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

Energy Efficient Supercomputing, SC’16

November 14, 2016

Tapasya Patki, David Lowenthal, Barry Rountree, Martin Schulz, Bronis R. de Supinski
The *holy grail* of large-scale system design: achieve scientific progress with high throughput, high utilization, and low cost.

**Diagram:**
- **TF:** 1997
- **PF:** 2008
- **~100PF:** 2016
- **Exa = 1000 PF**
- **Moore’s Law**
- **Exceptional Science**
- **Modern technology: HW-SW co-design**
- **20-30 MW Hard Bound**
- **850 KW**
- **2.4 MW**
- **15 MW**

**Chart:**
- 1997: 850 KW
- 2008: 2.4 MW
- 2016: 15 MW
- Exa = 1000 PF
Power constraints make it very challenging to balance throughput, utilization, and cost.
Design choices: conservative or liberal?
Worst-case power provisioning and hardware overprovisioning

\[ M > N \text{ and } P_{wc} = P_{ovp} \]

More hardware under the same power budget
(managed with power capping)
The case for hardware overprovisioning: a simple example

- Intel Sandy Bridge cluster of 32 nodes
  - 2 sockets, 8 cores per socket, 2 DRAM modules
- NAS SP-MZ, CFD solver kernel, malleable
- 350 configurations
  - Nodes: 14 to 32, cores per node (scatter): 4 to 16
  - Processor power caps (W): 51, 65, 80, 95, 115
- Peak system power
  - $32 \times 2 \times (115_{\text{cpu}} + 25_{\text{dram}})$, or $\sim 9000 \text{ W}$

Assumed Budget: 4500 W
The case for hardware overprovisioning: we gain performance with intelligent power distribution, memory tuning and scaling

Considerations:

- Application’s time to solution
- Energy = Power * Time
- Underutilizing power is bad for performance as well as energy

<table>
<thead>
<tr>
<th>Config: (n x c, p)</th>
<th>Time (s)</th>
<th>Power* (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WC: (24 x 16, 115)</td>
<td>7.16</td>
<td>3806</td>
</tr>
<tr>
<td>OVP: (30 x 14, 80)</td>
<td>2.94</td>
<td>4459</td>
</tr>
</tbody>
</table>

*Actual Consumption of power across n nodes

Bound: 4500W

Figure 3.6: Example of SP-MZ at 4500 W
Overprovisioning improves throughput and utilization, but introduces operational safety and infrastructure cost concerns

• Dynamic power management techniques require application models, which may be error prone

• We can cap node and memory power, but we cannot guarantee network, I/O and other power through software

• How many *extra* nodes should we add before we lose the benefit and flip this into a problem of underutilized, idle nodes?

• More hardware implies added costs \(\rightarrow\) focus of this paper
Given a fixed power budget and cost budget, can we build an overprovisioned system that results in a net performance benefit?

- **Key intuition:** server processors that are a generation older offer features similar to current generation at a much lower price

<table>
<thead>
<tr>
<th>Feature</th>
<th>Intel Ivy Bridge, 22nm</th>
<th>Intel Sandy Bridge, 32nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Price (USD)</td>
<td>$3300</td>
<td>$1700</td>
</tr>
<tr>
<td>PassMark Performance*</td>
<td>17,812 (27% faster*)</td>
<td>13,895</td>
</tr>
<tr>
<td>Processors (Cores)</td>
<td>2 (24)</td>
<td>2 (16)</td>
</tr>
<tr>
<td>Clock Speed (Turbo)</td>
<td>2.7 (3.5) GHz</td>
<td>2.6 (3.3) GHz</td>
</tr>
<tr>
<td>TDP</td>
<td>130 W</td>
<td>115 W</td>
</tr>
</tbody>
</table>

*On a single node, all cores considered
Let us build a high-end HPC system and a older-generation over provisioned HPC system with fixed cost and power budgets

<table>
<thead>
<tr>
<th><strong>Input Parameters</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Bound*, $P_{sys}$</td>
<td>Power budget allocated to the <em>computational</em> components</td>
</tr>
<tr>
<td>Maximum Node Power, $P_{n_max}$</td>
<td>Maximum possible node power for the <em>high-end node</em> based on its overall <em>TDP</em></td>
</tr>
<tr>
<td>Minimum Node Power, $P_{n_min}$</td>
<td>Minimum possible node power for the <em>older-generation node</em> based on its <em>idle</em> power</td>
</tr>
<tr>
<td>Cost Ratio*, $r_c$</td>
<td>Ratio of the <em>effective</em> per-node cost of the high-end node to that of the older-generation node (&gt;1.0)</td>
</tr>
<tr>
<td>Performance, $r_p$</td>
<td>Percentage the high-end node is faster by on a single-node (&gt;0%)</td>
</tr>
</tbody>
</table>

*These can incorporate rack and interconnect information.*
A workload scalability model to predict multi-node performance at scale is also needed

- Predict performance of workload on the high-end system at a different node count based on multi-node data from older-generation system

- HPC systems are typically designed with a purpose and target workload
  - RFPs come with specific benchmarks and hardware options

- Orthogonal problem
  - Assume a linear model valid over a limited node range for simplicity
Let us now design our two HPC systems based on the power constraint $P_{sys}$, and the derived cost constraint, $c_{wc}$.

- Determine maximum WC nodes based on power budget, derive cost budget

\[
\begin{align*}
  n_{wc} &= \frac{P_{sys}}{P_{n_{max}}} \\
  c_{wc} &= n_{wc} \times r_c
\end{align*}
\]

Represents OVP nodes $\rightarrow$

- Determine maximum possible OVP nodes. Note that cost of older-generation node is 1 based on how we defined $r_c$

\[
\begin{align*}
  n_{lim} &= \frac{P_{sys}}{P_{n_{min}}} \\
  n_{ovp} &= \min(n_{lim}, c_{wc})
\end{align*}
\]
Simple performance prediction based on the workload scalability linear model (slope, intercept)

• For the OVP system, performance on $n_{ovp}$ nodes is:

\[ t_{ovp} = m \times n_{ovp} + b \]

• For the WC system, performance on $n_{wc}$ nodes is:

\[ t_{wc} = (m \times n_{wc} + b) \left(1 - \left(\frac{r_p}{100}\right)\right) \]

• For overprovisioning to be beneficial, speedup, $s_{ovp}$, should be greater than 1

\[ s_{ovp} = \frac{t_{wc}}{t_{ovp}} \]
Two examples of workload scaling models with the best configuration selected at each node count.

For each system power bound, we choose the best configuration (one that does not exceed the specified power bound) has been plotted (black triangle, best for power bound). We run each configuration at least three times to mitigate noise. Our median prediction error across all applications is under 7%.

A common technique for the effectiveness of overprovisioning is to run our models with the best configuration (one that does not exceed the specified power bound) under consideration for a given application (less than 2% difference). Thus, we assume that for a given application, the slope and intercept of this fitted line as the inputs to our model.

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 Section IV, we use application-specific scalability models; an overview of these models was provided in Section II-B. The values for the node power for the high-end and the older-generation node include the CPU, memory and the socket power base power. We use application-specific scalability models; an overview of these models was provided in Section IV.

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, overprovisioning) vary. For example, the benefits for an application such as SPhot are significant. However, adding more nodes is beneficial for applications such as SP-MZ. We use the ratio of the absolute packed-max, has been marked with a red triangle. Here, we assume that this limit is 48 nodes.

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 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 configuration (one that does not exceed the specified power bound) has been plotted (black triangle, best for power bound). We run each configuration at least three times to mitigate noise. Our median prediction error across all applications is under 7%.

A common technique for the effectiveness of overprovisioning is to run our models with the best configuration (one that does not exceed the specified power bound) under consideration for a given application (less than 2% difference). Thus, we assume that for a given application, the slope and intercept of this fitted line as the inputs to our model.

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 Section IV, we use application-specific scalability models; an overview of these models was provided in Section II-B. The values for the node power for the high-end and the older-generation node include the CPU, memory and the socket power base power. We use application-specific scalability models; an overview of these models was provided in Section IV.

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, overprovisioning) vary. For example, the benefits for an application such as SPhot are significant. However, adding more nodes is beneficial for applications such as SP-MZ. We use the ratio of the absolute packed-max, has been marked with a red triangle. Here, we assume that this limit is 48 nodes.

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 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 configuration (one that does not exceed the specified power bound) has been plotted (black triangle, best for power bound). We run each configuration at least three times to mitigate noise. Our median prediction error across all applications is under 7%.
Evaluation Example: we benefit if $s_{ovp}$ is greater than 1

<table>
<thead>
<tr>
<th>Workload</th>
<th>$N_{wc}$</th>
<th>$N_{ovp}$</th>
<th>$s_{ovp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU-MZ</td>
<td>18</td>
<td>30</td>
<td>1.22</td>
</tr>
<tr>
<td>BT-MZ</td>
<td>18</td>
<td>30</td>
<td>0.83</td>
</tr>
</tbody>
</table>

- LU-MZ represents workloads that scale well, BT-MZ otherwise

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{sys}$</td>
<td>7000 W</td>
</tr>
<tr>
<td>$P_{n_{max}}$</td>
<td>380 W</td>
</tr>
<tr>
<td>$P_{n_{min}}$</td>
<td>180</td>
</tr>
<tr>
<td>Cost Ratio, $r_c$</td>
<td>1.7</td>
</tr>
<tr>
<td>Performance, $r_p$</td>
<td>27%</td>
</tr>
<tr>
<td>LU-MZ model, $(m,b)$</td>
<td>(-0.542, 25.93)</td>
</tr>
<tr>
<td>BT-MZ model, $(m,b)$</td>
<td>(-0.069, 8.50)</td>
</tr>
</tbody>
</table>
Significant benefit for workloads such as LU-MZ (Cost Ratio: better when the crossover is toward the left)
No win with workloads such as BT-MZ (Cost ratio: better when the crossover is toward the left)
Significant benefit for workloads such as LU-MZ (Node performance: better when the crossover is toward the right)
No win with workloads such as BT-MZ (Node performance Better when the crossover is toward the right)
Significant benefit for workloads such as LU-MZ (Power budget: better when the crossover is toward the left)
No win with workloads such as BT-MZ (Power budget: better when the crossover is toward the left)
Summary

• Design choices: worst-case and hardware overprovisioning
  • Careful cost-benefit analysis is necessary for large-scale design

• An overprovisioned system can be built without additional cost using older-generation nodes with similar features

• Net benefit depends on several factors
  • Relative cost
  • Relative single-node performance
  • Expected workload characteristics

• More research is needed for throughput and utilization analysis