Large-Memory Nodes for Energy Efficient High-Performance Computing

Darko Zivanovic
Barcelona Supercomputing Center (BSC)
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain
darko.zivanovic@bsc.es

Milan Radulovic
BSC & UPC, Barcelona, Spain
milan.radulovic@bsc.es

Germán Llort
BSC & UPC, Barcelona, Spain
german.llort@bsc.es

David Zaragoza
BSC & UPC, Barcelona, Spain
david.zaragoza@bsc.es

Janko Strassburg
BSC, Barcelona, Spain
janko.strassburg@bsc.es

Paul M. Carpenter
BSC, Barcelona, Spain
paul.carpenter@bsc.es

Petar Radojković
BSC, Barcelona, Spain
petar.radojkovic@bsc.es

Eduard Ayguadé
BSC & UPC, Barcelona, Spain
eduard.ayguade@bsc.es

ABSTRACT

Energy consumption is by far the most important contributor to HPC cluster operational costs, and it accounts for a significant share of the total cost of ownership. Advanced energy-saving techniques in HPC components have received significant research and development effort, but a simple measure that can dramatically reduce energy consumption is often overlooked. We show that, in capacity computing, where many small to medium-sized jobs have to be solved at the lowest cost, a practical energy-saving approach is to *scale-in* the application on large-memory nodes. We evaluate scaling-in; i.e. decreasing the number of application processes and compute nodes (servers) to solve a fixed-sized problem, using a set of HPC applications running in a production system. Using standard-memory nodes, we obtain average energy savings of 36%, already a huge figure. We show that the main source of these energy savings is a decrease in the node-hours ($node\_hours = \#nodes \times exe\_time$), which is a consequence of the more efficient use of hardware resources.

Scaling-in is limited by the per-node memory capacity. We therefore consider using large-memory nodes to enable a greater degree of scaling-in. We show that the additional energy savings, of up to 52%, mean that in many cases the investment in upgrading the hardware would be recovered in a typical system lifetime of less than five years.

CCS Concepts

•Computer systems organization → Distributed architectures; •Hardware → Power and energy;

MEMSYS 2016 October 3–6, 2016, Washington, DC, USA

DOI: http://dx.doi.org/10.1145/2989081.2989083

Keywords

High-performance computing, Large-memory nodes, Energy efficiency, Capacity computing, Scaling-in.

1. INTRODUCTION

Energy consumption is by far the most important contributor to HPC cluster operational costs, and it accounts for a large share of the total cost of ownership [20, 23]. For this reason, advanced energy-saving techniques in CPUs, cooling systems, next-generation memories and interconnects have been the subjects of significant industrial and academic research and development effort. Despite this investment, as shown in this paper, researchers and users continue to overlook a simple measure that can dramatically reduce energy consumption, that of simply optimizing the number of compute nodes (servers) used to execute each job.

High-performance computing is broadly divided into capability and capacity computing. Capability computing refers to the use of a large-scale HPC installation to solve a single problem in the shortest possible time; e.g. simulating the human brain on a Tier-0 HPC system. In contrast, *capacity* computing refers to optimizing system efficiency to solve many mid-size or smaller problems at the lowest possible cost [3]. Typical examples of capacity computing would be small and medium enterprises using on-demand HPC resources to explore future product designs. In the context of capacity computing, the user is concerned not with the running time of a single job, but with the total running time of a batch of jobs and total system throughput.

This paper investigates the potential for saving energy through *scaling-in* on large-memory nodes. Scaling-in refers to executing a fixed problem on a fixed machine, but using a reduced number of application processes and compute nodes. Scale-in increases single job execution time, but, as we quantify in this paper, it substantially decreases energy consumption and reduces the running time of a batch of jobs. It is therefore of particular interest in the context of capacity computing. We study the trade-off between job/batch execution time, energy consumption and node-hours ($node\_hours = \#nodes \times exe\_time$) using a set of

large-scale HPC applications running on a production HPC system. In summary, we find that scaling-in on standard memory nodes improves energy consumption by 36% on average, a huge figure. We investigate the sources of this energy savings, and show that its main source is a reduction in node-hours.

Scaling-in is limited by the per-node memory capacity, since, for a fixed size problem, reducing the number of nodes increases the memory required at each node. We therefore investigate the benefits of upgrading the per-node memory capacity in terms of energy savings and reducing the node-hours, and follow this with a financial cost-benefit analysis. We show that the additional energy savings, of up to 52%, mean that an investment in upgrading the memory would be typically recovered in less than five years.

2. METHODOLOGY

2.1 Hardware platform

We execute experiments on the MareNostrum III supercomputer [7], which is one of the six Tier-0 (largest) HPC systems in the Partnership for Advanced Computing in Europe (PRACE)[2]. MareNostrum III contains 3,056 compute nodes, each with two eight-core Sandy Bridge-EP E5-2670 at 2.6 GHz, connected via InfiniBand FDR 10. As in most HPC systems, hyperthreading is disabled. Standard MareNostrum compute nodes have 32 GB of DDR3-1600 main memory, i.e. 2 GB per core. Large-memory nodes are identical to standard nodes except that their memory capacity has been upgraded to 128 GB.

2.2 Applications

We study HPC scaling behaviour using the Unified European Application Benchmark Suite (UEABS) [19], the set of production applications and datasets designed for benchmarking the European PRACE HPC systems for procurement and comparison purposes [2]. All applications are parallelized using MPI, and we executed them with the Test Case A dataset, which is scalable up to 1,024 processes. We ran the benchmarks with one MPI process per core, i.e. sixteen processes per node. Table 1 shows the six benchmarks that we analysed. The remaining benchmarks either could not be installed or they came with a Test Case A dataset compatible with an insufficiently wide range of nodes for our analysis. The table also shows the range of nodes on which we ran the benchmarks. In all cases, the maximum was 64 nodes (1,024 cores). The minimum was either a single node or the least number of nodes necessary to meet the memory requirements. We also list per-node memory requirements for the benchmarks running on the minimum number of nodes.

2.3 Power and energy measurements

The node power consumption was measured using IBM Active Energy Manager power modules, which monitor the voltage and current at the node power supply [11]. Active Energy Manager is part of the Integrated Management Module II in the firmware of the System x iDataPlex dx360 M4 [5]. The modules measure the node's total power consumption, including power supply, motherboard with all its components, CPUs, and memory. The MareNostrum node firmware samples the power consumption every second, and it computes the energy consumption by multiplying the measured power sample by the interval length of one second. Finally, the LSF batch job manager [12] sums the energy measurements during the whole execution of a job, and it reports the total in the job execution log file.

We estimate the energy consumption of the interconnect. Measurements from the node power modules already include the network interfaces in the node, so we focus on the energy consumption of the switches. Current network components are observed to have close-to-constant power demand, independent of load, with a deviation of less than 5% [17, 21]. This means that the total switch power consumption can be determined by adding up the vendor's figures for typical use, and, since power consumption is independent of activity, the total can be attributed to the nodes equally. We calculate a constant 7.0 W/node for the top-of-rack switches, 15.3 W/node for the core switches, and 4.8 W/node for the Ethernet storage and management networks, giving a total of 27.1 W per node. For a given job, the interconnect energy consumption can therefore be calculated assuming a constant power of 27.1 W per node.

Table 1: UEABS applications used in the study.

Application	Science area	Memory [GB] [a]	Number of nodes
ALYA	Computational mechanics	15.1	1–64
NAMD	Computational chemistry	25.9	1–64
QE[b]	Computational chemistry	17.7	1–64
BQCD	Particle physics	14.4	4–64
GENE	Plasma physics	16.2	4–64
CP2K	Computational chemistry	17.0	8–64

[a] Per-node memory usage when application runs on the minimum number of nodes.
[b] QE stands for Quantum Espresso application.

3. SCALING-IN ON STANDARD NODES

3.1 Execution time *vs.* node-hours *vs.* energy

Before running any experiment on an HPC machine, the user must choose to run the application on a particular number of nodes. This scenario, of a fixed problem to solve on a variable number of nodes, is known as strong scaling. The largest number of nodes is limited by the machine size and application's scalability: beyond a certain point, adding further nodes delivers diminishing returns. The *smallest* number of nodes is constrained by the amount of memory needed by the application: scaling-in the application too far would require more memory per node than is available.

Until now, the number of nodes has been chosen as a trade-off between execution time and node-hours, where the latter is the main "cost" exposed to the user, and is given by the number of nodes multiplied by the job's execution time. Figure 1 shows this trade-off for the ALYA application. As the number of nodes, on the x-axis, is increased from 1 to 64, the execution time drops by a factor of 27 (1/0.04),[1] while the node-hours increase by a factor of 2.37. At the same

[1]We report only the execution time of the HPC job. Time waiting in the job queue and/or moving the results to an interactive server for post-processing can significantly reduce the effective speed-up.

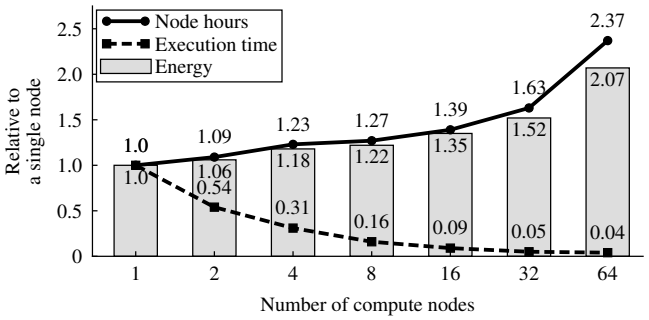


Figure 1: ALYA, 1–64 nodes: Increasing the number of nodes increases both energy and node-hours, with strong correlation.

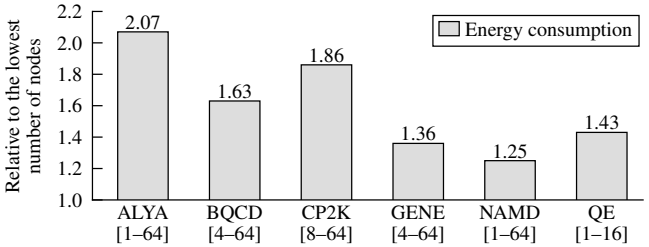


Figure 2: UEABS applications: Increasing the number of nodes causes significant energy overheads.

time, the energy consumption increases by a factor of 2.07. The energy consumption is about 90% compute nodes and 10% switches, and this ratio was roughly the same in all our experiments.

Figure 2 summarizes the energy results for all the applications under study.[2] Increasing the number of nodes above the minimum always leads to significant energy overheads, between 1.25× and 2.07×, with an average of 1.6×.

To understand these results in the context of capacity computing, we analyze how execution time, node-hours and energy are affected by scale-in and scale-out, for a single job and for many jobs. The upper half of Table 2 refers to the single-job experiments, discussed in the previous paragraph. In this case, scale-in greatly increases the execution time, since it reduces the use of concurrent hardware resources (to 1 node instead of 64). When we move to 64 jobs, however, as illustrated in the bottom half of the table the analysis changes. The number of 64 jobs is selected to simplify the illustration of the phenomena; the conclusions are applicable for *any* large number of jobs. Scale-in executes the set of jobs across all 64 nodes, with an independent job on each node. Although the total experiment size increases by a factor of 64, the execution time remains the same. In the scale-out approach, however, since each job already executes across all 64 nodes, the jobs execute one after another, and the execution time increases by a factor of 64. The scale-in approach is 27× slower for a single job, but 2.37× *faster* for 64 jobs. In both cases, scale-in approach reduces energy consumption by 2.07×.

[2]Although QE processing Test Case A should scale up to 64 nodes [19], we observed good scalability up to 16 nodes but slowdowns for 32 and 64 nodes. We therefore report results for the range 1–16 nodes for QE.

Table 2: ALYA, 1 vs. 64 jobs: The scale-in approach is 27× slower for a single job, but 2.37× *faster* for 64 jobs. In both experiments scaling-in reduces node-hours by 2.37× and energy consumption by 2.07×.

	Nodes per job	Nodes	Exe time [min]	Node-hours	Energy [kWh]
1 job					
Scale-out	64	64	**1.14**	1.21	0.30
Scale-in	1	1	30.62	**0.51**	**0.14**
Ratio			0.04	2.37	2.07
Better approach			*Scale-out*	*Scale-in*	*Scale-in*
64 jobs					
Scale-out	64	64	72.71	77.56	19.10
Scale-in	1	64	**30.62**	**32.66**	**9.25**
Ratio			2.37	2.37	2.07
Better approach			*Scale-in*	*Scale-in*	*Scale-in*

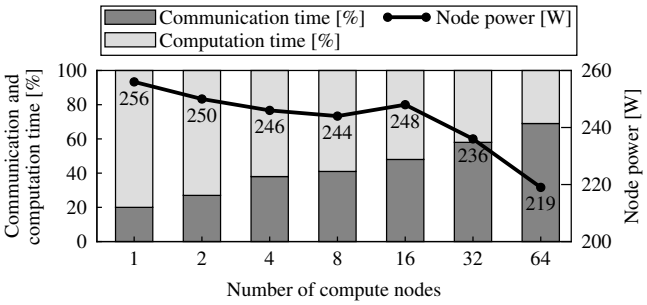


Figure 3: ALYA, 1–64 nodes: Scaling-out decreases power per node, since nodes spend more time in communication.

Therefore, in capacity computing, where there are many smaller jobs, scaling-in improves all three metrics: execution time, node-hours, and energy consumption. Although important, this fact is overlooked by most of HPC research. To the best of our knowledge, this is the first study that *quantifies* improvements of scaling-in of large-scale HPC applications. We believe that the presented results will motivate further research in this direction and impact the policies for operational use of large HPC clusters.

3.2 Understanding energy *vs.* node-hours

For all the applications, as for ALYA shown in Figure 1, there is a clear correlation between the increments in node-hours and energy; however the two curves do not grow at the same pace. In fact, the node-hours curve always exceeds the energy curve. Since interconnect switch energy is proportional to node-hours (constant 27.1 W/node), the gap between the curves must come from a reduction in per-node power consumption. Figure 3 explores the node power consumption for the ALYA application, with each data point being the average of ten experiments. As the number of nodes, on the x-axis, is increased from 1 to 64, the per-node power reduces from 256 W to 219 W, a drop of 15%. The trend is not followed precisely, but the results are repeatable,

as sample standard deviation was negligible.

The per-node power reduction comes due to changes in the behavior of HPC applications when scaling-out, mainly because of the increase in the communication-to-computation ratio. We trace the applications and measure the time spent in communication and computation with the Limpio instrumentation tool [18]. As shown in Figure 3, in the case of ALYA, increasing the number of nodes from one (16 processes) to 64 (1,024 processes) increases the proportion of time spent in communication from 20% to 69%. Since the power consumption of the communication (MPI functions) ranges between 200 W and 220 W, compared with about 280 W for computation, increasing the time spent in communication would pull down the average power consumption. In summary, for all applications under study, the higher the number of nodes, the higher the proportion of time that is spent in communication, and the lower the average per-node power consumption.

3.3 Implications and impact

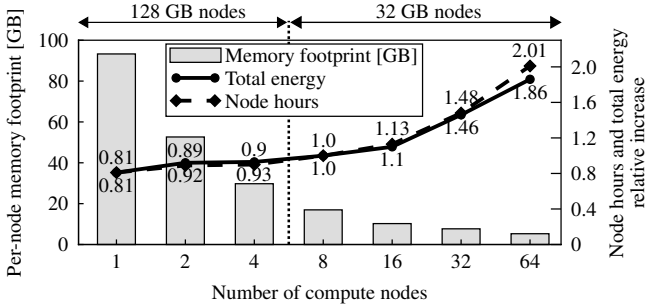
The number of compute nodes to use in a given experiment impacts the application's execution time, node-hours and energy consumption, and, in aggregate, the throughput of the whole HPC system. This topic has not yet been thoroughly explored in the context of capacity computing. This is perhaps because HPC was traditionally biased to large public research centers and academia, which are heavily focused on capability computing.

In recent years, however, HPC has entered industry, including small and medium enterprises, and many users now pay for their time on rented HPC resources. With this change, the cost of HPC experiments has become highly visible, and therefore of prime importance, and scaling-out of HPC applications is now a serious trade-off between execution time and cost. In addition to this, energy efficiency has become an important consideration in state-of-the-art HPC systems, and it is one of the main limitations in the design of future ones [20].

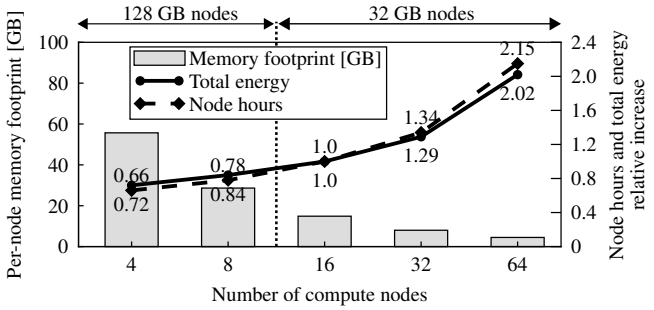
Considering these recent changes in HPC, and future requirements and limitations, it is important to rethink the scaling-out of HPC applications. The results in the previous section show that application scaling-out increases energy consumption on average by a factor of 1.6. Equivalently, from the point-of-view of current practice, scaling-in reduces the energy consumption by 36%, on average. To the best of our knowledge, this paper is the first to emphasize how the number of nodes impacts energy consumption and to quantify the potential energy savings.

4. LARGE-MEMORY NODES FOR ENERGY EFFICIENCY

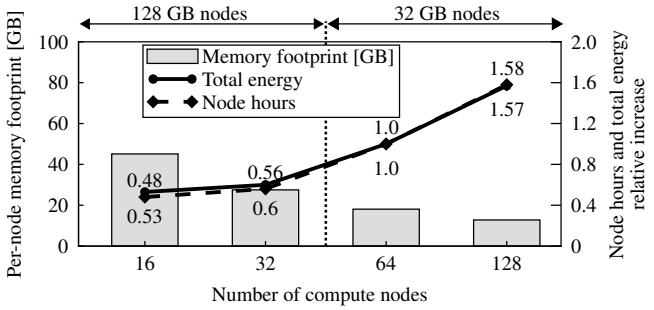
As described in previous section, reducing the number of nodes improves energy efficiency but it increases the memory demand per node, with the result that scaling-in is limited by the nodes' memory capacity. In Table 1 it was shown that CP2K, for example, requires at least eight MareNostrum nodes to fit the problem size. This is explained further in Figure 4. CP2K results are presented in Figure 4(a), and they show how reducing the number of nodes, shown on the x-axis, to four or fewer causes the per-node memory footprint to exceed the standard node memory capacity of 32 GB. Figures 4(b) and 4(c) show the same trend for ALYA and QE applications processing Test Case B, the larger input dataset intended for Tier-0 HPC systems. ALYA application exceeds the 32 GB per node memory footprint on eight or fewer nodes, while QE requires at least 64 standard nodes to fit into the available main memory.



(a) CP2K (Test Case A)



(b) ALYA (Test Case B)



(c) QE (Test Case B)

Figure 4: Scaling-in increases memory requirements and energy efficiency of HPC applications. Node-hours and energy are shown relative to the experiment on the minimum number of standard nodes.

In addition to the memory footprint, Figure 4 also plots the node-hour and energy consumption curves. For CP2K in Figure 4(a), the experiments with eight or more nodes use standard nodes, whereas the experiments with four or fewer nodes by necessity use large-memory nodes. Results are normalized to the eight-node experiment, which is the best result on standard nodes. It is clearly seen that, for the CP2K application, scale-in to large-memory nodes further improves the energy efficiency. Moving from eight standard nodes to a single large-memory node leads to 19% savings in both node-hours and energy consumption. In Figures 4(b) and 4(c) node-hours and energy curves follow the same trend as for CP2K, and the savings are even higher. For ALYA,

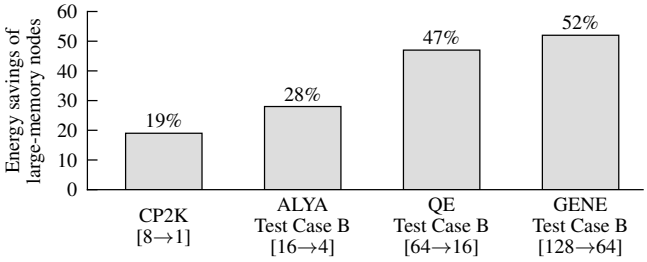


Figure 5: Summary of energy savings enabled by using large-memory nodes. $[a \rightarrow b]$ refers to a shift from a standard to b large-memory nodes.

scale-in from 16 standard to 4 large-memory nodes led to 28% energy and 34% node-hours savings, while QE saved 47% of energy and 52% of node-hours when moving from 64 to 16 nodes. We also analyze GENE running larger Test Case B. For GENE, shift from 128 standard to 64 large-memory nodes saved 52% of energy and 55% of node-hours.[3]

Figure 5 summarizes energy savings enabled by running the experiments on large-memory nodes. We detect energy savings from 19% for CP2K, up to 52% for GENE running Test Case B, with an average of 36%, a huge figure.

4.1 Large-memory nodes for capacity computing

To understand benefits of using large-memory nodes in the context of capacity computing, we analyze how execution time, node-hours and energy are affected on standard and large-memory nodes, for a single job and for many jobs. The upper half of Table 3 refers to the single-job experiments. In this case, using large-memory nodes increases the execution time by 6.5×, since it reduces the use of concurrent hardware resources (to one node instead of eight). When we move to eight jobs, however, as illustrated in the bottom half of the table, the analysis changes. Scale-in approach on large-memory nodes executes the set of jobs across all eight nodes, with an independent job on each node. Although the total experiment size increases by a factor of eight, the execution time remains the same. In the scale-in approach on standard nodes, however, since each job already executes across all eight nodes, the jobs execute one after another, and the execution time increases by a factor of eight. The scale-in on large-memory nodes approach is 6.5× slower for a single job, but 1.23× *faster* for eight jobs. In both cases, the scale-in on large-memory nodes reduces energy consumption by 1.24×. Therefore, in capacity computing, where there are many smaller jobs, using large-memory nodes improves all three metrics: execution time, node-hours, and energy consumption.

4.2 Large-memory node cost-benefit analysis

Finally, this section explores whether large-memory nodes are worthwhile from a financial point-of-view; i.e. whether the decrease in electricity costs would be sufficient to recover the cost of the extra memory. This analysis concentrates only on the financial return. Large memory nodes also increase system throughput, providing an extra benefit beyond that evaluated in this section.

[3]GENE is excluded from Figure 4 as it has only two data points, for 128 standard and 64 large-memory nodes.

Table 3: CP2K, 1 vs. 8 jobs: Execution on large-memory nodes is 6.5× slower for one job, but 1.23× *faster* for eight jobs. For both job sizes, node-hours and energy reduce when using large-memory nodes.

	Nodes per job	Nodes	Exe time [min]	Node-hours	Energy [kWh]
1 job					
Standard	8	8	**25.9**	3.45	1.07
Large-mem	1	1	168	**2.8**	**0.86**
Ratio			0.15	1.23	1.24
Better approach			*Standard*	*Large-mem*	*Large-mem*
8 jobs					
Standard	8	8	207.2	27.6	8.56
Large-mem	1	8	**168**	**22.4**	**6.88**
Ratio			1.23	1.23	1.24
Better approach			*Large-mem*	*Large-mem*	*Large-mem*

Table 4: Payback from large-memory nodes over five-year system lifetime [%].

Country	$/kWh	Reduction in energy consumption					
		10%	20% (CP2K)	30% (ALYA)	40%	50% (QE/GENE)	60%
U.S.	0.07	16	32	48	64	80	97
U.K.	0.15	33	66	99	**132**	**165**	**198**
Germany	0.17	38	75	**112**	**150**	**188**	**225**
France	0.10	22	44	66	88	**110**	**132**

In Table 4, each entry indicates the percentage payback, over a five-year system lifetime, from the reduced electrical costs delivered by large-memory nodes. Entries that exceed the break-even point of 100% are indicated in bold. The electricity cost for the U.S. is the average industrial price from August 2015 [25], whereas for the U.K, Germany and France industrial prices are from 2014 [1]. The memory upgrade from 32 GB to 128 GB was estimated to cost around $600 per node [4]. The benefit clearly depends on the mix of applications ran on the large-memory nodes, as different applications obtain greater or lesser energy savings. For CP2K, the 20% reduction in energy is not sufficient to recover the costs. For ALYA, a 30% energy saving means that the costs would be recovered in Germany. For QE and GENE, which both obtained roughly 50% reduction, the investment would be recovered in France, Germany and the U.K.

5. RELATED WORK

Significant industrial and academic research has been invested into energy-saving mechanisms for HPC components, such as CPUs, interconnects and memories. Several studies investigate how to employ CPU low-power modes in HPC. Current practice is to run the CPUs at the maximum voltage and frequency even while busy-waiting for an MPI message. Freeh et al. [10] investigate the tradeoff between energy and performance in MPI programs using DVFS. Using the NAS

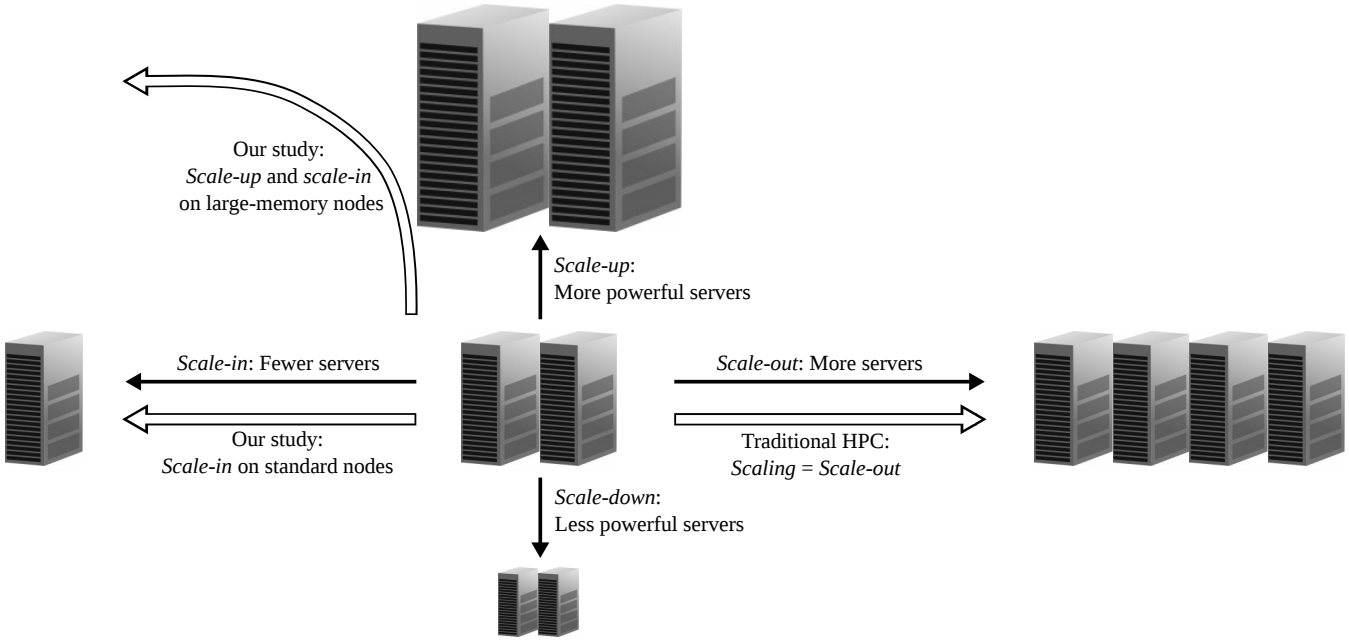


Figure 6: System scaling can be horizontal (*scale-in* or *-out*) and vertical (*scale-up* or *-down*). Traditionally, HPC community is focused mainly on *scale-out*, referring to it simply as *scaling*. Our study analyzes *scale-in* on standard nodes, and a combined *scale-up* and *scale-in* approach on large-memory nodes.

Benchmark Suite, they show that on one node it is possible to use 10% less energy while increasing time by only 1%. Lim et al. [14] propose an MPI runtime system that dynamically reduces the CPU performance during communication phases in order to minimize the energy-delay product (EDP). They show an average reduction in EDP of 10% across the NAS benchmarks suite.

Laros et al. [13] study how to combine CPU frequency scaling (for computation) and network bandwidth scaling (for communication) to reduce the energy consumption. On a set of Department of Energy (DOE) production applications running at large scale, they measure energy savings of up to 39%, with little or no impact on runtime performance. Their results also indicate that each application has a sweet spot based on its computation and communication requirements.

Regarding HPC interconnect energy consumption, Dickov et al. [8] reduce InfiniBand link energy by 21% by powering down the network links during the computation phases and using prediction to ensure that they are powered up in time for the next communication phase. Karthikeyan et al. [22] use prediction and an adaptive stall timer to reduce Ethernet link energy by 68%, while respecting a 1% bound on the increase in execution time.

Several previous studies deal with the energy efficiency of DRAM memory, through different memory management policies, intelligent data placement, and by creating opportunities to transition between power states [9, 16, 24]. Malladi et al. [15] use mobile DRAM devices in order to trade bandwidth for energy efficiency. These studies are validated for datacenter workloads, and it would be interesting to see to what extent their results are applicable to HPC.

In this paper, we show that upgrading the memory capacity in HPC systems for capacity computing is a simple approach to save energy and reduce node-hours. In contrast to most of the prior research, our approach can be applied immediately, and it requires no changes to the system architecture, Operating System, system software or applications.

6. SECOND THOUGHTS ON SCALABILITY

The big data community distinguishes between two dimensions of system scaling — *horizontal*, which refers to the number of compute units, and *vertical*, referring to the hardware capabilities of each compute unit (see Figure 6). There are main two corresponding approaches for the analysis of huge data volumes: *scale-out* and *scale-up*. Scale-out means using more servers in parallel to spread out the workload, while scaling-up means using larger and faster servers to each handle a greater workload. The big data community is very active in analyzing the trade-offs between these two approaches, and whether both of them should co-exist within the same cluster [6].

In HPC, the dominant approach for addressing ever increasing HPC problems is scale-out. Actually, the community uses a general term *scalability* or *scaling* to refer to scale-out; while the more precise terms *scale-up/out*, *horizontal* and *vertical* scaling are rarely used or not used at all.

In modern HPC, the cost and energy consumption of the experiments has become highly visible and of prime importance. Our study demonstrates that simple but unconventional approaches of scale-in (standard node) or scale-up and scale-in (large memory nodes) can lead to significant savings in cost and energy, and improvements in throughput. Therefore, we hope that the study will motivate the community to consider the trade-offs between horizontal and vertical scaling when provisioning and using HPC clusters. Maybe we could start this journey with some second thoughts about the way that we use the word *scalability*.

7. CONCLUSIONS

The importance of energy consumption of current and future HPC machines means that significant research effort has been spent on advanced energy-saving techniques in HPC components. Despite this investment, the simple measure of scaling-in applications to reduce energy consumption has received little attention.

Scaling-in is most appropriate in the context of capacity computing, where a large number of mid-size or smaller problems have to be solved at the lowest cost, and the users are less interested in the execution time of a single job. We therefore advocate upgrading the memory capacity that allows further scaling-in in capacity computing. We validate this approach on a set of large-scale HPC applications running on a production system, and obtain average energy savings of 36%, a huge figure. Finally, we investigate the economical benefits of this approach, and show that the investment in upgrading the hardware would be typically recovered in less than five years.

Overall, we believe that this study will motivate further analysis of the trade-offs between horizontal and vertical scaling in HPC, especially in application domains that are on the border between HPC and big data analytics.

8. ACKNOWLEDGMENTS

This work was supported by the Collaboration Agreement between Samsung Electronics Co., Ltd. and BSC, Spanish Government through Severo Ochoa programme (SEV-2015-0493), by the Spanish Ministry of Science and Technology through TIN2015-65316-P project and by the Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272). This work has also received funding from the European Union's Horizon 2020 research and innovation programme under ExaNoDe project (grant agreement Nº 671578). Darko Zivanovic holds the Severo Ochoa grant (SVP-2014-068501) of the Ministry of Economy and Competitiveness of Spain.

9. REFERENCES

[1] European Commission Eurostat. http://ec.europa.eu/eurostat/.

[2] PRACE Research Infrastructure. http://www.prace-ri.eu.

[3] ETP4HPC Strategic Research Agenda Achieving HPC Leadership in Europe, June 2013.

[4] http://www.pinnaclemicro.com/, Nov. 2015.

[5] System x iDataPlex dx360 M4 Product Guide. https://lenovopress.com/tips0878, Jan. 2015.

[6] R. Appuswamy, C. Gkantsidis, D. Narayanan, O. Hodson, and A. Rowstron. Scale-up vs scale-out for hadoop: Time to rethink? In *Proc. of the Symp. on Cloud Computing (SOCC)*, pages 20:1–20:13, Oct. 2013.

[7] BSC. MareNostrum III System Architecture. http://www.bsc.es/marenostrum-support-services/mn3, 2013.

[8] B. Dickov, M. Pericàs, P. Carpenter, N. Navarro, and E. Ayguadé. Software-Managed Power Reduction in Infiniband Links. In *Proc. of the Int. Conference on Parallel Processing (ICPP)*, pages 311–320, Sept. 2014.

[9] B. Diniz, D. Guedes, W. Meira, Jr., and R. Bianchini. Limiting the Power Consumption of Main Memory. In *Proc. of the Int. Symp. on Computer Architecture (ISCA)*, pages 290–301, June 2007.

[10] V. W. Freeh, F. Pan, N. Kappiah, D. K. Lowenthal, and R. Springer. Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster. In *Proc. of the Int. Parallel and Distributed Processing Symp. (IPDPS)*, page 4.a, Apr. 2005.

[11] IBM. IBM Integrated Management Module II Firmware 3.70. https://www.kernel.org/doc/Documentation/hwmon/ibmaem.

[12] IBM. *Administering Platform LSF*, 2014.

[13] J. H. Laros, K. T. Pedretti, S. M. Kelly, W. Shu, and C. T. Vaughan. Energy Based Performance Tuning for Large Scale High Performance Computing Systems. In *Proc. of the Symp. on High Performance Computing (HPC)*, pages 6:1–6:10, Mar. 2012.

[14] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal. Adaptive, Transparent Frequency and Voltage Scaling of Communication Phases in MPI Programs. In *Proc. of the Conference on Supercomputing (SC)*, Nov. 2006.

[15] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz. Towards Energy-proportional Datacenter Memory with Mobile DRAM. In *Proc. of the Int. Symp. on Computer Architecture (ISCA)*, pages 37–48, June 2012.

[16] K. T. Malladi, I. Shaeffer, L. Gopalakrishnan, D. Lo, B. C. Lee, and M. Horowitz. Rethinking DRAM Power Modes for Energy Proportionality. In *Proc. of the Int. Symp. on Microarchitecture (MICRO)*, pages 131–142, Dec. 2012.

[17] P. Reviriego et al. An initial evaluation of energy efficient ethernet. *IEEE Communications Letters*, 15(5):578–580, May 2011.

[18] M. Pavlovic, M. Radulovic, A. Ramirez, and P. Radojkovic. Limpio - LIghtweight MPI instrumentatiOn. In *Proc. of the Int. Conference on Program Comprehension (ICPC)*, pages 303–306, https://www.bsc.es/computer-sciences/computer-architecture/memory-systems/limpio, May 2015.

[19] PRACE. Unified European Applications Benchmark Suite. www.prace-ri.eu/ueabs/, 2013.

[20] S. Ashby et al. The Opportunities and Challenges of Exascale Computing. Technical report, 2010.

[21] S. L. Song et al. Unified performance and power modeling of scientific workloads. In *Proc. of the Int. Workshop on Energy Efficient Supercomputing (E2SC)*, pages 4:1–4:8, Nov. 2013.

[22] K. P. Saravanan, P. M. Carpenter, and A. Ramirez. A Performance Perspective on Energy Efficient HPC Links. In *Proc. of the Int. Conference on Supercomputing (ICS)*, pages 313–322, June 2014.

[23] R. Stevens and A. White. Architectures and Technology for Extreme Scale Computing. Technical report, DOE, Dec. 2009.

[24] M. E. Tolentino, J. Turner, and K. W. Cameron. Memory MISER: Improving Main Memory Energy Efficiency in Servers. *IEEE Transactions on Computers*, 58(03):336–350, Sept. 2008.

[25] U.S. Energy Information Administration. Electric Power Monthly with Data for August 2015. Technical report, DOE, Oct. 2015.