Economic Viability of Hardware Overprovisioning in Power-Constrained High Performance Computing

Tapasya Patki†, David K. Lowenthal†, Barry L. Rountree*, Martin Schulz*, Bronis R. de Supinski*
*Lawrence Livermore National Laboratory
†Department of Computer Science, The University of Arizona

***Abstract*—Recent research has established that hardware overprovisioning can improve system power utilization as well as job throughput in power-constrained, high-performance computing environments significantly. These benefits, however, may be associated with an additional infrastructure cost, making hardware overprovisioned systems less viable economically. It is thus important to conduct a detailed cost-benefit analysis before investing in such systems at a large-scale. In this paper, we develop a model to conduct this analysis and show that for a given, fixed infrastructure cost budget and a system power budget, it is possible for hardware overprovisioned systems to lead to a net performance benefit when compared to traditional, worst-case provisioned HPC systems.**

I. Introduction

Maximizing system throughput under a limited power budget is one of the main challenges for the next generation of supercomputers. At present, supercomputers are designed to be *worst-case provisioned* with respect to power. While this enables them to simultaneously operate all components at peak power, such worst-case provisioning often leads to under-utilization of power, higher capital costs, and wasted system capacity [18], [15], [16], [7]. Recent research has established that the aforementioned concerns can be addressed by designing future supercomputers to be *hardware overprovisioned* (or, overprovisioned, for short) with respect to power [14], [8]. In such systems, it will no longer be possible to fully power all components at the same time as they will support more capacity (nodes) under a limited power budget. It has been shown that reconfiguring based on workload characteristics on overprovisioned systems can improve individual application performance under a power bound by up to 62% [14]. Furthermore, power-aware scheduling on overprovisioned systems can improve both system throughput and utilization significantly [15], [16], [7]. While the benefits of overprovisioned systems are manifold, one of the key concerns with the deployment of such systems is the infrastructure cost involved. It is essential to conduct a thorough cost-benefit analysis before investing in a large-scale overprovisioned system.

In this paper, we develop a model for comparing the costs and the benefits of a traditional worst-case provisioned system with that of an overprovisioned system. The investment cost of an overprovisioned system depends on several factors, such as the underlying processor architecture or the chosen interconnection network. The benefits, on the other hand, are strongly correlated with the nature of the workload that the system is being tailored for. We note that a given hardware cost budget provides a choice between purchasing *fewer*, high-end processors that result in a worst-case provisioned system, or *more* low-cost, previous generation processors that can be overprovisioned. Based on this premise, we develop an analytical tool that enables future HPC system designers to determine whether overprovisioning is a viable option for their site. By analyzing workload characteristics, we show that it is indeed possible to achieve better performance with overprovisioning while keeping the system acquisition cost and the system power budget constant.

The rest of the paper is organized as follows. Section II presents background and related work, and Section III explains the details of our model. In Sections IV and V, we present our results and identify the scenarios in which overprovisioning leads to a net benefit when compared to worst-case provisioning. Section VI concludes this paper.

II. Background

A system is overprovisioned with respect to power if all its components (especially nodes) cannot operate at their peak power simultaneously. The performance benefits of overprovisioning can be attributed to the observation that most HPC applications do not utilize the peak power on each node, and therefore power can be redistributed to exploit their scalability characteristics. Thus, reconfiguring dynamically based on application characteristics and choosing an ideal, application-specific *configuration* (number of nodes, cores per node and power per node) can lead to significant performance improvements under a power constraint [14]. Recent research has focused on understanding these configurations [17], [2], [3] and developing runtime systems and schedulers [13], [15], [8], [7], [18], [5] for overprovisioned systems. However, the economic viability of such systems has not been studied.

A. Understanding System Procurement

The process for procuring a supercomputing system involves releasing a public Request For Proposals (RFP), which requests details of both the hardware and associated system software from each competing vendor along with a price quote. The hardware components include the nodes (processors, memory and accelerators), the interconnection network, and the I/O subsystem. Typically, a benchmark suite with representative applications is made available by the purchaser. This is used by the vendors to provide performance

results on the proposed hardware and system software. An example of a recent RFP for the CORAL systems can be found at `https://asc.llnl.gov/CORAL/`.

B. Node Cost

In general, most server processors that are a generation older offer fairly similar features at a significantly lower unit price. Therefore, one *could* buy a supercomputer using *more* inefficient, older-generation processors, which enables overprovisioning. It is important to note that the older-generation processors can be less power efficient than the high-end processors. However, intelligently reconfiguring multiple such processors in an HPC system under a global power constraint while exploiting workload characteristics can lead to better overall performance and system power efficiency.

As an example, let us compare the Intel Xeon E5 2697 v2 (Ivy Bridge, 22nm) processor with the Intel Xeon E5-2670 (Sandy Bridge, 32nm) processor. The former is a high-end, dodeca-core processor operating at 2.7 GHz priced at about $3300 (as of June 15, 2015 on Amazon, Inc.). The latter is an older-generation octa-core processor operating at 2.6 GHz, priced at about $1700 (as of June 15, 2015 on Amazon, Inc.). These list prices include DDR3 memory and are reported in USD. We used PassMark scores to compare the performance of these two processors [1]. Table I shows the details of these two server processors.

TABLE I
COMPARING 32NM AND 22NM PROCESSORS

Features	**Intel Xeon E5 2697 v2**	**Intel Xeon E5-2690**
Manufacturing Technology	Ivy Bridge 22nm	Sandy Bridge 32nm
Processors Per Node	2	2
Cores Per Processor (Threads)	12 (24)	8 (16)
L2 Cache	3 GB	2 GB
L3 Cache	30 GB	20 GB
Maximum Memory	786,432 MB	393,216 MB
Clock Speed (Turbo)	2.7 GHz (3.5 GHz)	2.6 GHz (3.3 GHz)
PassMark Performance Test Result	17,812	13,985
TDP	130 W	115 W
List Price (USD)	$3300	$1700

Based on the PassMark score, it is expected that a system designed with the Intel Xeon E5 2697 processor will be about 27% faster on most single-node, computational workloads, but will require more per-node peak power. However, most HPC applications do not utilize peak node power [14], [17], which implies that workloads executed on a system designed with the former, high-end processor, will result in significant power being wasted and performance being limited significantly, as the applications cannot scale to more nodes or utilize allocated power. For a given *processor cost budget*, it is possible to buy 94% more nodes of the latter Sandy Bridge processor and design an overprovisioned system. Based on this intuitive analysis, we propose a model in the next section to determine when overprovisioning can be beneficial. The key parameters for the model include node price, node performance, node power, system power, and workload scalability characteristics.

III. COST MODEL

The main goal of our model is to determine whether it is possible to use overprovisioning to improve performance under a fixed power budget without added infrastructure cost.

We design a worst-case provisioned system using high-end nodes and an overprovisioned system using older-generation nodes, both under the same cost and computational power constraints. We then predict performance on these two systems in order to conduct a cost-benefit analysis.

As shown in Section II-B, it is possible to buy more older-generation nodes with similar performance characteristics for a given *processor cost budget*. While compute servers contribute to a significant portion of the total hardware cost, it is important to consider the cost of other components such as the interconnect and the I/O subsystem. For example, a worst-case provisioned system might need fewer total racks than an overprovisioned system. We translate these other costs to the node level in order to correctly estimate their influence. This allows our model to adjust for the total *hardware cost budget*.

It is important to analyze the performance difference between the high-end and the older-generation nodes. In the example in Section II-B, we observed that the high-end node is expected to be 27% faster for the single-node case. However, performance at a different node count greatly depends on the workload. In order to accommodate this, we also create a model to predict performance (execution time) on the two architectures based on representative HPC workloads. The input parameters for our model are summarized in in Table II; the details of the model are presented in Section III-C.

A. Power Budget and Node-Level Values

The main input parameter when designing a worst-case provisioned or an overprovisioned system is the global power budget. In our model, we limit this to the *power associated with computation* (P_{sys}) consumed by compute nodes and other components on the rack. The total site-wide power budget, of course, will include power from other components, such as cooling. It is important to note that while such *base* power is required for successful operation, it does not directly affect workload performance. Therefore, we do not consider it in our model, since we focus on the impact of dynamic power on performance.

In order to ensure that the computational power budget is always met, we need to account for the maximum or peak power on each node for a worst-case provisioned system. Similarly, for an overprovisioned system, we need to account for the minimum power required for each node's operation (idle power). We thus need two input parameters, P_{n_max} and P_{n_min}, which correspond to the maximum node power for the high-end node, and the minimum node power for the older-generation node. The node power is determined

TABLE II
MODEL INPUT PARAMETERS

Parameter	Variable	Description
Power Bound	P_{sys}	Power budget allocated to the *computational* components of the system.
Maximum Node Power	P_{n_max}	Maximum possible node power for the *high-end node* based on its overall TDP.
Minimum Node Power	P_{n_min}	Minimum possible node power for the *older-generation node* based on its idle power.
Effective Cost Ratio	r_c	Ratio of the *effective* per-node cost of the high-end node to that of the older-generation node.
Performance	r_p	Percentage by which the high-end node is faster than the older-generation node
Workload Scalability Model	$\{m, b\}$	The slope and intercept of the linear performance model based on workload scalability characteristics (obtained on the older-generation node)

based on the thermal design point (TDP) of the processor and memory, as well as the associated rack and interconnect power (translated to a per-node value). These values help us determine the maximum number of nodes that we can power up safely.

We define the *effective cost ratio* (r_c) as the ratio of the per-node costs of the high-end and the older-generation node. In our model, r_c takes into account the last rack being underutilized because the number of racks does not evenly divide the number of nodes. However, this can typically be ignored because a real HPC system has several dozens of racks, which minimizes the effect of the last rack.

B. Performance and Scalability Models

The difference in single-node performance of the two node choices is captured as a percentage value, r_p. This represents the percentage amount by which the high-end node is faster than the older-generation node, and can be obtained by using standardized benchmarking tests or vendor specified values.

We also need a *scalability model* for the workloads under consideration to predict performance on multiple nodes at scale. In our case, we obtain this scalability model on the system with older-generation nodes, and project the performance using r_p. We assume that workloads will scale similarly. This is a fair assumption when the nodes under consideration are separated by one or two generations and have similar features (such as clock rates, vector units and cache sizes).

The scalability model should accurately represent the workload characteristics on the older-generation nodes. Developing estimation models and extracting their characteristics is an orthogonal problem to our work, and several application-specific as well as general models are being studied [4], [9], [12]. We assume a linear scalability model and require two inputs, the slope and intercept (m and b). These inputs are determined by gathering data for the representative applications at several node counts at various power caps. We explain this process in Section IV.

C. Model Formulation

We now describe the formulation of our model. The input parameters are listed in Table II. The goal is to compute s_{ovp}, which is the ratio of the performance of the high-end, worst-case provisioned system to that of the overprovisioned system.

A summary of variables used to compute s_{ovp} is provided in Table III. We design two systems, one with worst-case provisioning using high-end nodes and another with overprovisioning using older-generation nodes. In both cases we assume the same cost budget and ensure that the power budget is met. We then predict workload performance on both systems to understand scenarios that result in a net benefit.

A key constraint is to ensure that the power budget, P_{sys}, is honored. When designing the worst-case system, P_{sys} constrains the maximum number of nodes we can run at full power. Similarly, for the overprovisioned system, this constrains the maximum number of nodes that we can run at idle power.

The worst-case provisioned node count is determined by the maximum power on the high-end node (P_{n_max}), and the node limit for the overprovisioned system is determined by the minimum power on the cheaper node (P_{n_min}). Both the node-level power values are calculated by considering the power consumption of the processor, memory and associated interconnect. We derive the maximum number of nodes for both the systems as shown in Equations 1-2.

$$n_{wc} = \frac{P_{sys}}{P_{n\_max}} \tag{1}$$

$$n_{lim} = \frac{P_{sys}}{P_{n\_min}} \tag{2}$$

We derive a fixed *cost constraint* (c_{wc}) for both systems. We use the *effective cost ratio*, r_c, and the number of worst-case provisioned nodes for this (Equation 1). Using r_c allows us to assume cost for the older-generation node to be 1. We can thus analyze the impact of the cost difference between the two types of nodes clearly. This step is shown in Equation 3.

$$c_{wc} = n_{wc} \times r_c \tag{3}$$

TABLE III
COMPUTED (INTERMEDIATE) VALUES IN MODEL

Parameter	Variable	Description
Worst-case Nodes	n_{wc}	Maximum number of nodes for worst-case provisioning based on the power budget
Node Limit	n_{lim}	Maximum possible number of nodes for overprovisioning based on the power budget
Overprovisioned Nodes	n_{ovp}	Actual number of nodes used for overprovisioning based on the computational cost constraint as well as the power budget
Worst-case Cost Constraint	c_{wc}	Computational cost constraint derived from the worst-case provisioning system. The overprovisioned system has the same constraint.
Worst-case Time	t_{wc}	Predicted workload execution time for the high-end, worst-case provisioned system
Overprovisioned Time	t_{ovp}	Predicted workload execution time for the overprovisioned system with same cost constraint

Given the power and the derived cost constraints, we now determine the maximum number of overprovisioned nodes (n_{ovp}) that we can buy. The cost for the older generation node is assumed to be 1, so the number of nodes that we can buy within the cost constraint is $\lfloor c_{wc} \rfloor$. If this exceeds the number of nodes that are supported by the power budget (n_{lim}), we will end up buying a cheaper overprovisioned machine and will have money left over. The ratio of n_{ovp} to n_{wc} can also be referred to as the *degree of overprovisioning*.

$$n_{ovp} = min(n_{lim}, \lfloor c_{wc} \rfloor) \tag{4}$$

Finally, we predict workload performance on both the systems. We use the performance parameter, r_p, and the parameters $\{m, b\}$ (slope, intercept) from the workload-specific scalability model to accomplish this. If the execution time taken on a single older-generation node is 1, and the high-end node is faster by r_p percent, the execution time for the high-end node is $1 - r_p$.

We determine the ratio s_{ovp}, which represents the *speedup due to overprovisioning* and is defined as the ratio of the workload's execution time on the worst-case provisioned system to that of the overprovisioned system. Note that the workload scalability model applies in a specified node range where linearity holds. As a result, the slope and intercept values can be directly used to determine performance. Equations 5-7 depict these steps. For overprovisioning to have a net benefit, s_{ovp} should be greater than one.

$$t_{ovp} = m \times n_{ovp} + c \tag{5}$$

$$t_{wc} = (m \times n_{wc} + c) \times (1 - r_p) \tag{6}$$

$$s_{ovp} = \frac{t_{wc}}{t_{ovp}} \tag{7}$$

IV. EXPERIMENTAL SETUP

In order to efficiently understand our cost model, we restrict the workload scalability models to individual applications instead of a set of applications. We use execution time under a power bound as a metric for our analysis. A linear model can be extended to include a set of jobs by considering scheduling policies that minimize the overall turnaround time under a power bound.

We consider the hybrid (MPI + OpenMP) versions of the SPhot [11] benchmark and the NAS-MZ [6] suite. SPhot is a 2D photon transport code that uses a Monte Carlo approach to understand the behavior of photons in different materials. The NAS Multi-zone parallel benchmarks (NAS-MZ) are derived from Computational Fluid Dynamics (CFD) applications, and include three benchmarks: Block Tri-diagonal solver, or BT-MZ; Scalar Penta-diagonal solver, or SP-MZ; and Lower-Upper Gauss-Seidel solver, or LU-MZ. We use the class C inputs.

We conduct our experiments on the *rzmerl* cluster at LLNL, which is a 162-node Sandy Bridge cluster. Each node is an Intel Xeon E5-2690 with support for RAPL power capping [10], with two sockets and 8 cores per socket. The memory per node is 32GB. The clock speed is 2.6 GHz, and the maximum Turbo Boost frequency is 3.3 GHz. The socket TDP is 115 W, and the minimum recommended power cap is 51 W.

The cluster has a 32-node per job limit, so our study is limited to 32-nodes. We use MVAPICH2 version 1.7 and compile all codes with the Intel compiler version 12.1.5. OpenMP threads are scheduled using the scatter policy. We run all experiments with power capping on the PKG domain with a single capping window and the shortest possible time window (0.000977 seconds).

We choose six system level power bounds: 2500 W, 3500 W, 4500 W, 5500 W, 6500 W and unlimited (which is 7360 W for 32 nodes based on a TDP of 115 W). We use four socket-level PKG power caps with Turbo Boost disabled: 51 W; 65 W; 80 W; and 95 W. We also run at the TDP power of 115 W with Turbo Boost enabled. We run each application from 8 to 32 nodes, and 4 to 16 cores on each node (both in increments of 2), thus amounting to a total of 2730 *configurations*, where each configuration is defined as a set of three values: number

of nodes, number of cores per node, and the socket power cap. We run each configuration at least three times to mitigate noise.

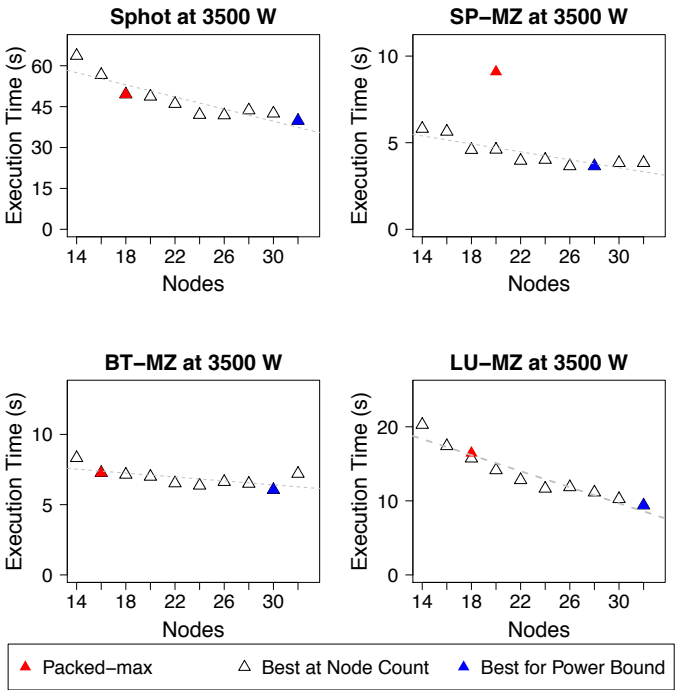

Fig. 1. Benefits of Adding More Nodes

We then develop linear models for each application and each system power bound by choosing the best performing *configuration* at each valid node count. This also enables us to understand the impact of the degree of overprovisioning. This is important because of the law of diminishing returns—adding more nodes beyond a certain limit while keeping the system power bound constant will not result in performance benefits.

As an example, Figure 1 shows the four applications at 3500 W. The y-axis in each subgraph is the raw execution time of the application, and the x-axis represents a node count. For each node count, the best *valid* configuration (one that does not exceed the specified power bound) has been plotted (black empty triangle). The best performing configuration under the power bound has been marked with a blue triangle. The linear model has been shown with the dashed gray line. We use the slope and intercept of this fitted line as the inputs to our model.

In our data, the coefficients of the linear model (slope and intercept) were similar across all six system power bounds under consideration for a given application (less than 2% difference). Thus, we assume that for a given application, the input parameters m and b are identical across different power bounds. Our median prediction error across all applications is under 7%.

The application-specific worst-case configuration, or *packed-max*, has been marked with a red triangle. Here, we use as many nodes as possible under the power bound while using all cores and power on a node with Turbo Boost enabled. The *actual* power consumed by the application is used to determine this worst-case configuration. It is important to note that for a procurement, we have to follow the *absolute* worst-case configuration instead of the application-specific, *packed-max* configuration. The absolute worst-case configuration can be derived by dividing the system-level power bound by the maximum possible power for the node. In the example in Figure 1, the absolute worst-case *node count* is 12 nodes for 3500 W. This has been derived by calculating the maximum possible power on each node, assuming that each socket takes 115 W and each memory unit takes 30 W.

Depending on the application, the benefits of adding more nodes under the same power bound (degree of overprovisioning) vary. For example, the benefits for an application such as BT-MZ are limited. On the other hand, adding more nodes is beneficial for applications such as SPhot and LU-MZ.

V. Evaluation Results

We use the ratio s_{ovp} derived in Section III-C to determine situations where overprovisioning has a net benefit ($s_{ovp} > 1$). We illustrate this with an example by choosing default input parameters based on real data for our model and designing a worst-case provisioned and an overprovisioned system. We then analyze the impact of each individual parameter (while keeping others constant) on the performance of the two systems. These default input parameters are specified in Table IV and have been derived based on the node data from Section II-B. The values for the node power for the high-end and the older-generation node include the CPU, memory and base power. We use application-specific scalability models; an overview of these models was provided in Section IV.

TABLE IV
Example: Default Input Parameters

Variable	Value
Effective cost ratio, r_c	1.7
Performance Parameter, r_p	27%
System Power Budget, P_{sys}	7000 W
Maximum Node Power, P_{n_max}	380 W
Minimum Node Power, P_{n_min}	180 W

TABLE V
Example: Workload Scalability Model

Application	Parameters $\{m, c\}$
SPhot	$\{-1.114, 73.07\}$
SP-MZ	$\{-0.112, 7.00\}$
BT-MZ	$\{-0.069, 8.50\}$
LU-MZ	$\{-0.542, 25.93\}$

Table V specifies the workload scalability model parameters for the four applications, and Table VI shows the intermediate values as well as the predicted s_{ovp} values for our example. Because we use a simple linear workload scalability model, we need to enforce a limit on the maximum number of nodes for the validity of this linear behavior. For our applications in this example, we assume that this limit is 48 nodes.

TABLE VI
EXAMPLE: INTERMEDIATE AND OUTPUT VALUES

Application	n_{wc}	n_{lim}	c_{wc}	n_{ovp}	t_{ovp} (s)	t_{wc} (s)	s_{ovp}
SPhot	18	50	30.6	30	39.64	38.70	0.98
SP-MZ	18	50	30.6	30	3.51	3.61	**1.02**
BT-MZ	18	50	30.6	30	6.41	5.29	0.83
LU-MZ	18	50	30.6	30	9.66	11.80	**1.22**

We present results for our four applications because they exhibit distinct scalability characteristics. As can be observed from Table VI, for workloads that scale well, such as the ones with characteristics similar to LU-MZ (refer to Figure 1), it is possible to achieve better performance with overprovisioning (s_{ovp} of 1.22). Similarly, for workloads with characteristics similar to SP-MZ and SPhot, a break-even point can be determined. On the other hand, for applications that do not scale well, such as BT-MZ, worst-case provisioning leads to better performance. This can be observed from the s_{ovp} value of 0.83 for BT-MZ.

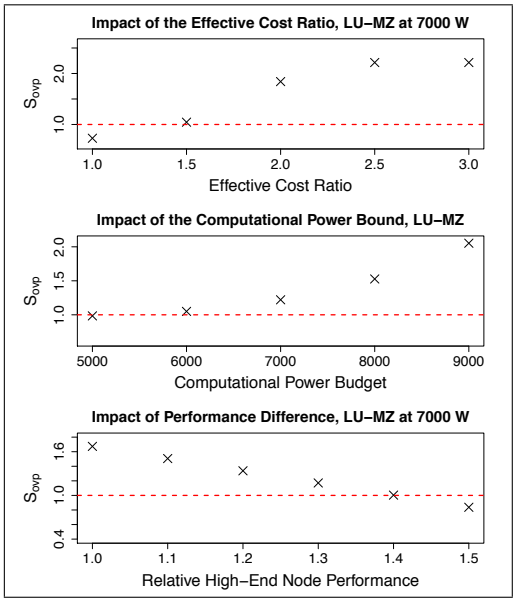

Fig. 2. LU-MZ Analysis

We now present some detailed graphs to better understand the scenarios where overprovisioning leads to a net benefit and to understand the impact of the input parameters on s_{ovp}. For each graph, the y-axis is the derived ratio, s_{ovp}. The x-axis varies based on the input parameter under consideration. For each input parameter that is being varied, all other input parameters are held constant and have values given by Table IV. For readability, the graphs are not centered at the origin. The break even points have been marked by drawing a dashed red line. Anything above this line is a scenario where overprovisioning does better.

We conduct this analysis to explore the scenarios that may occur during the procurement of a real HPC system.

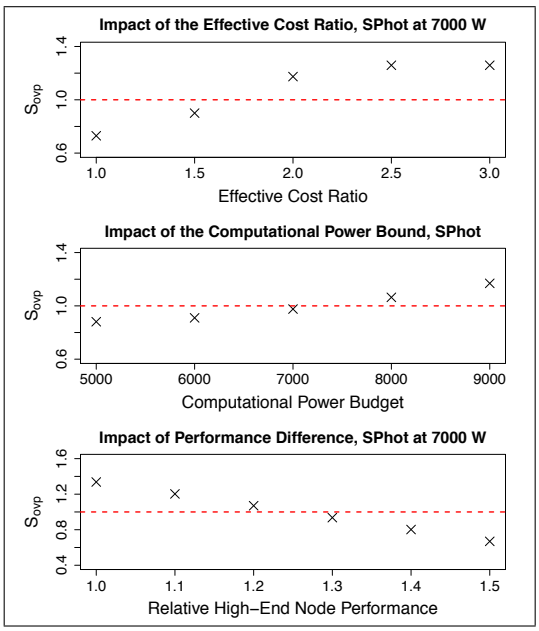

Fig. 3. SPhot Analysis

For example, the effective cost ratio may vary based on the negotiation with the vendor. Similarly, the performance across two nodes may differ based on which micro-architectures are being considered.

Figure 2 shows results for the LU-MZ application. In this figure, there are three sub-graphs. The first one depicts the impact of varying the effective cost ratio, r_c, on s_{ovp}. The effective cost ratio affects the degree of overprovisioning directly. When the effective cost ratio is high, it is possible to buy many more cheaper, older-generation nodes than when the effective cost ratio is low. A cost ratio of 1 indicates that the high-end node and the older-generation node have the same price. This is not a realistic scenario and it is expected that the high-end node will be more expensive than the older-generation node. The higher the effective cost ratio, the easier it is to overprovision by a larger degree. For LU-MZ, overprovisioning leads to a net benefit when the effective cost ratio is about 1.5.

The second sub-graph shows the impact of varying the system power bound, P_{sys} on s_{ovp}. We observe that a higher system power bound results in almost super-linear improvement in s_{ovp} for LU-MZ. This can be attributed to the fact that a higher system power bound has the potential to

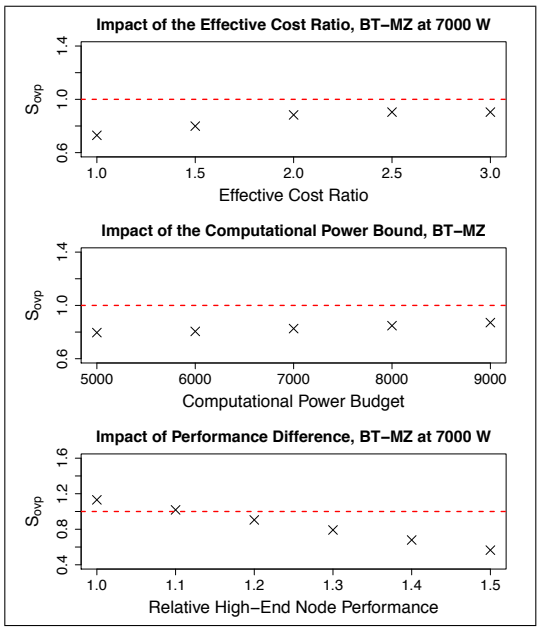

Fig. 4. BT-MZ Analysis

increase the degree of overprovisioning. With a higher degree of overprovisioning, it is possible for an application to scale to more nodes. However, this is dependent on the scalability characteristics of the workload under consideration, which is good for LU-MZ. As a result of this, s_{ovp} is impacted significantly when the system power bound is varied. This will not always be the case.

The third sub-graph in Figure 2 analyzes the impact of the performance parameter, r_p on s_{ovp}. This parameter is determined by benchmarking the high-end node and the older-generation node using a standard single-node test. A high value of r_p indicates that the high-end node is significantly faster. When r_p is 0%, it means that both nodes have the same performance. A value of 0% for r_p is unrealistic though, because the high-end node will always perform better than the older-generation node. For LU-MZ, we observe that even when the high-end node is 40% faster, the overprovisioned system results in a net benefit.

Figure 3 shows the results for SPhot. SP-MZ produces nearly identical results and so is omitted. As discussed previously, the scalability characteristics of the application affect s_{ovp} significantly. With SPhot as well as SP-MZ, the overprovisioned system results in a net benefit when the effective cost ratio of the high-end node to the older-generation node is around two. Also, increasing the effective cost ratio further does not result in proportional improvements. One might expect that a higher cost ratio means that the degree of overprovisioning will be high, and that this will result in performance improvements. However, this is not the case, since the degree of overprovisioning is also constrained by the power budget, which determines n_{lim}. Workloads with scaling characteristics similar to SPhot and SP-MZ are also less sensitive to changes in the system power budget, unlike LU-MZ.

Let us now analyze BT-MZ, which has poor scaling characteristics when compared to the other three applications (refer to Figure 1). As a result, for BT-MZ, the worst-case provisioned system is almost always superior to the overprovisioned system, as can be observed from Figure 4. While increasing the effective cost ratio and the system power budget improves the performance of the overprovisioned system, these improvements are not sufficient to reach a break even point. The only scenario in which the overprovisioned system does better than the worst-case provisioned system is when r_p is in the range of 10-20%. It is thus critical to consider the characteristics of the workload carefully when designing overprovisioned systems.

VI. Conclusions

In this paper, we developed a model for understanding the economic viability of hardware overprovisioned systems in power-constrained high-performance computing. We showed that it is possible to design overprovisioned systems that lead to significant performance benefits without exceeding a given infrastructure cost budget. Future work involves conducting a similar analysis for sets of applications and extending the scalability models to include metrics, such as job turnaround times and throughput.

VII. Acknowledgments

Part of 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-692100).

References

[1] PassMark Software, Pty Limited, 2016. https://www.cpubenchmark.net/.

[2] P. Bailey, D. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B. de Supinski. Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems. In *International Conference on Parallel Processing*, ICPP '14, 2014.

[3] P. Bailey, A. Marathe, D. Lowenthal, B. Rountree, and M. Schulz. Finding the Limits of Power-constrained Application Performance. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, 2015.

[4] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. de Supinski, and M. Schulz. A Regression-based Approach to Scalability Prediction. In *Proceedings of the 22nd Annual International Conference on Supercomputing*, 2008.

[5] A. Borghesi, C. Conficoni, M. Lombardi, and A. Bartolini. Ms3: A mediterranean-stile job scheduler for supercomputers-do less when it's too hot! In *High Performance Computing & Simulation (HPCS), 2015 International Conference on*, pages 88–95. IEEE, 2015.

[6] R. F. V. der Wijngaart and H. Jin. NAS Parallel Benchmarks, Multi-Zone Versions. Technical report, July 2003.

[7] D. Ellsworth, A. Malony, B. Rountree, and M. Schulz. Dynamic Power Sharing for Higher Job Throughput. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, 2015.

[8] D. Ellsworth, A. Malony, B. Rountree, and M. Schulz. POW: System-wide Dynamic Reallocation of Limited Power in HPC. In *High Performance Parallel and Distributed Computing (HPDC)*, June 2015.

[9] H. Gahvari, A. H. Baker, M. Schulz, U. M. Yang, K. E. Jordan, and W. Gropp. Modeling the Performance of an Algebraic Multigrid Cycle on HPC Platforms. In *Proceedings of the International Conference on Supercomputing*, ICS '11, 2011.

[10] Intel. Intel-64 and IA-32 Architectures Software Developer's Manual, System Programming Guide, 2011.

[11] Lawrence Livermore National Laboratory. SPhot–Monte Carlo Transport Code, 2001. https://asc.llnl.gov/computing_resources/purple/archive/benchmarks/sphot/.

[12] R. Long, S. Moore, and B. Rountree. Iso-power-efficiency: An approach to scaling application codes with a power budget. In *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*, pages 905–910, May 2015.

[13] A. Marathe, P. Bailey, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski. A Run-Time System for Power-Constrained HPC Applications. In *In International Supercomputing Conference (ISC)*, July 2015.

[14] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski. Exploring Hardware Overprovisioning in Power-constrained, High Performance Computing. In *International Conference on Supercomputing*, June 2013.

[15] T. Patki, A. Sasidharan, M. Maiterth, D. Lowenthal, B. Rountree, M. Schulz, and B. de Supinski. Practical Resource Management in Power-Constrained, High Performance Computing. In *High Performance Parallel and Distributed Computing (HPDC)*, June 2015.

[16] O. Sarood, A. Langer, A. Gupta, and L. V. Kale. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. In *Supercomputing*, Nov. 2014.

[17] O. Sarood, A. Langer, L. V. Kale, B. Rountree, and B. R. de Supinski. Optimizing Power Allocation to CPU and Memory Subsystems in Overprovisioned HPC Systems. In *International Conference on Cluster Computing*, 2013.

[18] Z. Zhang, M. Lang, S. Pakin, and S. Fu. Trapped Capacity: Scheduling under a Power Cap to Maximize Machine-room Throughput. In *Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing*, pages 41–50. IEEE Press, 2014.