RPM: Reward Power Manager for Power Distribution over a Cluster

Sunil Kumar
IIIT Delhi, India

Vivek Kumar
IIIT Delhi, India

Sridutt Bhalachandra
Lawrence Berkeley National Laboratory, USA

I. Introduction

The amount of power required to run an exascale system has become a challenging factor, as the recommended power budget for such systems is typically around 20-30 MW. These systems must operate power-efficiently to achieve exascale computation under a limited power budget. Hardware overprovisioning is a widely used technique to address this challenge, which allows for the inclusion of more compute nodes in a cluster without exceeding the global power budget. Overprovisioning can be achieved by limiting the power allocation to each compute node below its Thermal Design Power (TDP) limit [1]. This approach is commonly known as system-wide power capping (PCAP). Modern processors now support PCAP, which enforces a user-set PCAP limit by throttling the processor's frequency at the hardware level. However, uniform power allocation across all nodes can severely impact the performance of jobs that have high power needs. In such scenarios, non-uniform power distribution across nodes while maintaining the global power budget can prevent performance degradation and power wastage.

Many researchers have proposed various power managers (PMs) for distributing power non-uniformly across the nodes to reduce slack generated due to uneven CPU usage on multicore processors [2], [3], [4], [5], [6]. Pshifter [5] and GeoPM [4] introduced PMs for managing intra-application slacks generated within an MPI application. Here, PM shifts power from MPI ranks (power donors) encountering slacks to other MPI ranks (power receivers). Inadomi et al. [2] proposed a similar approach for single MPI applications where slacks are generated not due to MPI ranks but due to performance variability in processors. There are also studies focusing on designing PMs for inter-application power distribution in co-running applications [3], [6]. To reduce inter-application slacks, PM shifts unused power from applications having slacks (power donors) to other applications fully utilizing CPUs (power receivers).

To the best of our knowledge, all prior works, whether intra-application PMs or inter-application PMs, have concentrated on improving the performance of power receivers by giving them unused power from power donors. None of the existing PMs have considered rewarding power donors who assisted power receivers in improving their performance for better quality of service (QoS). In this poster, we propose a reward power manager (RPM) to extend existing PMs for inter-application power distribution by introducing a reward policy that also improves the performance of power donors. This reward policy allows donors to reclaim the power given to the power receivers in the past when the power donor transitions back from slack to high-power usage region.

II. Design and Implementation

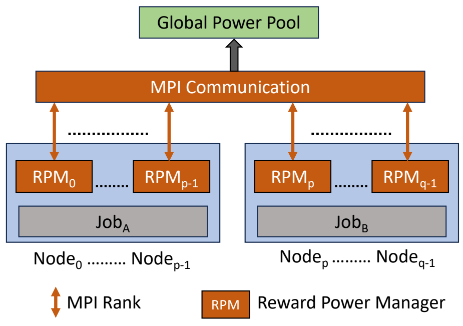


Figure 1. Framework of proposed RPM for a cluster under PCAP

This section describes the work-in-progress of our Reward Power Manager (RPM). Figure 1 demonstrates a scenario where two different parallel jobs (jobA and jobB) are running over a cluster under a user-set PCAP using different sets of nodes. Each node has its own RPM instance sharing a CPU along with the running job on that node. RPM runs periodically at a fixed interval (100 ms epoch) that monitors the node-level power usage at each epoch during the job's execution carried on the node. It detects slack in that job by monitoring the node-level CPU usage. Whenever a slack is detected, it reduces the PCAP of that node and transfers this surplus power to a global power pool. It then changes the node's state as a power donor and notifies other RPM instances running at other nodes using MPI. RPMs of the nodes running under full CPU utilization would become power receivers. The surplus power in the global power pool would then be distributed among all power receivers. In this prototype implementation, we currently distribute the power equally among the receivers. However, in future work, we will distribute the power according to the power sensitivity of jobs running in the respective nodes. Each power receiver RPM also records the total number of epochs for which they receive the donated power. When the original power donor transitions back from the slack into the high CPU utilization phase, it will broadcast its state transition to all other RPMs. Each power receiver RPM would then reward the power donor by returning back a fraction of the received power for the same number of epochs they received in the past by reducing their respective PCAP. Here, RPMs return a fraction of power than what they received. Otherwise, performance gain at the power receiver will vanish if they return the same amount of power they received in the past. Hence, RPM promotes fair power utilization and improves the quality of services (QoS).

III. Experimental Evaluation

We evaluated our current implementation by simulating a four-node cluster on a four-socket server. Each socket is an Intel Xeon 5318H Cooper Lake processor that supports PCAP in the range of (83W to 150W). We chose eight exascale proxy applications from the ECP[1] suite. These applications are SimpleMOC, RSBench, PathFinder, QuickSilver, CoMD, HPCCG, MiniFE and XSBench. They use MPI+OpenMP programming model. In our exp, we run a single MPI application (rank) at each socket with their OpenMP threads pinned to each core of corresponding sockets. We created two different mixes of apps (Mix1 and Mix2) shown in 3. Each mix is evaluated under three different user-set PCAP (55%, 60% and 65% of the TDP).

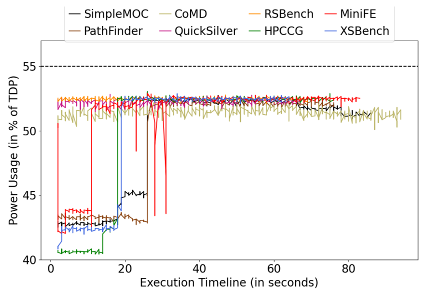
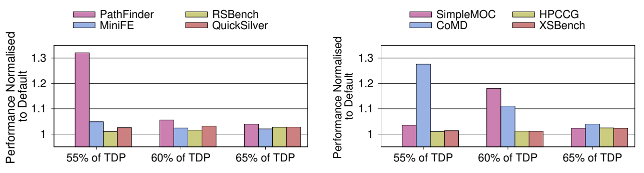


Figure 2. Power variation during application execution due to slack and CPU intensive phases

In our chosen eight applications, slack is generated during the execution of PathFinder, MiniFE, SimpleMOC, HPCCG and XSBench(see Figire 2). In Mix1, PathFinder and MiniFE are the power donors, whereas RSBench and QuickSilver are the power receivers. In Mix2, SimpleMOC, HPCCG and XSBench are power donors, whereas CoMD is a power receiver. At 55% PCAP, we observed performance gain in the range of 1 to 32% for Mix1 and 1 to 27% for Mix2. However, performance gain diminishes with increasing PCAP as the reciever application doesn't benefit much at higher PCAP. It is worthwhile to note that even under limited PCAP, RPM doesn't degrade the performance of any applications.

[1]https://proxyapps.exascaleproject.org/ecp-proxy-apps-suite/



(a) Mix1 (b) Mix2

Figure 3. Evaluation of RPM on co-running jobs at different PCAPs

IV. Conclusion and Future Work

In this poster, we presented the initial design of a Reward Power Manager (RPM) for inter-application power distribution that doesn't incur performance loss in any applications. RPM transfers the unused power at a node running under slack to other nodes running under high CPU utilization. It uses a novel reward policy to return back the power to the power donor node when it is outside the slack region. Our experimental evaluation demonstrates a performance gain of up to 32%.

In future work, we aim to extend the RPM by implementing support for power distribution according to the power sensitivity applications and implementing a hybrid of inter and intra applications power distribution. We also want to compare RPM experimentally with state-of-the-art PMs such as DPS [6]. Finally, we would like to carry out the experimental evaluation of the RPM on a real cluster.

V. Acknowledgements

This research is supported by the Google PhD Fellowship 2022 program. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation.

References

[1] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "Exploring hardware overprovisioning in power-constrained, high performance computing," in *ICS '13*, 2013, p. 173–182.

[2] Y. Inadomi, T. Patki, K. Inoue, M. Aoyagi, B. Rountree, M. Schulz, D. Lowenthal, Y. Wada, K. Fukazawa, M. Ueda *et al.*, "Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing," in *SC '15*, 2015, pp. 1–12.

[3] S. Lee, D. K. Lowenthal, B. R. De Supinski, T. Islam, K. Mohror, B. Rountree, and M. Schulz, "I/o aware power shifting," in *IPDPS '16*. IEEE, 2016, pp. 740–749.

[4] J. Eastep, S. Sylvester, C. Cantalupo, B. Geltz, F. Ardanaz, A. Al-Rawi, K. Livingston, F. Keceli, M. Maiterth, and S. Jana, "Global extensible open power manager: a vehicle for hpc community collaboration on co-designed energy management solutions," in *ISC '17*. Cham: Springer, 2017, pp. 394–412.

[5] N. Gholkar, F. Mueller, B. Rountree, and A. Marathe, "Pshifter: Feedback-based dynamic power shifting within hpc jobs for performance," in *HPDC '18*, 2018, p. 106–117.

[6] J. Ding and H. Hoffmann, "Dps: Adaptive power management for overprovisioned systems," in *SC '23*, 2023, pp. 1–14.