlin Heidelberg, 96–115.

[79] Z. Zhou, X. Yang, Z. Lan, P. Rich, W. Tang, V. Morozov, and N. Desai. 2015. Improving Batch Scheduling on Blue Gene/Q by Relaxing 5D Torus Network Allocation Constraints. In 2015 IEEE International Parallel and Distributed Processing Symposium. 439–448. https://doi.org/10.1109/IPDPS.2015.110

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

We ran five representative benchmarks: NAS LU, MG, FT (Class D), and Kripke and CoMD. We used the 150 TFLOPS/s 324-node catalyst cluster at Lawrence Livermore National Laboratory. Each node in this cluster has two 12-core Intel Xeon CPUs and two HCA network adapters. It is supported by an InfiniBand QDR network with a two-level fat-tree topology, and has a layout of a total of 18 switches, with 18 nodes per switch.

For our dataset, we varied core count (16, 20, 24 cores per node), task count (512, 1024, 2048, 4096 MPI ranks), power cap (64 W, 80 W, 115 W per socket), network QoS levels (combinations ranging from service level 0 to 2), and application placement algorithms (packed, spread and random assignment). All benchmarks are MPI-based, compiled with Intel compiler 16.0.3 and MVAPICH 2.2. Each representative application ran for at least 15 seconds at the fastest configuration (we added iteration counts) to ensure that we have reliable timing data. These have been described in detail in our paper.

We used msr-safe (https://github.com/LLNL/msr-safe) and libmsr (https://github.com/LLNL/libmsr) for power capping. Our machine learning scripts and python visualization scripts will be made available along with our dataset.

ARTIFACT AVAILABILITY

Software Artifact Availability: All author-created software artifacts are maintained in a public repository under an OSI-approved license.

Hardware Artifact Availability: There are no author-created hardware artifacts.

Data Artifact Availability: All author-created data artifacts are maintained in a public repository under an OSI-approved license.

Proprietary Artifacts: None of the associated artifacts, author-created or otherwise, are proprietary.

List of URLs and/or DOIs where artifacts are available:

```
We will make our repository URL with data and scripts
↪ publicly available by November 2019 after release
↪ from LLNL. The link will be included in our final
↪ slides.

This will include our unique dataset, scripts for our
↪ novel machine learning model based on graph
↪ signal analysis, and scripts for our novel
↪ visualization diagram.
```

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: Catalyst cluster at LLNL with 324 nodes. Each node in this cluster has two 12-core Intel Xeon CPUs and two HCA network adapters. It is supported by an InfiniBand QDR network with a two-level fat-tree topology, and has a layout of a total of 18 switches, with 18 nodes per switch.

Operating systems and versions: TOSS Operating System (Trilab's version of Linux)

Compilers and versions: Intel Compiler 16.0.3

Applications and versions: NAS Parallel Benchmarks (MG, LU, FT), Kripke, CoMD

Libraries and versions: MVAPICH 2.2

Key algorithms: N/A (we proposed a new ML algorithm)

Input datasets and versions: Class D for NAS. 240-480 for the x,y,z inputs of CoMD. 122-256 for the zone inputs for Kripke. More details in paper.

Paper Modifications: Note: Note that the collect.sh scripts prints a different Intel compiler and complains about requiring superuser permissions. When we ran these experiments, we used Intel Compiler 16.0.3 with MVAPICH 2.2. They were run in October 2017, so the environment listed below isn't a 100% accurate, but our results are still valid and acceptable and we have given the details of all the right environments in the paper.

Output from scripts that gathers execution environment information.

```
We have attached the output of the requested
↪ collect.sh script as desired. This was collected
↪ at the time of writing this paper in April 2019,
↪ but our dataset was collected in October 2017.

\LMOD_FAMILY_COMPILER_VERSION=18.0.1
_ModuleTable011_=ZXMvTGludXg6L3Vzci9zaGFyZS9tb2R1bGV
↪ maWxlcy9Db3JlOi91c3Ivc2hhcmUvbG1vZC9sbW9kL21vZHV
↪ sZWZpbGVzL0NvcmUiLH0=
MANPATH=/g/g90/USER/src/spack/opt/spack/linux-rhel7-
↪ x86_64/gcc-4.9.3/lz4-1.8.1.2-p7grndyqisjn4t6q4ki
↪ icwykmx7ppumx/share/man:/g/g90/USER/src/spack/op
↪ t/spack/linux-rhel7-x86_64/gcc-4.9.3/python-2.7.
↪ 14-4agkfvsiakut5hgegek4psxq7i3wp5be/share/man:/u
↪ sr/tce/packages/mvapich2/mvapich2-2.3-intel-18.0
↪ .1/man:/usr/tce/packages/intel/intel-18.0.1/man/
↪ common:/usr/tce/man:/usr/share/lmod/lmod/share/m
↪ an:/usr/man:/usr/share/man:/usr/local/man:/usr/X
↪ 11R6/man:/usr/lib64/mvapich/default/man
HOSTNAME=catalyst160
GUESTFISH_INIT=\e[1;34m
```

```
_ModuleTable003_=dmFwaWNoMj17WyJmbiJdPSIvdXNyL3RjZS9
↪ tb2R1bGVmaWxlcy9Db21waWxlci9pbnRlbC8xOC4wLjEvbXZ
↪ hcGljaDIvMi4zLmx1YSIsWyJmdWxsTmFtZSJdPSJtdmFwaWN
↪ oMi8yLjMiLFsibG9hZE9yZGVyIl09Mixwcm9wVD17fSxbInN
↪ 0YWNrRGVwdGgiXT0xLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ
↪ 1c2VyTmFtZSJdPSJtdmFwaWNoMi8yLjMiLH0sWyJweS1jZmZ
↪ pLTEuMS4yLWdjYy00LjkuMy1mamtyeng1Il09e1siZm4iXT0
↪ iL2cvZzkwL3BhdGtpMS9zcmMvc3BhY2svc2hhcmUvc3BhY2s
↪ vbW9kdWxlcy9saW51eC1yaGVsNy14ODZfNjQvcHktY2ZmaS0
↪ xLjEuMi1nY2MtNC45LjMtZmprcnp4NSIsWyJmdWxsTmFtZSJ
↪ dPSJweS1jZmZpLTEuMS4yLWdjYy00LjkuMy1mamtyeng1
SPACK_ROOT=/g/g90/USER/src/spack
INTEL_LICENSE_FILE=/usr/tce/packages/intel/intel-18.
↪ 0.1/compilers_and_libraries_2018.1.163/linux/Lic
↪ enses/license.client.intel.lic
_ModuleTable009_=WyJzdGF0dXMiXT0iYWN0aXZlIixbInVzZXJ
↪ OYW1lIl09InB5dGhvbi0yLjcuMTQtZ2NjLTQuOS4zLTRhZ2t
↪ mdnMiLH0sdGV4bGl2ZT17WyJmbiJdPSIvdXNyL3RjZS9tb2R
↪ 1bGVmaWxlcy9Db3JlL3RleGxpdmUvMjAxNi5sdWEiLFsiZnV
↪ sbE5hbWUiXT0idGV4bGl2ZS8yMDE2IixbImxvYWRPcmRlciJ
↪ dPTMscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09MSxbInN0YXR
↪ 1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0idGV4bGl2ZS8
↪ yMDE2Iix9LH0sbXBhdGhBPXsiL2cvZzkwL3BhdGtpMS9zcmM
↪ vc3BhY2svc2hhcmUvc3BhY2svbW9kdWxlcy9saW51eC1yaGV
↪ sNy14ODZfNjQiLCIvdXNyL3RjZS9tb2R1bGVmaWxlcy9NUEk
↪ vbXZhcGljaDIvMi4zIiwiL3Vzci90Y2UvbW9kdWxlZmls
SHELL=/bin/bash
TERM=xterm-256color
__LMOD_REF_COUNT_MODULEPATH=/g/g90/USER/src/spack/sh
↪ are/spack/modules/linux-rhel7-x86_64:1;/usr/tce/
↪ modulefiles/MPI/mvapich2/2.3:1;/usr/tce/modulefi
↪ les/MPI/intel/18.0.1/mvapich2/2.3:1;/usr/tce/mod
↪ ulefiles/Compiler/intel/18.0.1:1;/collab/usr/glo
↪ bal/tools/modulefiles/toss_3_x86_64_ib/Core:1;/u
↪ sr/tce/modulefiles/Core:1;/usr/apps/modulefiles:
↪ 1;/usr/share/modulefiles/Linux:1;/usr/share/modu
↪ lefiles/Core:1;/usr/share/lmod/lmod/modulefiles/
↪ Core:1
HISTSIZE=1000
WISECONFIGDIR=/usr/share/wise2/
CLICOLOR=1
SSH_CLIENT=76.102.78.63 56641 22
TMPDIR=/var/tmp/USER
MODULEPATH_ROOT=/usr/share/modulefiles
LMOD_SYSTEM_DEFAULT_MODULES=StdEnv
LDMS_XPRT_LIBPATH=/g/g90/USER/Source/local_ldms/ldms
↪ .usr/lib/ovis-ldms
LIBRARY_PATH=/g/g90/USER/src/spack/opt/spack/linux-r
↪ hel7-x86_64/gcc-4.9.3/lz4-1.8.1.2-p7grndyqisjn4t
↪ 6q4kiicwykmx7ppumx/lib:/g/g90/USER/src/spack/opt
↪ /spack/linux-rhel7-x86_64/gcc-4.9.3/py-functools
↪ 32-3.2.3-2-3xjeigiqad6thua7ycs3j7ndcegeo7vb/lib:
↪ /g/g90/USER/src/spack/opt/spack/linux-rhel7-x86_
↪ 64/gcc-4.9.3/py-jsonschema-2.5.1-73eyv2likbxldo7
↪ s5rqt4uqhvzga2azq/lib:/g/g90/USER/src/spack/opt/
↪ spack/linux-rhel7-x86_64/gcc-4.9.3/py-pyyaml-3.1
↪ 3-nmgcaksh22c5p25lwwwhgtouz6rg4uu4/lib:/g/g90/US
↪ ER/src/spack/opt/spack/linux-rhel7-x86_64/gcc-4.
↪ 9.3/py-six-1.10.0-z6ym6ink72tpqqpaw3mnr7xxzj7ci2
↪ 5d/lib:/g/g90/USER/src/spack/opt/spack/linux-rhe
↪ l7-x86_64/gcc-4.9.3/py-pycparser-2.17-eifsucgo4w
↪ 7fjiiyfbz4gyj5wsmytkd2/lib:/g/g90/USER/src/spack
↪ /opt/spack/linux-rhel7-x86_64/gcc-4.9.3/py-cffi-
↪ 1.1.2-fjkrzx5dyljwlikbjlgv7z3ibnzlc5fw/lib:/g/g9
↪ 0/USER/src/spack/opt/spack/linux-rhel7-x86_64/gc
↪ c-4.9.3/python-2.7.14-4agkfvsiakut5hgegek4psxq7i
↪ 3wp5be/lib
LMOD_PKG=/usr/share/lmod/lmod
QTDIR=/usr/lib64/qt-3.3
QTINC=/usr/lib64/qt-3.3/include
LMOD_VERSION=7.8.16
SSH_TTY=/dev/pts/6
LC_ALL=C
__LMOD_REF_COUNT_LOADEDMODULES=intel/18.0.1:1;mvapic
↪ h2/2.3:1;texlive/2016:1;StdEnv:1;python-2.7.14-g
↪ cc-4.9.3-4agkfvs:1;py-cffi-1.1.2-gcc-4.9.3-fjkrz
↪ x5:1;py-pycparser-2.17-gcc-4.9.3-eifsucg:1;py-si
↪ x-1.10.0-gcc-4.9.3-z6ym6in:1;py-pyyaml-3.13-gcc-
↪ 4.9.3-nmgcaks:1;py-jsonschema-2.5.1-gcc-4.9.3-73
↪ eyv2l:1;py-functools32-3.2.3-2-gcc-4.9.3-3xjeigi
↪ :1;lz4-1.8.1.2-gcc-4.9.3-p7grndy:1
SPACK_SHELL=bash
__LMOD_REF_COUNT_CMAKE_PREFIX_PATH=/g/g90/USER/src/s
↪ pack/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/lz4-
↪ 1.8.1.2-p7grndyqisjn4t6q4kiicwykmx7ppumx:1;/g/g9
↪ 0/USER/src/spack/opt/spack/linux-rhel7-x86_64/gc
↪ c-4.9.3/py-functools32-3.2.3-2-3xjeigiqad6thua7y
↪ cs3j7ndcegeo7vb:1;/g/g90/USER/src/spack/opt/spac
↪ k/linux-rhel7-x86_64/gcc-4.9.3/py-jsonschema-2.5
↪ .1-73eyv2likbxldo7s5rqt4uqhvzga2azq:1;/g/g90/USE
↪ R/src/spack/opt/spack/linux-rhel7-x86_64/gcc-4.9
↪ .3/py-pyyaml-3.13-nmgcaksh22c5p25lwwwhgtouz6rg4u
↪ u4:1;/g/g90/USER/src/spack/opt/spack/linux-rhel7
↪ -x86_64/gcc-4.9.3/py-six-1.10.0-z6ym6ink72tpqqpaw
↪ 3mnr7xxzj7ci25d:1;/g/g90/USER/src/spack/opt/spac
↪ k/linux-rhel7-x86_64/gcc-4.9.3/py-pycparser-2.17
↪ -eifsucgo4w7fjiiyfbz4gyj5wsmytkd2:1;/g/g90/USER/s
↪ rc/spack/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/
↪ py-cffi-1.1.2-fjkrzx5dyljwlikbjlgv7z3ibnzlc5fw:1
↪ ;/g/g90/USER/src/spack/opt/spack/linux-rhel7-x86
↪ _64/gcc-4.9.3/python-2.7.14-4agkfvsiakut5hgegek4
↪ psxq7i3wp5be:1
```

```
_ModuleTable007_=c3BhY2svc2hhcmUvc3BhY2svbW9kdWxlcy9⌟
↪  saW51eC1yaGVsNy14ODZfNjQvcHktcHl5YW1sLTMuMTMtZ2N⌟
↪  jLTQuOS4zLW5tZ2Nha3MiLFsiZnVsbE5hbWUiXT0icHktcHl⌟
↪  5YW1sLTMuMTMtZ2NjLTQuOS4zLW5tZ2Nha3MiLFsibG9hZE9⌟
↪  yZGVyIl09OSxwcm9wVD17fSxbInN0YWNrRGVwdGgiXT0wLFs⌟
↪  ic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZSJdPSJweS1⌟
↪  weXlhbWwtMy4xMy1nY2MtNC45LjMtbm1nY2FrcyIsfSxbInB⌟
↪  5LXNpeC0xLjEwLjAtZ2NjLTQuOS4zLXo2eW02aW4iXT17WyJ⌟
↪  mbiJdPSIvZy9nOTAvcGF0a2kxL3NyYy9zcGFjay9zaGFyZS9⌟
↪  zcGFjay9tb2R1bGVzL2xpbnV4LXJoZWw3LXg4Nl82NC9weS1⌟
↪  zaXgtMS4xMC4wLWdjYy00LjkuMy16NnltNmluIixbImZ1
USER=USER
LS_COLORS=rs=0:di=38;5;27:ln=38;5;51:mh=44;38;5;15:p⌟
↪  i=40;38;5;11:so=38;5;13:do=38;5;5:bd=48;5;232;38⌟
↪  ;5;11:cd=48;5;232;38;5;3:or=48;5;232;38;5;9:mi=0⌟
↪  5;48;5;232;38;5;15:su=48;5;196;38;5;15:sg=48;5;1⌟
↪  1;38;5;16:ca=48;5;196;38;5;226:tw=48;5;10;38;5;1⌟
↪  6:ow=48;5;10;38;5;21:st=48;5;21;38;5;15:ex=38;5;⌟
↪  34:*.tar=38;5;9:*.tgz=38;5;9:*.arc=38;5;9:*.arj=⌟
↪  38;5;9:*.taz=38;5;9:*.lha=38;5;9:*.lz4=38;5;9:*.⌟
↪  lzh=38;5;9:*.lzma=38;5;9:*.tlz=38;5;9:*.txz=38;5⌟
↪  ;9:*.tzo=38;5;9:*.t7z=38;5;9:*.zip=38;5;9:*.z=38⌟
↪  ;5;9:*.Z=38;5;9:*.dz=38;5;9:*.gz=38;5;9:*.lrz=38⌟
↪  ;5;9:*.lz=38;5;9:*.lzo=38;5;9:*.xz=38;5;9:*.bz2=⌟
↪  38;5;9:*.bz=38;5;9:*.tbz=38;5;9:*.tbz2=38;5;9:*.⌟
↪  tz=38;5;9:*.deb=38;5;9:*.rpm=38;5;9:*.jar=38;5;9⌟
↪  :*.war=38;5;9:*.ear=38;5;9:*.sar=38;5;9:*.rar=38⌟
↪  ;5;9:*.alz=38;5;9:*.ace=38;5;9:*.zoo=38;5;9:*.cp⌟
↪  io=38;5;9:*.7z=38;5;9:*.rz=38;5;9:*.cab=38;5;9:*⌟
↪  .jpg=38;5;13:*.jpeg=38;5;13:*.gif=38;5;13:*.bmp=⌟
↪  38;5;13:*.pbm=38;5;13:*.pgm=38;5;13:*.ppm=38;5;1⌟
↪  3:*.tga=38;5;13:*.xbm=38;5;13:*.xpm=38;5;13:*.ti⌟
↪  f=38;5;13:*.tiff=38;5;13:*.png=38;5;13:*.svg=38;⌟
↪  5;13:*.svgz=38;5;13:*.mng=38;5;13:*.pcx=38;5;13:⌟
↪  *.mov=38;5;13:*.mpg=38;5;13:*.mpeg=38;5;13:*.m2v⌟
↪  =38;5;13:*.mkv=38;5;13:*.webm=38;5;13:*.ogm=38;5⌟
↪  ;13:*.mp4=38;5;13:*.m4v=38;5;13:*.mp4v=38;5;13:*⌟
↪  .vob=38;5;13:*.qt=38;5;13:*.nuv=38;5;13:*.wmv=38⌟
↪  ;5;13:*.asf=38;5;13:*.rm=38;5;13:*.rmvb=38;5;13:⌟
↪  *.flc=38;5;13:*.avi=38;5;13:*.fli=38;5;13:*.flv=⌟
↪  38;5;13:*.gl=38;5;13:*.dl=38;5;13:*.xcf=38;5;13:⌟
↪  *.xwd=38;5;13:*.yuv=38;5;13:*.cgm=38;5;13:*.emf=⌟
↪  38;5;13:*.axv=38;5;13:*.anx=38;5;13:*.ogv=38;5;1⌟
↪  3:*.ogx=38;5;13:*.aac=38;5;45:*.au=38;5;45:*.fla⌟
↪  c=38;5;45:*.mid=38;5;45:*.midi=38;5;45:*.mka=38;⌟
↪  5;45:*.mp3=38;5;45:*.mpc=38;5;45:*.ogg=38;5;45:*⌟
↪  .ra=38;5;45:*.wav=38;5;45:*.axa=38;5;45:*.oga=38⌟
↪  ;5;45:*.spx=38;5;45:*.xspf=38;5;45:
LMOD_sys=Linux
LD_LIBRARY_PATH=/g/g90/USER/src/spack/opt/spack/linu⌟
↪  x-rhel7-x86_64/gcc-4.9.3/lz4-1.8.1.2-p7grndyqisj⌟
↪  n4t6q4kiicwykmx7ppumx/lib:/g/g90/USER/src/spack/⌟
↪  opt/spack/linux-rhel7-x86_64/gcc-4.9.3/py-functo⌟
↪  ols32-3.2.3-2-3xjeigiqad6thua7ycs3j7ndcegeo7vb/l⌟
↪  ib:/g/g90/USER/src/spack/opt/spack/linux-rhel7-x⌟
↪  86_64/gcc-4.9.3/py-jsonschema-2.5.1-73eyv2likbxl⌟
↪  do7s5rqt4uqhvzga2azq/lib:/g/g90/USER/src/spack/o⌟
↪  pt/spack/linux-rhel7-x86_64/gcc-4.9.3/py-pyyaml-⌟
↪  3.13-nmgcaksh22c5p25lwwwhgtouz6rg4uu4/lib:/g/g90⌟
↪  /USER/src/spack/opt/spack/linux-rhel7-x86_64/gcc⌟
↪  -4.9.3/py-six-1.10.0-z6ym6ink72tpqqpaw3mnr7xxzj7c⌟
↪  i25d/lib:/g/g90/USER/src/spack/opt/spack/linux-r⌟
↪  hel7-x86_64/gcc-4.9.3/py-pycparser-2.17-eifsucgo⌟
↪  4w7fjiiyfbz4gyj5wsmytkd2/lib:/g/g90/USER/src/spa⌟
↪  ck/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/py-cff⌟
↪  i-1.1.2-fjkrzx5dyljwlikbjlgv7z3ibnzlc5fw/lib:/g/⌟
↪  g90/USER/src/spack/opt/spack/linux-rhel7-x86_64/⌟
↪  gcc-4.9.3/python-2.7.14-4agkfvsiakut5hgegek4psxq⌟
↪  7i3wp5be/lib:/usr/tce/packages/mvapich2/mvapich2⌟
↪  -2.3-intel-18.0.1/lib:/usr/tce/packages/intel/int⌟
↪  el-18.0.1/lib/intel64:/g/g90/USER/src/libmsr_ins⌟
↪  tall/lib:/usr/lib:/usr/lib64:/g/g90/USER/Source/⌟
↪  local_ldms/lib64:/g/g90/USER/Source/local_ldms/l⌟
↪  dms.usr/lib/ovis-ldms
_ModuleTable010_=ZXMvTVBJL2ludGVsLzE4LjAuMS9tdmFwaWN⌟
↪  oMi8yLjMiLCIvdXNyL3RjZS9tb2R1bGVmaWxlcy9Db21waWx⌟
↪  lci9pbnRlbC8xOC4wLjEiLCIvY29sbGFiL3Vzci9nbG9iYWw⌟
↪  vdG9vbHMvbW9kdWxlZmlsZXMvdG9zc18zX3g4Nl82NF9pYi9⌟
↪  Db3JlIiwiL3Vzci90Y2UvbW9kdWxlZmlsZXMvQ29yZSIsIi9⌟
↪  1c3IvYXBwcy9tb2R1bGVmaWxlcyIsIi91c3Ivc2hhcmUvbW9⌟
↪  kdWxlZmlsZXMvTGludXgiLCIvdXNyL3NoYXJlL21vZHVsZWZ⌟
↪  pbGVzL0NvcmUiLCIvdXNyL3NoYXJlL2xtb2QvbG1vZC9tb2R⌟
↪  1bGVmaWxlcy9Db3JlIix9LFsic3lzdGVtQmFzZU1QQVRIIl0⌟
↪  9Ii91c3IvdGNlL21vZHVsZWZpbGVzL0NvcmU6L3Vzci9hcHB⌟
↪  zL21vZHVsZWZpbGVzOi91c3Ivc2hhcmUvbW9kdWxlZmls
ENV=/g/g90/USER/.bashrc
PFTP_CONFIG_FILENAME=/etc/pftp_config
HOST_GRP=linux
_ModuleTable004_=IixbImxvYWRPcmRlciJdPTYscHJvcFQ9e30⌟
↪  sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmU⌟
↪  iLFsidXNlck5hbWUiXT0icHktY2ZmaS0xLjEuMi1nY2MtNC4⌟
↪  5LjMtZmprcnp4NSIsfSxbInB5LWZ1bmN0b29sczMyLTMuMi4⌟
↪  zLTItZ2NjLTQuOS4zLTN4amVpZ2kiXT17WyJmbiJdPSIvZy9⌟
↪  nOTAvcGF0a2kxL3NyYy9zcGFjay9zaGFyZS9zcGFjay9tb2R⌟
↪  1bGVzL2xpbnV4LXJoZWw3LXg4Nl82NC9weS1mdW5jdG9vbHM⌟
↪  zMi0zLjIuMy0yLWdjYy00LjkuMy0zeGplaWdpIixbImZ1bGx⌟
↪  OYW1lIl09InB5LWZ1bmN0b29sczMyLTMuMi4zLTItZ2NjLTQ⌟
↪  uOS4zLTN4amVpZ2kiLFsibG9hZE9yZGVyIl09MTEscHJvcFQ⌟
↪  9e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJh
CPATH=/g/g90/USER/src/spack/opt/spack/linux-rhel7-x8⌟
↪  6_64/gcc-4.9.3/lz4-1.8.1.2-p7grndyqisjn4t6q4kiic⌟
↪  wykmx7ppumx/include:/g/g90/USER/src/spack/opt/sp⌟
↪  ack/linux-rhel7-x86_64/gcc-4.9.3/python-2.7.14-4⌟
↪  agkfvsiakut5hgegek4psxq7i3wp5be/include
```

```
__LMOD_REF_COUNT__LMFILES_=/usr/tce/modulefiles/Core
↪  /intel/18.0.1.lua:1;/usr/tce/modulefiles/Compile
↪  r/intel/18.0.1/mvapich2/2.3.lua:1;/usr/tce/modul
↪  efiles/Core/texlive/2016.lua:1;/usr/tce/modulefi
↪  les/Core/StdEnv.lua:1;/g/g90/USER/src/spack/shar
↪  e/spack/modules/linux-rhel7-x86_64/python-2.7.14
↪  -gcc-4.9.3-4agkfvs:1;/g/g90/USER/src/spack/share/
↪  spack/modules/linux-rhel7-x86_64/py-cffi-1.1.2-g
↪  cc-4.9.3-fjkrzx5:1;/g/g90/USER/src/spack/share/s
↪  pack/modules/linux-rhel7-x86_64/py-pycparser-2.1
↪  7-gcc-4.9.3-eifsucg:1;/g/g90/USER/src/spack/shar
↪  e/spack/modules/linux-rhel7-x86_64/py-six-1.10.0
↪  -gcc-4.9.3-z6ym6in:1;/g/g90/USER/src/spack/share/
↪  spack/modules/linux-rhel7-x86_64/py-pyyaml-3.13-
↪  gcc-4.9.3-nmgcaks:1;/g/g90/USER/src/spack/share/
↪  spack/modules/linux-rhel7-x86_64/py-jsonschema-2
↪  .5.1-gcc-4.9.3-73eyv2l:1;/g/g90/USER/src/spack/s
↪  hare/spack/modules/linux-rhel7-x86_64/py-functoo
↪  ls32-3.2.3-2-gcc-4.9.3-3xjeigi:1;/g/g90/USER/src
↪  /spack/share/spack/modules/linux-rhel7-x86_64/lz
↪  4-1.8.1.2-gcc-4.9.3-p7grndy:1
GUESTFISH_PS1=\[\e[1;32m\]><fs>\[\e[0;31m\]
LMOD_PREPEND_BLOCK=normal
LMOD_FAMILY_MPI_VERSION=2.3
LSCOLORS=ExFxBxDxCxegedabagacad
PATH=/g/g90/USER/Source/local_ldms/ldms.usr/sbin/:/g
↪  /g90/USER/src/spack/opt/spack/linux-rhel7-x86_64
↪  /gcc-4.9.3/lz4-1.8.1.2-p7grndyqisjn4t6q4kiicwykm
↪  x7ppumx/bin:/g/g90/USER/src/spack/opt/spack/linu
↪  x-rhel7-x86_64/gcc-4.9.3/py-jsonschema-2.5.1-73e
↪  yv2likbxldo7s5rqt4uqhvzga2azq/bin:/g/g90/USER/sr
↪  c/spack/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/p
↪  ython-2.7.14-4agkfvsiakut5hgegek4psxq7i3wp5be/bi
↪  n:/g/g90/USER/src/spack/bin:/usr/tce/packages/te
↪  xlive/texlive-2016/2016/bin/x86_64-linux:/usr/tc
↪  e/packages/mvapich2/mvapich2-2.3-intel-18.0.1/bi
↪  n:/usr/tce/packages/intel/intel-18.0.1/bin:/usr/
↪  tce/bin:/usr/lib64/qt-3.3/bin:/usr/local/bin:/us
↪  r/bin:/usr/local/sbin:/usr/sbin
MAIL=/var/spool/mail/USER
_ModuleTable001_=X01vZHVsZVRhYmxlXz17WyJNVHZlcnNpb24
↪  iXT0zLFsiY19yZWJ1aWxkVGltZSJdPWZhbHNlLFsiY19zaG9
↪  ydFRpbWUiXT1mYWxzZSxkZXB0aFQ9e30sZmFtaWx5PXtbImN
↪  vbXBpbGVyIl09ImludGVsIixbIm1waSJdPSJtdmFwaWNoMiI
↪  sfSxtVD17U3RkRW52PXtbImZuIl09Ii91c3IvdGNlL21vZHV
↪  sZWZpbGVzL0NvcmUvU3RkRW52Lmx1YSIsWyJmdWxsTmFtZSJ
↪  dPSJTdGRFbnYiLFsibG9hZE9yZGVyIl09NCxwcm9wVD17fSx
↪  bInN0YWNrRGVwdGgiXT0wLFsic3RhdHVzIl09ImFjdGl2ZSI
↪  sWyJ1c2VyTmFtZSJdPSJTdGRFbnYiLH0saW50ZWw9e1siZm4
↪  iXT0iL3Vzci90Y2UvbW9kdWxlZmlsZXMvQ29yZS9pbnRlbC8
↪  xOC4wLjEubHVhIixbImZ1bGxOYW1lIl09ImludGVsLzE4
LCSCHEDCLUSTER=catalyst
_=/usr/bin/env
LDMSD_PLUGIN_LIBPATH=/g/g90/USER/Source/local_ldms/l
↪  dms.usr/lib/ovis-ldms
LMOD_SETTARG_CMD=:
PWD=/g/g90/USER
INPUTRC=/etc/inputrc
_LMFILES_=/usr/tce/modulefiles/Core/intel/18.0.1.lua
↪  :/usr/tce/modulefiles/Compiler/intel/18.0.1/mvap
↪  ich2/2.3.lua:/usr/tce/modulefiles/Core/texlive/2
↪  016.lua:/usr/tce/modulefiles/Core/StdEnv.lua:/g/
↪  g90/USER/src/spack/share/spack/modules/linux-rhe
↪  l7-x86_64/python-2.7.14-gcc-4.9.3-4agkfvs:/g/g90
↪  /USER/src/spack/share/spack/modules/linux-rhel7-
↪  x86_64/py-cffi-1.1.2-gcc-4.9.3-fjkrzx5:/g/g90/US
↪  ER/src/spack/share/spack/modules/linux-rhel7-x86
↪  _64/py-pycparser-2.17-gcc-4.9.3-eifsucg:/g/g90/U
↪  SER/src/spack/share/spack/modules/linux-rhel7-x8
↪  6_64/py-six-1.10.0-gcc-4.9.3-z6ym6in:/g/g90/USER
↪  /src/spack/share/spack/modules/linux-rhel7-x86_6
↪  4/py-pyyaml-3.13-gcc-4.9.3-nmgcaks:/g/g90/USER/s
↪  rc/spack/share/spack/modules/linux-rhel7-x86_64/
↪  py-jsonschema-2.5.1-gcc-4.9.3-73eyv2l:/g/g90/USE
↪  R/src/spack/share/spack/modules/linux-rhel7-x86_
↪  64/py-functools32-3.2.3-2-gcc-4.9.3-3xjeigi:/g/g
↪  90/USER/src/spack/share/spack/modules/linux-rhel
↪  7-x86_64/lz4-1.8.1.2-gcc-4.9.3-p7grndy
LDMSD_SOCKPATH=/g/g90/USER/Source/local_ldms/run/ldm
↪  sd
EDITOR=/bin/vi
LANG=en_US.UTF-8
__LMOD_REF_COUNT_PYTHONPATH=/g/g90/USER/src/spack/op
↪  t/spack/linux-rhel7-x86_64/gcc-4.9.3/py-functool
↪  s32-3.2.3-2-3xjeigiqad6thua7ycs3j7ndcegeo7vb/lib
↪  /python2.7/site-packages:1;/g/g90/USER/src/spack
↪  /opt/spack/linux-rhel7-x86_64/gcc-4.9.3/py-jsons
↪  chema-2.5.1-73eyv2likbxldo7s5rqt4uqhvzga2azq/lib
↪  /python2.7/site-packages:1;/g/g90/USER/src/spack
↪  /opt/spack/linux-rhel7-x86_64/gcc-4.9.3/py-pyyam
↪  l-3.13-nmgcaksh22c5p25lwwwhgtouz6rg4uu4/lib/pyth
↪  on2.7/site-packages:1;/g/g90/USER/src/spack/opt/
↪  spack/linux-rhel7-x86_64/gcc-4.9.3/py-six-1.10.0
↪  -z6ym6ink72tpqqpaw3mnr7xxzj7ci25d/lib/python2.7/s
↪  ite-packages:1;/g/g90/USER/src/spack/opt/spack/l
↪  inux-rhel7-x86_64/gcc-4.9.3/py-pycparser-2.17-ei
↪  fsucgo4w7fjiiyfbz4gyj5wsmytkd2/lib/python2.7/sit
↪  e-packages:1;/g/g90/USER/src/spack/opt/spack/lin
↪  ux-rhel7-x86_64/gcc-4.9.3/py-cffi-1.1.2-fjkrzx5d
↪  yljwlikbjlgv7z3ibnzlc5fw/lib/python2.7/site-pack
↪  ages:1
MODULEPATH=/g/g90/USER/src/spack/share/spack/modules
↪  /linux-rhel7-x86_64:/usr/tce/modulefiles/MPI/mva
↪  pich2/2.3:/usr/tce/modulefiles/MPI/intel/18.0.1/
↪  mvapich2/2.3:/usr/tce/modulefiles/Compiler/intel
↪  /18.0.1:/collab/usr/global/tools/modulefiles/tos
↪  s_3_x86_64_ib/Core:/usr/tce/modulefiles/Core:/us
↪  r/apps/modulefiles:/usr/share/modulefiles/Linux:
↪  /usr/share/modulefiles/Core:/usr/share/lmod/lmod
↪  /modulefiles/Core
GUESTFISH_OUTPUT=\e[0m
KDEDIRS=/usr
```

```
LOADEDMODULES=intel/18.0.1:mvapich2/2.3:texlive/2016
↪  :StdEnv:python-2.7.14-gcc-4.9.3-4agkfvs:py-cffi-
↪  1.1.2-gcc-4.9.3-fjkrzx5:py-pycparser-2.17-gcc-4.
↪  9.3-eifsucg:py-six-1.10.0-gcc-4.9.3-z6ym6in:py-p
↪  yyaml-3.13-gcc-4.9.3-nmgcaks:py-jsonschema-2.5.1
↪  -gcc-4.9.3-73eyv2l:py-functools32-3.2.3-2-gcc-4.9
↪  .3-3xjeigi:lz4-1.8.1.2-gcc-4.9.3-p7grndy
_ModuleTable_Sz_=11
LMOD_CMD=/usr/share/lmod/lmod/libexec/lmod
_ModuleTable005_=Y3RpdmUiLFsidXNlck5hbWUiXT0icHktZnV
↪  uY3Rvb2xzMzItMy4yLjMtMi1nY2MtNC45LjMtM3hqZWlnaSI
↪  sfSxbInB5LWpzb25zY2hlbWEtMi41LjEtZ2NjLTQuOS4zLTc
↪  zZXl2MmwiXT17WyJmbiJdPSIvZy9nOTAvcGF0a2kxL3NyYy9
↪  zcGFjay9zaGFyZS9zcGFjay9tb2R1bGVzL2xpbnV4LXJoZWw
↪  3LXg4Nl82NC9weS1qc29uc2NoZW1hLTIuNS4xLWdjYy00Ljk
↪  uMy03M2V5djJsIixbImZ1bGxOYW1lIl09InB5LWpzb25zY2h
↪  lbWEtMi41LjEtZ2NjLTQuOS4zLTczZXl2MmwiLFsibG9hZE9
↪  yZGVyIl09MTAscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09MCx
↪  bInN0YXR1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0icHk
↪  tanNvbnNjaGVtYS0yLjUuMS1nY2MtNC45LjMtNzNleXYy
KRB5CCNAME=FILE:/tmp/krb5cc_36985_pE1fNA
HISTCONTROL=ignoredups
ENVIRONMENT=INTERACTIVE
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
LDMSD_PIDFILE=/g/g90/USER/Source/local_ldms/run/ldms
↪  d.pid
HOME=/g/g90/USER
SHLVL=2
__LMOD_REF_COUNT_PATH=/g/g90/USER/src/spack/opt/spac
↪  k/linux-rhel7-x86_64/gcc-4.9.3/lz4-1.8.1.2-p7grn
↪  dyqisjn4t6q4kiicwykmx7ppumx/bin:1;/g/g90/USER/sr
↪  c/spack/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/p
↪  y-jsonschema-2.5.1-73eyv2likbxldo7s5rqt4uqhvzga2
↪  azq/bin:1;/g/g90/USER/src/spack/opt/spack/linux-
↪  rhel7-x86_64/gcc-4.9.3/python-2.7.14-4agkfvsiaku
↪  t5hgegek4psxq7i3wp5be/bin:1;/g/g90/USER/src/spac
↪  k/bin:1;/usr/tce/packages/texlive/texlive-2016/2
↪  016/bin/x86_64-linux:1;/usr/tce/packages/mvapich
↪  2/mvapich2-2.3-intel-18.0.1/bin:1;/usr/tce/packa
↪  ges/intel/intel-18.0.1/bin:1;/usr/tce/bin:1;/usr
↪  /lib64/qt-3.3/bin:1;/usr/local/bin:1;/usr/bin:1;
↪  /usr/local/sbin:1;/usr/sbin:1
__LMOD_REF_COUNT_CPATH=/g/g90/USER/src/spack/opt/spa
↪  ck/linux-rhel7-x86_64/gcc-4.9.3/lz4-1.8.1.2-p7gr
↪  ndyqisjn4t6q4kiicwykmx7ppumx/include:1;/g/g90/US
↪  ER/src/spack/opt/spack/linux-rhel7-x86_64/gcc-4.
↪  9.3/python-2.7.14-4agkfvsiakut5hgegek4psxq7i3wp5
↪  be/include:1
_ModuleTable002_=LjAuMSIsWyJsb2FkT3JkZXIiXT0xLHByb3B
↪  UPXt9LFsic3RhY2tEZXB0aCJdPTEsWyJzdGF0dXMiXT0iYWN
↪  0aXZlIixbInVzZXJOYW1lIl09ImludGVsIix9LFsibHo0LTE
↪  uOC4xLjItZ2NjLTQuOS4zLXA3Z3JuZHkiXT17WyJmbiJdPSI
↪  vZy9nOTAvcGF0a2kxL3NyYy9zcGFjay9zaGFyZS9zcGFjay9
↪  tb2R1bGVzL2xpbnV4LXJoZWw3LXg4Nl82NC9sejQtMS44LjE
↪  uMi1nY2MtNC45LjMtcDdncm5keSIsWyJmdWxsTmFtZSJdPSJ
↪  sejQtMS44LjEuMi1nY2MtNC45LjMtcDdncm5keSIsWyJsb2F
↪  kT3JkZXIiXT0xMixwcm9wVD17fSxbInN0YWNrRGVwdGgiXT0
↪  wLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZSJdPSJ
↪  sejQtMS44LjEuMi1nY2MtNC45LjMtcDdncm5keSIsfSxt
BASH_ENV=/usr/share/lmod/lmod/init/bash
_ModuleTable008_=bGxOYW1lIl09InB5LXNpeC0xLjEwLjAtZ2N
↪  jLTQuOS4zLXo2eW02aW4iLFsibG9hZE9yZGVyIl09OCxwcm9
↪  wVD17fSxbInN0YWNrRGVwdGgiXT0wLFsic3RhdHVzIl09ImF
↪  jdGl2ZSIsWyJ1c2VyTmFtZSJdPSJweS1zaXgtMS4xMC4wLWd
↪  jYy00LjkuMy16NnltNmluIix9LFsicHl0aG9uLTIuNy4xNC1
↪  nY2MtNC45LjMtNGFna2Z2cyJdPXtbImZuIl09Ii9nL2c5MC9
↪  wYXRraTEvc3JjL3NwYWNrL3NoYXJlL3NwYWNrL21vZHVsZXM
↪  vbGludXgtcmhlbDcteDg2XzY0L3B5dGhvbi0yLjcuMTQtZ2N
↪  jLTQuOS4zLTRhZ2tmdnMiLFsiZnVsbE5hbWUiXT0icHl0aG9
↪  uLTIuNy4xNC1nY2MtNC45LjMtNGFna2Z2cyIsWyJsb2FkT3J
↪  kZXIiXT01LHByb3BUPXt9LFsic3RhY2tEZXB0aCJdPTAs
LOGNAME=USER
LMOD_arch=x86_64
PYTHONPATH=/g/g90/USER/src/spack/opt/spack/linux-rhe
↪  l7-x86_64/gcc-4.9.3/py-functools32-3.2.3-2-3xjei
↪  giqad6thua7ycs3j7ndcegeo7vb/lib/python2.7/site-p
↪  ackages:/g/g90/USER/src/spack/opt/spack/linux-rh
↪  el7-x86_64/gcc-4.9.3/py-jsonschema-2.5.1-73eyv2l
↪  ikbxldo7s5rqt4uqhvzga2azq/lib/python2.7/site-pac
↪  kages:/g/g90/USER/src/spack/opt/spack/linux-rhel
↪  7-x86_64/gcc-4.9.3/py-pyyaml-3.13-nmgcaksh22c5p2
↪  5lwwwhgtouz6rg4uu4/lib/python2.7/site-packages:/
↪  g/g90/USER/src/spack/opt/spack/linux-rhel7-x86_6
↪  4/gcc-4.9.3/py-six-1.10.0-z6ym6ink72tpqqpaw3mnr7
↪  xxzj7ci25d/lib/python2.7/site-packages:/g/g90/US
↪  ER/src/spack/opt/spack/linux-rhel7-x86_64/gcc-4.
↪  9.3/py-pycparser-2.17-eifsucgo4w7fjiiyfbz4gyj5ws
↪  mytkd2/lib/python2.7/site-packages:/g/g90/USER/s
↪  rc/spack/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/
↪  py-cffi-1.1.2-fjkrzx5dyljwlikbjlgv7z3ibnzlc5fw/l
↪  ib/python2.7/site-packages
CVS_RSH=ssh
QTLIB=/usr/lib64/qt-3.3/lib
SSH_CONNECTION=76.102.78.63 56641 134.9.50.63 22
XDG_DATA_DIRS=/g/g90/USER/.local/share/flatpak/expor
↪  ts/share:/var/lib/flatpak/exports/share:/usr/loc
↪  al/share:/usr/share
SYS_TYPE=toss_3_x86_64_ib
MODULESHOME=/usr/share/lmod/lmod
```

```
__LMOD_REF_COUNT_LIBRARY_PATH=/g/g90/USER/src/spack/
↪  opt/spack/linux-rhel7-x86_64/gcc-4.9.3/lz4-1.8.1
↪  .2-p7grndyqisjn4t6q4kiicwykmx7ppumx/lib:1;/g/g90
↪  /USER/src/spack/opt/spack/linux-rhel7-x86_64/gcc
↪  -4.9.3/py-functools32-3.2.3-2-3xjeigiqad6thua7ycs
↪  3j7ndcegeo7vb/lib:1;/g/g90/USER/src/spack/opt/sp
↪  ack/linux-rhel7-x86_64/gcc-4.9.3/py-jsonschema-2
↪  .5.1-73eyv2likbxldo7s5rqt4uqhvzga2azq/lib:1;/g/g
↪  90/USER/src/spack/opt/spack/linux-rhel7-x86_64/g
↪  cc-4.9.3/py-pyyaml-3.13-nmgcaksh22c5p25lwwwhgtou
↪  z6rg4uu4/lib:1;/g/g90/USER/src/spack/opt/spack/l
↪  inux-rhel7-x86_64/gcc-4.9.3/py-six-1.10.0-z6ym6i
↪  nk72tpqqpaw3mnr7xxzj7ci25d/lib:1;/g/g90/USER/src
↪  /spack/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/py
↪  -pycparser-2.17-eifsucgo4w7fjiiyfbz4gyj5wsmytkd2/
↪  lib:1;/g/g90/USER/src/spack/opt/spack/linux-rhel
↪  7-x86_64/gcc-4.9.3/py-cffi-1.1.2-fjkrzx5dyljwlik
↪  bjlgv7z3ibnzlc5fw/lib:1;/g/g90/USER/src/spack/op
↪  t/spack/linux-rhel7-x86_64/gcc-4.9.3/python-2.7.
↪  14-4agkfvsiakut5hgegek4psxq7i3wp5be/lib:1
LDMS_AUTH_FILE=/g/g90/USER/mysecret
LESSOPEN=||/usr/bin/lesspipe.sh %s
LMOD_SETTARG_FULL_SUPPORT=no
__LMOD_REF_COUNT_LD_LIBRARY_PATH=/g/g90/USER/src/spa
↪  ck/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/lz4-1.
↪  8.1.2-p7grndyqisjn4t6q4kiicwykmx7ppumx/lib:1;/g/
↪  g90/USER/src/spack/opt/spack/linux-rhel7-x86_64/
↪  gcc-4.9.3/py-functools32-3.2.3-2-3xjeigiqad6thua
↪  7ycs3j7ndcegeo7vb/lib:1;/g/g90/USER/src/spack/op
↪  t/spack/linux-rhel7-x86_64/gcc-4.9.3/py-jsonsche
↪  ma-2.5.1-73eyv2likbxldo7s5rqt4uqhvzga2azq/lib:1;
↪  /g/g90/USER/src/spack/opt/spack/linux-rhel7-x86_
↪  64/gcc-4.9.3/py-pyyaml-3.13-nmgcaksh22c5p25lwwwh
↪  gtouz6rg4uu4/lib:1;/g/g90/USER/src/spack/opt/spa
↪  ck/linux-rhel7-x86_64/gcc-4.9.3/py-six-1.10.0-z6
↪  ym6ink72tpqqpaw3mnr7xxzj7ci25d/lib:1;/g/g90/USER
↪  /src/spack/opt/spack/linux-rhel7-x86_64/gcc-4.9.
↪  3/py-pycparser-2.17-eifsucgo4w7fjiiyfbz4gyj5wsmy
↪  tkd2/lib:1;/g/g90/USER/src/spack/opt/spack/linux
↪  -rhel7-x86_64/gcc-4.9.3/py-cffi-1.1.2-fjkrzx5dylj
↪  wlikbjlgv7z3ibnzlc5fw/lib:1;/g/g90/USER/src/spac
↪  k/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/python-
↪  2.7.14-4agkfvsiakut5hgegek4psxq7i3wp5be/lib:1;/u
↪  sr/tce/packages/mvapich2/mvapich2-2.3-intel-18.0
↪  .1/lib:1;/usr/tce/packages/intel/intel-18.0.1/li
↪  b/intel64:1
PKG_CONFIG_PATH=/g/g90/USER/src/spack/opt/spack/linu
↪  x-rhel7-x86_64/gcc-4.9.3/lz4-1.8.1.2-p7grndyqisj
↪  n4t6q4kiicwykmx7ppumx/lib/pkgconfig:/g/g90/USER/
↪  src/spack/opt/spack/linux-rhel7-x86_64/gcc-4.9.3
↪  /python-2.7.14-4agkfvsiakut5hgegek4psxq7i3wp5be/
↪  lib/pkgconfig
LMOD_FULL_SETTARG_SUPPORT=no
LMOD_FAMILY_COMPILER=intel
__LMOD_REF_COUNT_INTEL_LICENSE_FILE=/usr/tce/package
↪  s/intel/intel-18.0.1/compilers_and_libraries_201
↪  8.1.163/linux/Licenses/license.client.intel.lic:1
CMAKE_PREFIX_PATH=/g/g90/USER/src/spack/opt/spack/li
↪  nux-rhel7-x86_64/gcc-4.9.3/lz4-1.8.1.2-p7grndyqi
↪  sjn4t6q4kiicwykmx7ppumx:/g/g90/USER/src/spack/op
↪  t/spack/linux-rhel7-x86_64/gcc-4.9.3/py-functool
↪  s32-3.2.3-2-3xjeigiqad6thua7ycs3j7ndcegeo7vb:/g/
↪  g90/USER/src/spack/opt/spack/linux-rhel7-x86_64/
↪  gcc-4.9.3/py-jsonschema-2.5.1-73eyv2likbxldo7s5r
↪  qt4uqhvzga2azq:/g/g90/USER/src/spack/opt/spack/l
↪  inux-rhel7-x86_64/gcc-4.9.3/py-pyyaml-3.13-nmgca
↪  ksh22c5p25lwwwhgtouz6rg4uu4:/g/g90/USER/src/spac
↪  k/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/py-six-
↪  1.10.0-z6ym6ink72tpqqpaw3mnr7xxzj7ci25d:/g/g90/U
↪  SER/src/spack/opt/spack/linux-rhel7-x86_64/gcc-4
↪  .9.3/py-pycparser-2.17-eifsucgo4w7fjiiyfbz4gyj5w
↪  smytkd2:/g/g90/USER/src/spack/opt/spack/linux-rh
↪  el7-x86_64/gcc-4.9.3/py-cffi-1.1.2-fjkrzx5dyljwl
↪  ikbjlgv7z3ibnzlc5fw:/g/g90/USER/src/spack/opt/sp
↪  ack/linux-rhel7-x86_64/gcc-4.9.3/python-2.7.14-4
↪  agkfvsiakut5hgegek4psxq7i3wp5be
__LMOD_REF_COUNT_PKG_CONFIG_PATH=/g/g90/USER/src/spa
↪  ck/opt/spack/linux-rhel7-x86_64/gcc-4.9.3/lz4-1.
↪  8.1.2-p7grndyqisjn4t6q4kiicwykmx7ppumx/lib/pkgco
↪  nfig:1;/g/g90/USER/src/spack/opt/spack/linux-rhe
↪  l7-x86_64/gcc-4.9.3/python-2.7.14-4agkfvsiakut5h
↪  gegek4psxq7i3wp5be/lib/pkgconfig:1
QT_PLUGIN_PATH=/usr/lib64/kde4/plugins:/usr/lib/kde4
↪  /plugins
LMOD_DIR=/usr/share/lmod/lmod/libexec
__LMOD_REF_COUNT_MANPATH=/g/g90/USER/src/spack/opt/s
↪  pack/linux-rhel7-x86_64/gcc-4.9.3/lz4-1.8.1.2-p7
↪  grndyqisjn4t6q4kiicwykmx7ppumx/share/man:1;/g/g9
↪  0/USER/src/spack/opt/spack/linux-rhel7-x86_64/gc
↪  c-4.9.3/python-2.7.14-4agkfvsiakut5hgegek4psxq7i
↪  3wp5be/share/man:1;/usr/tce/packages/mvapich2/mv
↪  apich2-2.3-intel-18.0.1/man:1;/usr/tce/packages/
↪  intel/intel-18.0.1/man/common:1;/usr/tce/man:1;/
↪  usr/share/lmod/lmod/share/man:1;/usr/man:1;/usr/
↪  share/man:1;/usr/local/man:1;/usr/X11R6/man:1;/u
↪  sr/lib64/mvapich/default/man:1
_ModuleTable006_=bCIsfSxbInB5LXB5Y3BhcnNlci0yLjE3LWd
↪  jYy00LjkuMy1laWZzdWNnIl09e1siZm4iXT0iL2cvZzkwL3B
↪  hdGtpMS9zcmMvc3BhY2svc2hhcmUvc3BhY2svbW9kdWxlcy9
↪  saW51eC1yaGVsNy14ODZfNjQvcHktcHljcGFyc2VyLTIuMTc
↪  tZ2NjLTQuOS4zLWVpZnN1Y2ciLFsiZnVsbE5hbWUiXT0icHk
↪  tcHljcGFyc2VyLTIuMTctZ2NjLTQuOS4zLWVpZnN1Y2ciLFs
↪  ibG9hZE9yZGVyIl09Nyxwcm9wVD17fSxbInN0YWNrRGVwdGg
↪  iXT0wLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZSJ
↪  dPSJweS1weWNwYXJzZXItMi4xNy1nY2MtNC45LjMtZWlmc3V
↪  jZyIsfSxbInB5LXB5eWFtbC0zLjEzLWdjYy00LjkuMy1ubWd
↪  jYWtzIl09e1siZm4iXT0iL2cvZzkwL3BhdGtpMS9zcmMv
GUESTFISH_RESTORE=\e[0m
HISTFILE=/g/g90/USER/.bash_history
LMOD_COLORIZE=yes
LMOD_FAMILY_MPI=mvapich2
DK_NODE=/g/g90/USER/src/spack/share/spack/dotkit/lin
↪  ux-rhel7-x86_64
```

```
BASH_FUNC_module()=() {  eval $($LMOD_CMD bash "$@")
↪  && eval $(${LMOD_SETTARG_CMD:-:} -s sh)
}
BASH_FUNC_spack()=() {  if [ -n "${ZSH_VERSION:-}" ];
↪  then
 emulate -L sh;
 fi;
 args=("$@");
 _sp_flags="";
 while [[ "$1" =~ ^- ]]; do
 _sp_flags="$_sp_flags $1";
 shift;
 done;
 if [[ ( ! -z "$_sp_flags" ) && ( "$_sp_flags" =~
 ↪  '.*h.*' || "$_sp_flags" =~ '.*V.*' ) ]]; then
 command spack $_sp_flags "$@";
 return;
 fi;
 _sp_subcommand="";
 if [ -n "$1" ]; then
 _sp_subcommand="$1";
 shift;
 fi;
 _sp_spec=("$@");
 case $_sp_subcommand in
 "cd")
 _sp_arg="";
 if [ -n "$1" ]; then
 _sp_arg="$1";
 shift;
 fi;
 if [[ "$_sp_arg" = "-h" || "$_sp_arg" = "--help" ]];
 ↪  then
 command spack cd -h;
 else
 LOC="$(spack location $_sp_arg "$@")";
 if [[ -d "$LOC" ]]; then
 cd "$LOC";
 else
 return 1;
 fi;
 fi;
 return
 ;;
 "env")
 _sp_arg="";
 if [ -n "$1" ]; then
 _sp_arg="$1";
 shift;
 fi;
 if [[ "$_sp_arg" = "-h" || "$_sp_arg" = "--help" ]];
 ↪  then
 command spack env -h;
 else
 case $_sp_arg in
 activate)
 _a="$@";
 if [ -z "$1" -o "${_a#*--sh}" != "$_a" -o
 ↪  "${_a#*--csh}" != "$_a" -o "${_a#*-h}" != "$_a"
 ↪  ]; then
 command spack "${args[@]}";
 else
 eval $(command spack $_sp_flags env activate --sh
 ↪  "$@");
 fi
 ;;
 deactivate)
 if [ -n "$1" ]; then
 command spack "${args[@]}";
 else
 eval $(command spack $_sp_flags env deactivate --sh);
 fi
 ;;
 *)
 command spack "${args[@]}"
 ;;
 esac;
 fi;
 return
 ;;
 "use" | "unuse" | "load" | "unload")
 _sp_subcommand_args="";
 _sp_module_args="";
 while [[ "$1" =~ ^- ]]; do
 if [ "$1" = "-r" -o "$1" = "--dependencies" ]; then
 _sp_subcommand_args="$_sp_subcommand_args $1";
 else
 _sp_module_args="$_sp_module_args $1";
 fi;
 shift;
 done;
 _sp_spec=("$@");
 case $_sp_subcommand in
 "use")
 if _sp_full_spec=$(command spack $_sp_flags module
 ↪  dotkit find $_sp_subcommand_args
 ↪  "${_sp_spec[@]}"); then
 use $_sp_module_args $_sp_full_spec;
 else
 $(exit 1);
 fi
 ;;
 "unuse")
 if _sp_full_spec=$(command spack $_sp_flags module
 ↪  dotkit find $_sp_subcommand_args
 ↪  "${_sp_spec[@]}"); then
 unuse $_sp_module_args $_sp_full_spec;
 else
 $(exit 1);
 fi
 ;;
 "load")
```

```
 if _sp_full_spec=$(command spack $_sp_flags module
 ↪  tcl find $_sp_subcommand_args "${_sp_spec[@]}");
 ↪  then
 module load $_sp_module_args $_sp_full_spec;
 else
 $(exit 1);
 fi
 ;;
 "unload")
 if _sp_full_spec=$(command spack $_sp_flags module
 ↪  tcl find $_sp_subcommand_args "${_sp_spec[@]}");
 ↪  then
 module unload $_sp_module_args $_sp_full_spec;
 else
 $(exit 1);
 fi
 ;;
 esac
 ;;
 *)
 command spack "${args[@]}"
 ;;
 esac
}
BASH_FUNC_ml()=() {  eval $($LMOD_DIR/ml_cmd "$@")
}
+ lsb_release -a
LSB Version:        :core-4.1-amd64:core-4.1-noarch:
↪  cxx-4.1-amd64:cxx-4.1-noarch:desktop-4.1-amd64:d
↪  esktop-4.1-noarch:languages-4.1-amd64:languages-
↪  4.1-noarch:printing-4.1-amd64:printing-4.1-noarch
Distributor ID:        RedHatEnterpriseServer
Description:        Red Hat Enterprise Linux Server
↪  release 7.6 (Maipo)
Release:        7.6
Codename:        Maipo
+ uname -a
Linux catalyst160 3.10.0-957.10.1.1chaos.ch6.x86_64
↪  #1 SMP Thu Mar 14 17:57:30 PDT 2019 x86_64 x86_64
↪  x86_64 GNU/Linux
+ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                48
On-line CPU(s) list:   0-47
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 62
Model name:            Intel(R) Xeon(R) CPU E5-2695 v2
↪  @ 2.40GHz
Stepping:              4
CPU MHz:               2401.000
CPU max MHz:           2401.0000
CPU min MHz:           1200.0000
BogoMIPS:              4788.70
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0-11,24-35
NUMA node1 CPU(s):     12-23,36-47
Flags:                 fpu vme de pse tsc msr pae mce
↪  cx8 apic sep mtrr pge mca cmov pat pse36 clflush
↪  dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
↪  pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
↪  bts rep_good nopl xtopology nonstop_tsc aperfmperf
↪  eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx
↪  smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1
↪  sse4_2 x2apic popcnt tsc_deadline_timer aes xsave
↪  avx f16c rdrand lahf_lm epb ssbd ibrs ibpb stibp
↪  tpr_shadow vnmi flexpriority ept vpid fsgsbase
↪  smep erms xsaveopt dtherm ida arat pln pts
↪  spec_ctrl intel_stibp flush_l1d
+ cat /proc/meminfo
MemTotal:       131592856 kB
MemFree:        61649204 kB
MemAvailable:   116663548 kB
Buffers:           10380 kB
Cached:         52193944 kB
SwapCached:            0 kB
Active:         14690904 kB
Inactive:       39828752 kB
Active(anon):    2671856 kB
Inactive(anon):   269928 kB
Active(file):   12019048 kB
Inactive(file): 39558824 kB
Unevictable:       11380 kB
Mlocked:           11412 kB
SwapTotal:       1951740 kB
SwapFree:        1951740 kB
Dirty:               108 kB
Writeback:             4 kB
AnonPages:       2327124 kB
Mapped:           178368 kB
Shmem:            618664 kB
Slab:           13536168 kB
SReclaimable:    4058368 kB
SUnreclaim:      9477800 kB
KernelStack:       22304 kB
PageTables:        30408 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    67748168 kB
Committed_AS:    3758436 kB
VmallocTotal:   34359738367 kB
VmallocUsed:     1439404 kB
```

```
VmallocChunk:   34291153764 kB
HardwareCorrupted:     0 kB
AnonHugePages:   1783808 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:     2772572 kB
DirectMap2M:    50624512 kB
DirectMap1G:    82837504 kB
+ inxi -F -c0
collect.sh: line 14: inxi: command not found
+ lsblk -a
NAME      MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda         8:0    0  1.8T  0 disk
|-sda1      8:1    0  1.9G  0 part  [SWAP]
|-sda2      8:2    0  1.9G  0 part  /localdisk
`-sda3      8:3    0  1.8T  0 part  /tmp
sdb         8:16   0 26.1G  1 disk
`-rootdev 253:0    0 26.1G  1 mpath
sdc         8:32   0 26.1G  1 disk
`-rootdev 253:0    0 26.1G  1 mpath
+ lsscsi -s
[6:0:0:0]    disk    ATA      WDC WD2000FYYZ-0 1K01
↪  /dev/sda   2.00TB
[8:0:0:0]    disk    LIO-ORG  FILEIO           4.0
↪  /dev/sdb   28.0GB
[9:0:0:0]    disk    LIO-ORG  FILEIO           4.0
↪  /dev/sdc   28.0GB
+ module list
++ /usr/share/lmod/lmod/libexec/lmod bash list

Currently Loaded Modules:
  1) intel/18.0.1   5) python-2.7.14-gcc-4.9.3-4agkfvs
  ↪        9) py-pyyaml-3.13-gcc-4.9.3-nmgcaks
  2) mvapich2/2.3   6) py-cffi-1.1.2-gcc-4.9.3-fjkrzx5
  ↪       10) py-jsonschema-2.5.1-gcc-4.9.3-73eyv2l
  3) texlive/2016   7)
  ↪  py-pycparser-2.17-gcc-4.9.3-eifsucg  11)
  ↪  py-functools32-3.2.3-2-gcc-4.9.3-3xjeigi
  4) StdEnv         8) py-six-1.10.0-gcc-4.9.3-z6ym6in
  ↪       12) lz4-1.8.1.2-gcc-4.9.3-p7grndy

+ eval 'MODULEPATH=/g/g90/patki1/src/spack/share/spa
↪  ck/modules/linux-rhel7-x86_64:/usr/tce/modulefil
↪  es/MPI/mvapich2/2.3:/usr/tce/modulefiles/MPI/int
↪  el/18.0.1/mvapich2/2.3:/usr/tce/modulefiles/Comp
↪  iler/intel/18.0.1:/collab/usr/global/tools/modul
↪  efiles/toss_3_x86_64_ib/Core:/usr/tce/modulefile
↪  s/Core:/usr/apps/modulefiles:/usr/share/modulefi
↪  les/Linux:/usr/share/modulefiles/Core:/usr/share
↪  /lmod/lmod/modulefiles/Core;' export
↪  'MODULEPATH;' '_ModuleTable001_=X01vZHVsZVRhYmxl
↪  Xz17WyJNVHZlcnNpb24iXT0zLFsiY19yZWJ1aWxkVGltZSJd
↪  PWZhbHNlLFsiY19zaG9ydFRpbWUiXT1mYWxzZSxkZXB0aFQ9
↪  e30sZmFtaWx5PXtbImNvbXBpbGVyIl09ImludGVsIixbIm1w
↪  aSJdPSJtdmFwaWNoMiIsfSxtVD17U3RkRW52PXtbImZuIl09
↪  Ii91c3IvdGNlL21vZHVsZWZpbGVzL0NvcmUvU3RkRW52Lmx1
↪  YSIsWyJmdWxsTmFtZSJdPSJTdGRFbnYiLFsibG9hZE9yZGVy
↪  Il09NCxwcm9wVD17fSxbInN0YWNrRGVwdGgiXT0wLFsic3Rh
↪  dHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZSJdPSJTdGRFbnYi
↪  LH0saW50ZWw9e1siZm4iXT0iL3Vzci90Y2UvbW9kdWxlZmls
↪  ZXMvQ29yZS9pbnRlbC8xOC4wLjEubHVhIixbImZ1bGxOYW1l
↪  Il09ImludGVsLzE4;' export '_ModuleTable001_;'
↪  '_ModuleTable002_=LjAuMSIsWyJsb2FkT3JkZXIiXT0xLH
↪  Byb3BUPXt9LFsic3RhY2tEZXB0aCJdPTEsWyJzdGF0dXMiXT
↪  0iYWN0aXZlIixbInVzZXJOYW1lIl09ImludGVsIix9LFsibH
↪  o0LTEuOC4xLjItZ2NjLTQuOS4zLXA3Z3JuZHkiXT17WyJmbi
↪  JdPSIvZy9nOTAvcGF0a2kxL3NyYy9zcGFjay9zaGFyZS9zcG
↪  Fjay9tb2R1bGVzL2xpbnV4LXJoZWw3LXg4Nl82NC9sejQtMS
↪  44LjEuMi1nY2MtNC45LjMtcDdncm5keSIsWyJmdWxsTmFtZS
↪  JdPSJsejQtMS44LjEuMi1nY2MtNC45LjMtcDdncm5keSIsWy
↪  Jsb2FkT3JkZXIiXT0xMixwcm9wVD17fSxbInN0YWNrRGVwdG
↪  giXT0wLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZS
↪  JdPSJsejQtMS44LjEuMi1nY2MtNC45LjMtcDdncm5keSIsfS
↪  xt;' export '_ModuleTable002_;'
↪  '_ModuleTable003_=dmFwaWNoMj17WyJmbiJdPSIvdXNyL3
↪  RjZS9tb2R1bGVmaWxlcy9Db21waWxlci9pbnRlbC8xOC4wLj
↪  EvbXZhcGljaDIvMi4zLmx1YSIsWyJmdWxsTmFtZSJdPSJtdm
↪  FwaWNoMi8yLjMiLFsibG9hZE9yZGVyIl09Mixwcm9wVD17fS
↪  xbInN0YWNrRGVwdGgiXT0xLFsic3RhdHVzIl09ImFjdGl2ZS
↪  IsWyJ1c2VyTmFtZSJdPSJtdmFwaWNoMi8yLjMiLH0sWyJweS
↪  1jZmZpLTEuMS4yLWdjYy00LjkuMy1mamtyeng1Il09e1siZm
↪  4iXT0iL2cvZzkwL3BhdGtpMS9zcmMvc3BhY2svc2hhcmUvc3
↪  BhY2svbW9kdWxlcy9saW51eC1yaGVsNy14ODZfNjQvcHktY2
↪  ZmaS0xLjEuMi1nY2MtNC45LjMtZmprcnp4NSIsWyJmdWxsTm
↪  FtZSJdPSJweS1jZmZpLTEuMS4yLWdjYy00LjkuMy1mamtyen
↪  g1;' export
↪  '_ModuleTable003_;'
```

```
'_ModuleTable004_=IixbImxvYWRPcmRlciJdPTYscHJvcFQ9e
↪  30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3Rp
↪  dmUiLFsidXNlck5hbWUiXT0icHktY2ZmaS0xLjEuMi1nY2M
↪  tNC45LjMtZmprcnp4NSIsfSxbInB5LWZ1bmN0b29sczMyLT
↪  MuMi4zLTItZ2NjLTQuOS4zLTN4amVpZ2kiXT17WyJmbiJdP
↪  SIvZy9nOTAvcGF0a2kxL3NyYy9zcGFjay9zaGFyZS9zcGFj
↪  ay9tb2R1bGVzL2xpbnV4LXJoZWw3LXg4Nl82NC9weS1mdW5
↪  jdG9vbHMzMi0zLjIuMy0yLWdjYy00LjkuMy0zeGplaWdpIi
↪  xbImZ1bGxOYW1lIl09InB5LWZ1bmN0b29sczMyLTMuMi4zL
↪  TItZ2NjLTQuOS4zLTN4amVpZ2kiLFsibG9hZE9yZGVyIl09
↪  MTEscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR
↪  1cyJdPSJh;' export '_ModuleTable004_;'
↪  '_ModuleTable005_=Y3RpdmUiLFsidXNlck5hbWUiXT0ic
↪  HktZnVuY3Rvb2xzMzItMy4yLjMtMi1nY2MtNC45LjMtM3hq
↪  ZWlnaSIsfSxbInB5LWpzb25zY2hlbWEtMi41LjEtZ2NjLTQ
↪  uOS4zLTczZXl2MmwiXT17WyJmbiJdPSIvZy9nOTAvcGF0a2
↪  kxL3NyYy9zcGFjay9zaGFyZS9zcGFjay9tb2R1bGVzL2xpb
↪  nV4LXJoZWw3LXg4Nl82NC9weS1qc29uc2NoZW1hLTIuNS4x
↪  LWdjYy00LjkuMy03M2V5djJsIixbImZ1bGxOYW1lIl09InB
↪  5LWpzb25zY2hlbWEtMi41LjEtZ2NjLTQuOS4zLTczZXl2Mm
↪  wiLFsibG9hZE9yZGVyIl09MTAscHJvcFQ9e30sWyJzdGFja
↪  0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiLFsidXNl
↪  ck5hbWUiXT0icHktanNvbnNjaGVtYS0yLjUuMS1nY2MtNC4
↪  5LjMtNzNleXYy;' export '_ModuleTable005_;'
↪  '_ModuleTable006_=bCIsfSxbInB5LXB5Y3BhcnNlci0yL
↪  jE3LWdjYy00LjkuMy1laWZzdWNnIl09e1siZm4iXT0iL2cv
↪  ZzkwL3BhdGtpMS9zcmMvc3BhY2svc2hhcmUvc3BhY2svbW9
↪  kdWxlcy9saW51eC1yaGVsNy14ODZfNjQvcHktcHljcGFyc2
↪  VyLTIuMTctZ2NjLTQuOS4zLWVpZnN1Y2ciLFsiZnVsbE5hb
↪  WUiXT0icHktcHljcGFyc2VyLTIuMTctZ2NjLTQuOS4zLWVp
↪  ZnN1Y2ciLFsibG9hZE9yZGVyIl09Nyxwcm9wVD17fSxbInN
↪  0YWNrRGVwdGgiXT0wLFsic3RhdHVzIl09ImFjdGl2ZSIsWy
↪  J1c2VyTmFtZSJdPSJweS1weWNwYXJzZXItMi4xNy1nY2MtN
↪  C45LjMtZWlmc3VjZyIsfSxbInB5LXB5eWFtbC0zLjEzLWdj
↪  Yy00LjkuMy1ubWdjYWtzIl09e1siZm4iXT0iL2cvZzkwL3B
↪  hdGtpMS9zcmMv;' export '_ModuleTable006_;'
↪  '_ModuleTable007_=c3BhY2svc2hhcmUvc3BhY2svbW9kd
↪  Wxlcy9saW51eC1yaGVsNy14ODZfNjQvcHktcHl5YW1sLTMu
↪  MTMtZ2NjLTQuOS4zLW5tZ2Nha3MiLFsiZnVsbE5hbWUiXT0
↪  icHktcHl5YW1sLTMuMTMtZ2NjLTQuOS4zLW5tZ2Nha3MiLF
↪  sibG9hZE9yZGVyIl09OSxwcm9wVD17fSxbInN0YWNrRGVwd
↪  GgiXT0wLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFt
↪  ZSJdPSJweS1weXlhbWwtMy4xMy1nY2MtNC45LjMtbm1nY2F
↪  rcyIsfSxbInB5LXNpeC0xLjEwLjAtZ2NjLTQuOS4zLXo2eW
↪  02aW4iXT17WyJmbiJdPSIvZy9nOTAvcGF0a2kxL3NyYy9zc
↪  GFjay9zaGFyZS9zcGFjay9tb2R1bGVzL2xpbnV4LXJoZWw3
↪  LXg4Nl82NC9weS1zaXgtMS4xMC4wLWdjYy00LjkuMy16Nnl
↪  tNmluIixbImZ1;' export
↪  '_ModuleTable007_;'
 '_ModuleTable008_=bGxOYW1lIl09InB5LXNpeC0xLjEwLjAtZ
 ↪  2NjLTQuOS4zLXo2eW02aW4iLFsibG9hZE9yZGVyIl09OCxw
 ↪  cm9wVD17fSxbInN0YWNrRGVwdGgiXT0wLFsic3RhdHVzIl0
 ↪  9ImFjdGl2ZSIsWyJ1c2VyTmFtZSJdPSJweS1zaXgtMS4xMC
 ↪  4wLWdjYy00LjkuMy16NnltNmluIix9LFsicHl0aG9uLTIuN
 ↪  y4xNC1nY2MtNC45LjMtNGFna2Z2cyJdPXtbImZuIl09Ii9n
 ↪  L2c5MC9wYXRraTEvc3JjL3NwYWNrL3NoYXJlL3NwYWNrL21
 ↪  vZHVsZXMvbGludXgtcmhlbDcteDg2XzY0L3B5dGhvbi0yLj
 ↪  cuMTQtZ2NjLTQuOS4zLTRhZ2tmdnMiLFsiZnVsbE5hbWUiX
 ↪  T0icHl0aG9uLTIuNy4xNC1nY2MtNC45LjMtNGFna2Z2cyIs
 ↪  WyJsb2FkT3JkZXIiXT01LHByb3BUPXt9LFsic3RhY2tEZXB
 ↪  0aCJdPTAs;' export '_ModuleTable008_;'
 ↪  '_ModuleTable009_=WyJzdGF0dXMiXT0iYWN0aXZlIixbI
 ↪  nVzZXJOYW1lIl09InB5dGhvbi0yLjcuMTQtZ2NjLTQuOS4z
 ↪  LTRhZ2tmdnMiLH0sdGV4bGl2ZT17WyJmbiJdPSIvdXNyL3R
 ↪  jZS9tb2R1bGVmaWxlcy9Db3JlL3RleGxpdmUvMjAxNi5sdW
 ↪  EiLFsiZnVsbE5hbWUiXT0idGV4bGl2ZS8yMDE2IixbImxvY
 ↪  WRPcmRlciJdPTMscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09
 ↪  MSxbInN0YXR1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0
 ↪  idGV4bGl2ZS8yMDE2Iix9LH0sbXBhdGhBPXsiL2cvZzkwL3
 ↪  BhdGtpMS9zcmMvc3BhY2svc2hhcmUvc3BhY2svbW9kdWxlc
 ↪  y9saW51eC1yaGVsNy14ODZfNjQiLCIvdXNyL3RjZS9tb2R1
 ↪  bGVmaWxlcy9NUEkvbXZhcGljaDIvMi4zIiwiL3Vzci90Y2U
 ↪  vbW9kdWxlZmls;' export '_ModuleTable009_;'
 ↪  '_ModuleTable010_=ZXMvTVBJL2ludGVsLzE4LjAuMS9td
 ↪  mFwaWNoMi8yLjMiLCIvdXNyL3RjZS9tb2R1bGVmaWxlcy9D
 ↪  b21waWxlci9pbnRlbC8xOC4wLjEiLCIvY29sbGFiL3Vzci9
 ↪  nbG9iYWwvdG9vbHMvbW9kdWxlZmlsZXMvdG9zc18zX3g4Nl
 ↪  82NF9pYi9Db3JlIiwiL3Vzci90Y2UvbW9kdWxlZmlsZXMvQ
 ↪  29yZSIsIi91c3IvYXBwcy9tb2R1bGVmaWxlcyIsIi91c3Iv
 ↪  c2hhcmUvbW9kdWxlZmlsZXMvTGludXgiLCIvdXNyL3NoYXJ
 ↪  lL21vZHVsZWZpbGVzL0NvcmUiLCIvdXNyL3NoYXJlL2xtb2
 ↪  QvbG1vZC9tb2R1bGVmaWxlcy9Db3JlIix9LFsic3lzdGVtQ
 ↪  mFzZU1QQVRIIl09Ii91c3IvdGNlL21vZHVsZWZpbGVzL0Nv
 ↪  cmU6L3Vzci9hcHBzL21vZHVsZWZpbGVzOi91c3Ivc2hhcmU
 ↪  vbW9kdWxlZmls;' export '_ModuleTable010_;'
 ↪  '_ModuleTable011_=ZXMvTGludXg6L3Vzci9zaGFyZS9tb
 ↪  2R1bGVmaWxlcy9Db3JlOi91c3Ivc2hhcmUvbG1vZC9sbW9k
 ↪  L21vZHVsZWZpbGVzL0NvcmUiLH0=;' export
 ↪  '_ModuleTable011_;' '_ModuleTable_Sz_=11;'
 ↪  export '_ModuleTable_Sz_;'
++ MODULEPATH=/g/g90/patki1/src/spack/share/spack/mo
 ↪  dules/linux-rhel7-x86_64:/usr/tce/modulefiles/MP
 ↪  I/mvapich2/2.3:/usr/tce/modulefiles/MPI/intel/18
 ↪  .0.1/mvapich2/2.3:/usr/tce/modulefiles/Compiler/
 ↪  intel/18.0.1:/collab/usr/global/tools/modulefile
 ↪  s/toss_3_x86_64_ib/Core:/usr/tce/modulefiles/Cor
 ↪  e:/usr/apps/modulefiles:/usr/share/modulefiles/L
 ↪  inux:/usr/share/modulefiles/Core:/usr/share/lmod
 ↪  /lmod/modulefiles/Core
++ export MODULEPATH
```

```
++ _ModuleTable001_=X01vZHVsZVRhYmxlXz17WyJNVHZlcnNp
↪  b24iXT0zLFsiY19yZWJ1aWxkVGltZSJdPWZhbHNlLFsiY19z
↪  aG9ydFRpbWUiXT1mYWxzZSxkZXB0aFQ9e30sZmFtaWx5PXtb
↪  ImNvbXBpbGVyIl09ImludGVsIixbIm1waSJdPSJtdmFwaWNo
↪  MiIsfSxtVD17U3RkRW52PXtbImZuIl09Ii91c3IvdGNlL21v
↪  ZHVsZWZpbGVzL0NvcmUvU3RkRW52Lmx1YSIsWyJmdWxsTmFt
↪  ZSJdPSJTdGRFbnYiLFsibG9hZE9yZGVyIl09NCxwcm9wVD17
↪  fSxbInN0YWNrRGVwdGgiXT0wLFsic3RhdHVzIl09ImFjdGl2
↪  ZSIsWyJ1c2VyTmFtZSJdPSJTdGRFbnYiLH0saW50ZWw9e1si
↪  Zm4iXT0iL3Vzci90Y2UvbW9kdWxlZmlsZXMvQ29yZS9pbnRl
↪  bC8xOC4wLjEubHVhIixbImZ1bGxOYW1lIl09ImludGVsLzE4
++ export _ModuleTable001_
++ _ModuleTable002_=LjAuMSIsWyJsb2FkT3JkZXIiXT0xLHBy
↪  b3BUPXt9LFsic3RhY2tEZXB0aCJdPTEsWyJzdGF0dXMiXT0i
↪  YWN0aXZlIixbInVzZXJOYW1lIl09ImludGVsIix9LFsibHo0
↪  LTEuOC4xLjItZ2NjLTQuOS4zLXA3Z3JuZHkiXT17WyJmbiJd
↪  PSIvZy9nOTAvcGF0a2kxL3NyYy9zcGFjay9zaGFyZS9zcGFj
↪  ay9tb2R1bGVzL2xpbnV4LXJoZWw3LXg4Nl82NC9sejQtMS44
↪  LjEuMi1nY2MtNC45LjMtcDdncm5keSIsWyJmdWxsTmFtZSJd
↪  PSJsejQtMS44LjEuMi1nY2MtNC45LjMtcDdncm5keSIsWyJs
↪  b2FkT3JkZXIiXT0xMixwcm9wVD17fSxbInN0YWNrRGVwdGgi
↪  XT0wLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZSJd
↪  PSJsejQtMS44LjEuMi1nY2MtNC45LjMtcDdncm5keSIsfSxt
++ export _ModuleTable002_
++ _ModuleTable003_=dmFwaWNoMj17WyJmbiJdPSIvdXNyL3Rj
↪  ZS9tb2R1bGVmaWxlcy9Db21waWxlci9pbnRlbC8xOC4wLjEv
↪  bXZhcGljaDIvMi4zLmx1YSIsWyJmdWxsTmFtZSJdPSJtdmFw
↪  aWNoMi8yLjMiLFsibG9hZE9yZGVyIl09Mixwcm9wVD17fSxb
↪  InN0YWNrRGVwdGgiXT0xLFsic3RhdHVzIl09ImFjdGl2ZSIs
↪  WyJ1c2VyTmFtZSJdPSJtdmFwaWNoMi8yLjMiLH0sWyJweS1j
↪  ZmZpLTEuMS4yLWdjYy00LjkuMy1mamtyeng1Il09e1siZm4i
↪  XT0iL2cvZzkwL3BhdGtpMS9zcmMvc3BhY2svc2hhcmUvc3Bh
↪  Y2svbW9kdWxlcy9saW51eC1yaGVsNy14ODZfNjQvcHktY2Zm
↪  aS0xLjEuMi1nY2MtNC45LjMtZmprcnp4NSIsWyJmdWxsTmFt
↪  ZSJdPSJweS1jZmZpLTEuMS4yLWdjYy00LjkuMy1mamtyeng1
++ export _ModuleTable003_
++ _ModuleTable004_=IixbImxvYWRPcmRlciJdPTYscHJvcFQ9
↪  e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3Rp
↪  dmUiLFsidXNlck5hbWUiXT0icHktY2ZmaS0xLjEuMi1nY2Mt
↪  NC45LjMtZmprcnp4NSIsfSxbInB5LWZ1bmN0b29sczMyLTMu
↪  Mi4zLTItZ2NjLTQuOS4zLTN4amVpZ2kiXT17WyJmbiJdPSIv
↪  Zy9nOTAvcGF0a2kxL3NyYy9zcGFjay9zaGFyZS9zcGFjay9t
↪  b2R1bGVzL2xpbnV4LXJoZWw3LXg4Nl82NC9weS1mdW5jdG9v
↪  bHMzMi0zLjIuMy0yLWdjYy00LjkuMy0zeGplaWdpIixbImZ1
↪  bGxOYW1lIl09InB5LWZ1bmN0b29sczMyLTMuMi4zLTItZ2Nj
↪  LTQuOS4zLTN4amVpZ2kiLFsibG9hZE9yZGVyIl09MTEscHJv
↪  cFQ9e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJh
++ export _ModuleTable004_
++ _ModuleTable005_=Y3RpdmUiLFsidXNlck5hbWUiXT0icHkt
↪  ZnVuY3Rvb2xzMzItMy4yLjMtMi1nY2MtNC45LjMtM3hqZWln
↪  aSIsfSxbInB5LWpzb25zY2hlbWEtMi41LjEtZ2NjLTQuOS4z
↪  LTczZXl2MmwiXT17WyJmbiJdPSIvZy9nOTAvcGF0a2kxL3Ny
↪  Yy9zcGFjay9zaGFyZS9zcGFjay9tb2R1bGVzL2xpbnV4LXJo
↪  ZWw3LXg4Nl82NC9weS1qc29uc2NoZW1hLTIuNS4xLWdjYy00
↪  LjkuMy03M2V5djJsIixbImZ1bGxOYW1lIl09InB5LWpzb25z
↪  Y2hlbWEtMi41LjEtZ2NjLTQuOS4zLTczZXl2MmwiLFsibG9h
↪  ZE9yZGVyIl09MTAscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09
↪  MCxbInN0YXR1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0i
↪  cHktanNvbnNjaGVtYS0yLjUuMS1nY2MtNC45LjMtNzNleXYy
++ export _ModuleTable005_
++ _ModuleTable006_=bCIsfSxbInB5LXB5Y3BhcnNlci0yLjE3
↪  LWdjYy00LjkuMy1laWZzdWNnIl09e1siZm4iXT0iL2cvZzkw
↪  L3BhdGtpMS9zcmMvc3BhY2svc2hhcmUvc3BhY2svbW9kdWxl
↪  cy9saW51eC1yaGVsNy14ODZfNjQvcHktcHljcGFyc2VyLTIu
↪  MTctZ2NjLTQuOS4zLWVpZnN1Y2ciLFsiZnVsbE5hbWUiXT0i
↪  cHktcHljcGFyc2VyLTIuMTctZ2NjLTQuOS4zLWVpZnN1Y2ci
↪  LFsibG9hZE9yZGVyIl09Nyxwcm9wVD17fSxbInN0YWNrRGVw
↪  dGgiXT0wLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFt
↪  ZSJdPSJweS1weWNwYXJzZXItMi4xNy1nY2MtNC45LjMtZWlm
↪  c3VjZyIsfSxbInB5LXB5eWFtbC0zLjEzLWdjYy00LjkuMy1u
↪  bWdjYWtzIl09e1siZm4iXT0iL2cvZzkwL3BhdGtpMS9zcmMv
++ export _ModuleTable006_
++ _ModuleTable007_=c3BhY2svc2hhcmUvc3BhY2svbW9kdWxl
↪  cy9saW51eC1yaGVsNy14ODZfNjQvcHktcHl5YW1sLTMuMTMt
↪  Z2NjLTQuOS4zLW5tZ2Nha3MiLFsiZnVsbE5hbWUiXT0icHkt
↪  cHl5YW1sLTMuMTMtZ2NjLTQuOS4zLW5tZ2Nha3MiLFsibG9h
↪  ZE9yZGVyIl09OSxwcm9wVD17fSxbInN0YWNrRGVwdGgiXT0w
↪  LFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZSJdPSJw
↪  eS1weXlhbWwtMy4xMy1nY2MtNC45LjMtbm1nY2FrcyIsfSxb
↪  InB5LXNpeC0xLjEwLjAtZ2NjLTQuOS4zLXo2eW02aW4iXT17
↪  WyJmbiJdPSIvZy9nOTAvcGF0a2kxL3NyYy9zcGFjay9zaGFy
↪  ZS9zcGFjay9tb2R1bGVzL2xpbnV4LXJoZWw3LXg4Nl82NC9w
↪  eS1zaXgtMS4xMC4wLWdjYy00LjkuMy16NnltNmluIixbImZ1
↪  bGxOYW1lIl09InB5LWpzb25zY2hlbWEtMi41LjEtZ2NjLTQu
↪  OS4zLTRhZ2tmdnMiLFsiZnVsbE5hbWUiXT0icHl0IixbImZk
↪  aG9uLTIuNy4xNC1nY2MtNC45LjMtNGFna2Z2cyIsWyJsb2Fk
↪  T3JkZXIiXT01LHByb3BUPXt9LFsic3RhY2tEZXB0aCJdPTAs
++ export _ModuleTable008_
```

```
++ _ModuleTable009_=WyJzdGF0dXMiXT0iYWN0aXZlIixbInVz
↪  ZXJOYW1lIl09InB5dGhvbi0yLjcuMTQtZ2NjLTQuOS4zLTRh
↪  Z2tmdnMiLH0sdGV4bGl2ZT17WyJmbiJdPSIvdXNyL3RjZS9t
↪  b2R1bGVmaWxlcy9Db3JlL3RleGxpdmUvMjAxNi5sdWEiLFsi
↪  ZnVsbE5hbWUiXT0idGV4bGl2ZS8yMDE2IixbImxvYWRPcmRl
↪  ciJdPTMscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09MSxbInN0
↪  YXR1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0idGV4bGl2
↪  ZS8yMDE2Iix9LH0sbXBhdGhBPXsiL2cvZzkwL3BhdGtpMS9z
↪  cmMvc3BhY2svc2hhcmUvc3BhY2svbW9kdWxlcy9saW51eC1y
↪  aGVsNy14ODZfNjQiLCIvdXNyL3RjZS9tb2R1bGVmaWxlcy9N
↪  UEkvbXZhcGljaDIvMi4zIiwiL3Vzci90Y2UvbW9kdWxlZmls
++ export _ModuleTable009_
++ _ModuleTable010_=ZXMvTVBJL2ludGVsLzE4LjAuMS9tdmFw
↪  aWNoMi8yLjMiLCIvdXNyL3RjZS9tb2R1bGVmaWxlcy9Db21w
↪  aWxlci9pbnRlbC8xOC4wLjEiLCIvY29sbGFiL3Vzci9nbG9i
↪  YWwvdG9vbHMvbW9kdWxlZmlsZXMvdG9zc18zX3g4Nl82NF9p
↪  Yi9Db3JlIiwiL3Vzci90Y2UvbW9kdWxlZmlsZXMvQ29yZSIs
↪  Ii91c3IvYXBwcy9tb2R1bGVmaWxlcyIsIi91c3Ivc2hhcmUv
↪  bW9kdWxlZmlsZXMvTGludXgiLCIvdXNyL3NoYXJlL21vZHVs
↪  ZWZpbGVzL0NvcmUiLCIvdXNyL3NoYXJlL2xtb2QvbG1vZC9t
↪  b2R1bGVmaWxlcy9Db3JlIix9LFsic3lzdGVtQmFzZU1QQVRI
↪  Il09Ii91c3IvdGNlL21vZHVsZWZpbGVzL0NvcmU6L3Vzci9h
↪  cHBzL21vZHVsZWZpbGVzOi91c3Ivc2hhcmUvbW9kdWxlZmls
++ export _ModuleTable010_
++ _ModuleTable011_=ZXMvTGludXg6L3Vzci9zaGFyZS9tb2R1
↪  bGVmaWxlcy9Db3JlOi91c3Ivc2hhcmUvbG1vZC9sbW9kL21v
↪  ZHVsZWZpbGVzL0NvcmUiLH0=
++ export _ModuleTable011_
++ _ModuleTable_Sz_=11
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ nvidia-smi
NVIDIA-SMI has failed because it couldn't communicate
↪  with the NVIDIA driver. Make sure that the latest
↪  NVIDIA driver is installed and running.

+ lshw -short -quiet -sanitize
+ cat
WARNING: you should run this program as super-user.
H/W path          Device  Class          Description
==========================================================
                          system         Computer
/0                        bus            Motherboard
/0/0                      memory         128GiB System
↪  memory
/0/6                      processor      Intel(R)
↪  Xeon(R) CPU E5-2695 v2 @ 2.40GHz
/0/7                      processor      Intel(R)
↪  Xeon(R) CPU E5-2695 v2 @ 2.40GHz
/0/100                    bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 DMI2
/0/100/1                  bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 1a
/0/100/1/0                bridge         C608/C606/X79
↪  series chipset PCI Express Upstream Port
/0/100/1/0/8              bridge         C608/C606/X79
↪  series chipset PCI Express Virtual Switch Port
/0/100/1/0/8/0            storage        C600/X79 series
↪  chipset Dual 4-Port SATA Storage Control Unit
/0/100/1/0/8/0.3          bus            C600/X79
↪  series chipset SMBus Controller 0
/0/100/1/0/8/0.4          bus            C608/C606/X79
↪  series chipset SMBus Controller 1
/0/100/1.1                bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 1b
/0/100/1.1/0      eno1    network        I350 Gigabit
↪  Network Connection
/0/100/1.1/0.1    eno2    network        I350 Gigabit
↪  Network Connection
/0/100/1.1/0.2    pub     network        I350 Gigabit
↪  Network Connection
/0/100/1.1/0.3    eno4    network        I350 Gigabit
↪  Network Connection
/0/100/2                  bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 2a
/0/100/2.2                bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 2c
/0/100/3                  bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 3a
/0/100/3/0        hsi0    bus            IBA7322 QDR
↪  InfiniBand HCA
/0/100/3.2                bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 3c
/0/100/4                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 0
/0/100/4.1                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 1
/0/100/4.2                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 2
/0/100/4.3                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 3
/0/100/4.4                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 4
/0/100/4.5                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 5
/0/100/4.6                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 6
/0/100/4.7                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 7
/0/100/5                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 VTd/Memory Map/Misc
/0/100/5.1                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Memory Hotplug
/0/100/5.2                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 IIO RAS
/0/100/5.4                generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 IOAPIC
/0/100/16                 communication  C600/X79
↪  series chipset MEI Controller #1
```

```
/0/100/16.1               communication  C600/X79
↪  series chipset MEI Controller #2
/0/100/1a                 bus            C600/X79
↪  series chipset USB2 Enhanced Host Controller #2
/0/100/1c                 bridge         C600/X79
↪  series chipset PCI Express Root Port 1
/0/100/1c.7               bridge         C600/X79
↪  series chipset PCI Express Root Port 8
/0/100/1c.7/0             display        MGA G200e
↪  [Pilot] ServerEngines (SEP1)
/0/100/1d                 bus            C600/X79
↪  series chipset USB2 Enhanced Host Controller #1
/0/100/1e                 bridge         82801 PCI
↪  Bridge
/0/100/1f                 bridge         C600/X79
↪  series chipset LPC Controller
/0/100/1f.2               storage        C600/X79
↪  series chipset 6-Port SATA AHCI Controller
/0/100/1f.3               bus            C600/X79
↪  series chipset SMBus Host Controller
/0/8                      generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 QPI Link 0
/0/9                      generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 QPI Link 1
/0/a                      generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Power Control Unit 0
/0/b                      generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Power Control Unit 1
/0/c                      generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Power Control Unit 2
/0/d                      generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Power Control Unit 3
/0/e                      generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 UBOX Registers
/0/f                      generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 UBOX Registers
/0/10                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/11                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/12                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/13                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/14                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/15                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/16                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/17                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/18                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/19                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/1a                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/1b                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/1c                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Home Agent 0
/0/1d                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Home Agent 0
/0/1e                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Target Address/Thermal Registers
/0/1f                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 RAS Registers
/0/20                    generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 0
↪  Channel Target Address Decoder Registers
/0/21                     generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 0
↪  Channel Target Address Decoder Registers
/0/22                     generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 0
↪  Channel Target Address Decoder Registers
/0/23                     generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 0
↪  Channel Target Address Decoder Registers
/0/24                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 Thermal Control 0
/0/25                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 Thermal Control 1
/0/26                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 ERROR Registers 0
/0/27                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 ERROR Registers 1
/0/28                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 Thermal Control 2
/0/29                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 Thermal Control 3
/0/2a                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 ERROR Registers 2
/0/2b                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 ERROR Registers 3
/0/2c                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 R2PCIe
/0/2d                     generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 R2PCIe
```

```
/0/2e                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 QPI Ring Registers
/0/2f                   generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 QPI Ring Performance Ring Monitoring
/0/30                   generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 QPI Ring Performance Ring Monitoring
/0/31                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 System Address Decoder
/0/32                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Broadcast Registers
/0/33                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Broadcast Registers
/0/34                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Home Agent 1
/0/35                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Target Address/Thermal Registers
/0/36                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 RAS Registers
/0/37                   generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 1
↪  Channel Target Address Decoder Registers
/0/38                   generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 1
↪  Channel Target Address Decoder Registers
/0/39                   generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 1
↪  Channel Target Address Decoder Registers
/0/3a                   generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 1
↪  Channel Target Address Decoder Registers
/0/3b                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 Thermal Control 0
/0/3c                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 Thermal Control 1
/0/3d                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 ERROR Registers 0
/0/3e                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 ERROR Registers 1
/0/3f                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 Thermal Control 2
/0/40                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 Thermal Control 3
/0/41                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 ERROR Registers 2
/0/42                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 ERROR Registers 3
/0/1                    bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 1a
/0/2                    bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 2a
/0/2/0           hsi1   bus            IBA7322 QDR
↪  InfiniBand HCA
/0/2.2                  bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 2c
/0/3                    bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 3a
/0/3.2                  bridge         Xeon E7
↪  v2/Xeon E5 v2/Core i7 PCI Express Root Port 3c
/0/4                    generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 0
/0/4.1                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 1
/0/4.2                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 2
/0/4.3                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 3
/0/4.4                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 4
/0/4.5                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 5
/0/4.6                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 6
/0/4.7                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Crystal Beach DMA Channel 7
/0/5                    generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 VTd/Memory Map/Misc
/0/5.1                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Memory Hotplug
/0/5.2                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 IIO RAS
/0/5.4                  generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 IOAPIC
/0/43                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 QPI Link 0
/0/44                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 QPI Link 1
/0/45                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Power Control Unit 0
/0/46                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Power Control Unit 1
/0/47                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Power Control Unit 2
/0/48                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Power Control Unit 3
/0/49                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 UBOX Registers
/0/4a                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 UBOX Registers
/0/4b                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
```

```
/0/4c                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/4d                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/4e                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/4f                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/50                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/51                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/52                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/53                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/54                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/55                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/56                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Unicast Registers
/0/57                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Home Agent 0
/0/58                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Home Agent 0
/0/59                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Target Address/Thermal Registers
/0/5a                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 RAS Registers
/0/5b                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 0
↪  Channel Target Address Decoder Registers
/0/5c                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 0
↪  Channel Target Address Decoder Registers
/0/5d                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 0
↪  Channel Target Address Decoder Registers
/0/5e                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 0
↪  Channel Target Address Decoder Registers
/0/5f                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 Thermal Control 0
/0/60                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 Thermal Control 1
/0/61                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 ERROR Registers 0
/0/62                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 ERROR Registers 1
/0/63                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 Thermal Control 2
/0/64                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 Thermal Control 3
/0/65                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 ERROR Registers 2
/0/66                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Channel 0-3 ERROR Registers 3
/0/67                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 R2PCIe
/0/68                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 R2PCIe
/0/69                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 QPI Ring Registers
/0/6a                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 QPI Ring Performance Ring Monitoring
/0/6b                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 QPI Ring Performance Ring Monitoring
/0/6c                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 System Address Decoder
/0/6d                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Broadcast Registers
/0/6e                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Broadcast Registers
/0/6f                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Home Agent 1
/0/70                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 Target Address/Thermal Registers
/0/71                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 1 RAS Registers
/0/72                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 1
↪  Channel Target Address Decoder Registers
/0/73                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 1
↪  Channel Target Address Decoder Registers
/0/74                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 1
↪  Channel Target Address Decoder Registers
/0/75                  generic       Xeon E7 v2/Xeon
↪  E5 v2/Core i7 Integrated Memory Controller 1
↪  Channel Target Address Decoder Registers
/0/76                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 Thermal Control 0
```

```
/0/77                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 Thermal Control 1
/0/78                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 ERROR Registers 0
/0/79                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 ERROR Registers 1
/0/7a                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 Thermal Control 2
/0/7b                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 Thermal Control 3
/0/7c                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 ERROR Registers 2
/0/7d                   generic        Xeon E7
↪  v2/Xeon E5 v2/Core i7 Integrated Memory
↪  Controller 0 Channel 0-3 ERROR Registers 3
/0/7e                   system         PnP device
↪  PNP0c02
/0/7f                   system         PnP device
↪  PNP0b00
/0/80                   generic        PnP device
↪  INT3f0d
/0/81                   communication  PnP device
↪  PNP0501
/0/82                   communication  PnP device
↪  PNP0501
/0/83                   system         PnP device
↪  PNP0c02
/0/84                   system         PnP device
↪  PNP0c01
```

ARTIFACT EVALUATION

Verification and validation studies: Yes. In our dataset, each test was run multiple times to ensure statistical significance. The paper describes these verification and validation steps in detail. Our paper targets the question of performance optimality and reproducibility in HPC, we ensure that our data is sanitized, and we build models based to determine influence of different parameters (e.g. power, network, concurrency) on performance. Additionally, we explore the tradeoff space between optimality and reproducibility using a novel quantifiable 'desirability' score.

Accuracy and precision of timings: Yes, precision of timing was tuned and verified multiple times through different sources (e.g. application reported time, profiler time, and system time). Our results report the application reported time in all cases.

Used manufactured solutions or spectral properties: No.

Quantified the sensitivity of results to initial conditions and/or parameters of the computational environment: Yes. See comment below.

Controls, statistics, or other steps taken to make the measurements and analyses robust to variability and unknowns in the system. Our paper addresses the issue of variability by presenting a new machine learning model. We statistically analyze how performance reproducibility can be an issue in HPC and how we can quantify and understand the impact of parameters such as network, power, and concurrency on application optimality and reproducibility. We ensured that our dataset was collected in a controlled environment with all system daemons and Lustre-like services turned off.