Process Variation-Aware Power Scheduling for HPC Applications

Neha Gholkar
North Carolina State University
ngholka@ncsu.edu

Frank Mueller (Advisor)
North Carolina State University
fmuelle@ncsu.edu

Barry Rountree (Advisor)
Lawrence Livermore National Laboratory
rountree1@llnl.gov

1. INTRODUCTION

Until recently research community has focused on minimizing energy usage of super computers. Considering the US DOE's mandate of power constraint of 20 MW for the exascale sites, efforts need to be directed towards minimizing the wasteful usage of power while maximizing performance under this constraint.

The research community has been looking at optimal power scheduling as a potential solution of maximizing the throughput of a system under a power constraint. Future machines may employ power schedulers that assign power budgets to each of the applications such that the aggregate power consumption of all applications remains within the DOE limit.

Most of the workloads on supercomputers are often tightly-coupled parallel scientific simulations executing on multiple nodes simultaneously. A näive strategy of enforcing power constraint for such an application is to distribute its power budget evenly across all the nodes such that the aggregate power consumption of all the nodes equals the power budget. Each node's power consumption can then be constrained to never exceed its power allocation. We refer to this as uniform power capping. While this strategy successfully constrains the power consumption of an application to its power budget, it leads to sub-optimal performance. However, in order to maximize the science done per Watt it is necessary to maximize the performance of every application under a its power budget. We define a metric called *power efficiency* which is the ratio of the performance of an application to its power consumption. Maximizing the performance of an application under a fixed power budget translates to maximizing the power efficiency of the application.

Previous research has demonstrated that process variations lead to non-uniformity in the performance of processors under any constant power bound. We observe that process variations also translate into variation in the peak power efficiency of the processors. As uniform power capping is oblivious of this variation, it fails at maximizing the power efficiency of the application.

SC '15

Problem Statement

When optimizing a parallel job for performance under a power constraint we must address at least the following questions:

1. How many nodes (say N) should a job run on?
2. What should be the power bound (say p_i) for each of the nodes?

We propose an optimal algorithm to answer the above questions.

Most of the dynamic power consumed by a node can be attributed to processors. Hence, we reduce the problem to processor level from node level.

Process Variation Under Power Bounds

We observed that process variations translate into variation in processor's peak *power efficiency*.

Processor, socket or package (PKG) in the context of this work refers to a single multi-core chip shipped by the manufacturer.

We characterized Ivy Bridge processors by executing Embarrassingly Parallel (EP) and Multigrid (MG) Solver of the NAS Parallel Benchmark suite at 13 different power bounds. Each experiment was repeated 10 times. Fig. 1 represents the averages across 10 runs. We observed performance variation of up to 30% on the cluster.

Power Efficiency

In order to characterize the processors of the cluster we establish a metric called *power efficiency* which we define as the number of instructions retired per second per Watt, i.e., IPS/W. Fig. 2 depicts the *power efficiency* of 180 processors on Catalyst for EP(left) and MG(right). The x-axis represents the power cap in Watts and the y-axis represents power efficiency in billion IPS/W.

We make the following observations from these experiments:

1. Under a power bound the performance variability translates into variation in processor's peak *power efficiency*.
2. Efficient processors are most efficient at lower power bounds while the inefficient processors are most efficient at higher power bounds.
3. The peak efficiency points for the compute-bound benchmark EP are at lower power bounds as compared to the memory-bound benchmark MG.

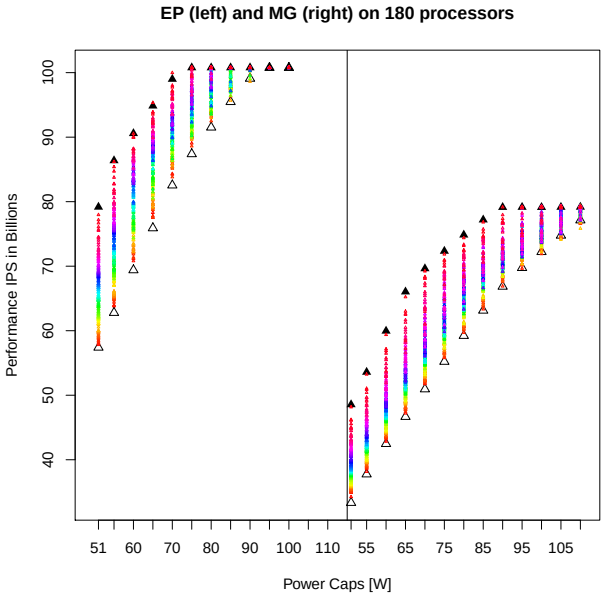

Figure 1: Performance vs Power data. Solid triangles represent the most efficient processor while the hollow triangle represents the least efficient processor. Intermediate data points represents the rest of the processors.

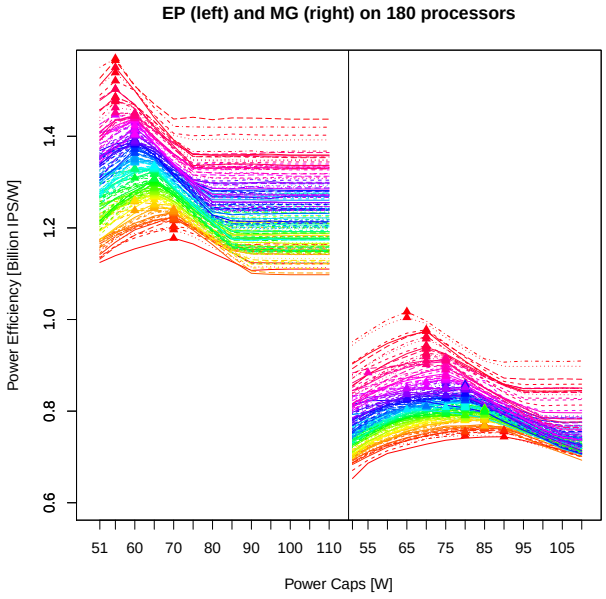

Figure 2: Power Efficiency = Billion IPS / Watt. Each curve represents a processor

The conclusion from these experiments is that uniform power capping of processors hosting a job is not optimal as all processors are not equally power efficient at any given power cap.

Based on these insights, we designed a variation aware power balancing model that determines an optimal configuration for a job.

2. POWER BALANCING

We propose an iterative algorithm to determine the optimal configuration for a job at its assigned power budget. Let PB be the power budget for a parallel job J. A job performance can be quantified in terms of the aggregate IPS of the processors that the job is scheduled on. A job's configuration can be described as $J\{N, P_{config}\}$ where N is the optimal number of processors, $P_{config} = \{p_1, p_2...p_N\}$ and p_i is the power cap of the i^{th} processor. Other parameters of the proposed model are described in Table 1.

Table 1: Model Parameters

n	number of processors in the system at any iteration
$SysP$	system power at any iteration
$\triangle P$	incremental increase in system power from one iteration to the other

We start with SysP=$\triangle P$ and n=0. We increment system power by $\triangle P$ until the system power equals the job power budget. Power balancing model determines the most effective usage of $\triangle P$ at every increment.

We first sort processors by power efficiency. Power balancing model strives to achieve maximum system IPS under a power constraint. To meet this goal, at every increment of $\triangle P$ it decides to perform one of the following.

1. To assign $\triangle P$ to one of the n processors already in the job's configuration.
2. To add $(n+1)^{th}$ processor to the system and perform power redistribution based on the resulting system IPS.

The model opts for the alternative that maximizes the system IPS. Processors are added to the job's configuration in the order of power efficiency starting from the most efficient in the earlier iterations to the least efficient in the latter iterations.

Our results are presented in Fig. 3. EP and SP were executed on 8, 16 and 32 processors at job power budgets of 4KW, 8KW and 16KW, respectively. At this scale, the observed performance improvement was up to 29% with respect to the uniform power capping approach.

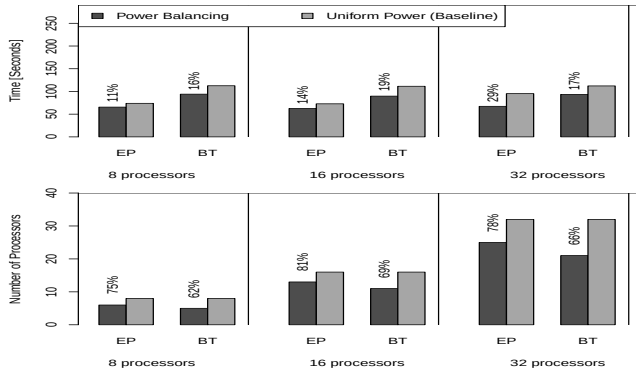

Figure 3: Performance Improvement and Optimal Number of Processors for EP and SP on 8, 16 and 32 processors