Toward Footprint-Aware Power Shifting for Hybrid Memory Based Systems (Extended Abstract)

Eishi Arima
The University of Tokyo, Japan
arima@cc.u-tokyo.ac.jp

Toshihiro Hanawa
The University of Tokyo, Japan
hanawa@cc.u-tokyo.ac.jp

Martin Schulz
Technical University of Munich, Germany
schulzm@in.tum.de

ABSTRACT

High Performance Computing (HPC) systems are facing severe limitations in both power and memory bandwidth/-capacity. Both limitations have been addressed individually: to exploit performance under a strict power constraint, power shifting, which allocates more power budget on the bottleneck component, is a promising approach; for memory bandwidth/capacity, the industry is gradually shifting towards hybrid main memory designs that comprise multiple technologies (e.g., DRAM and Non-Volatile RAM). However, few works look at the combination of both trends.

We propose a power management concept called *footprint-aware power shifting*, which explicitly targets hybrid main memory systems. The idea is based on the following observation: in spite of the system software's efforts to optimize the data allocations on such a system, the effective memory bandwidth decreases considerably when we scale the problem size of applications. As a result, the performance bottleneck changes among components depending on the footprint size, which then also changes the optimal power budget settings. We demonstrate the phenomenon and quantitatively show the impact of our power management approach.

1. INTRODUCTION

Power consumption is becoming one of the major design constraints when building supercomputers or High Performance Computing (HPC) systems. Therefore, to gain higher performance on such power-constrained systems, it is necessary to develop sophisticated power management schemes. The most common approaches for this are *power capping* (setting a power constraint to each job/node/component) and *power shifting* (shifting power among components depending on their needs) [4].

At the same time, we continue to face limited memory bandwidths and capacities in HPC systems. On one hand, to improve bandwidth, many vendors focus on main memories with emerging 3D stacked DRAM technologies. However, these memory systems have limited capacity-scalability compared to conventional DDR-based DRAM [3]. On the other hand, using emerging scalable NVRAM (Non-Volatile RAM, e.g., Intel 3D Xpoint memory [2]) is attractive in terms of capacity, but they are much slower than conventional DRAM. As consequence, the industry has been shifting toward hybrid memory designs: architecting main memories with multiple different technologies (e.g., 3D stacked DRAM + DDR-based DRAM [3] or DRAM + NVRAM [2]), which are usually heterogeneous in bandwidth and capacity.

© 2018 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

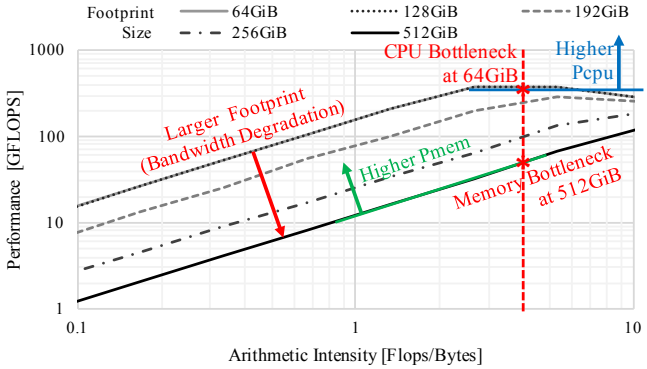

Figure 1: Roofline plots [5] measured on our hybrid memory based system.

```
#pragma omp parallel for simd
for (i = 0; i < N; i ++) {
    A[i] = A[i] * B[i] ... * B[i];
}
```

Figure 2: Tested streaming code (footprint size ∝ N, arithmetic intensity ∝ the number of * B[i])

2. BASELINES

We used a hybrid main memory based system and performed a preliminary evaluation (Figure 1). The system configuration is summarized in Table 1. The main memory consists of DDR4 DRAM and PCIe attached NVRAM (Intel 3D Xpoint Optane [2]). By using Intel Memory Drive Technology (IMDT) [2], we can use the NVRAM as a part of the main memory.

On this system, we executed a synthetic streaming code shown in Figure 2 while changing the footprint size (problem size) and the arithmetic intensity. Figure 1 describes the results. The shapes of the curves can be well-explained by the roofline model [5] — that is, for smaller arithmetic intensity, the performance is capped by the memory system bandwidth (the slope lines), but for higher arithmetic intensity, it is limited by the CPU throughput (the horizontal lines). Although the system software attempts to optimize the data mapping on the hybrid main memory, the effec-

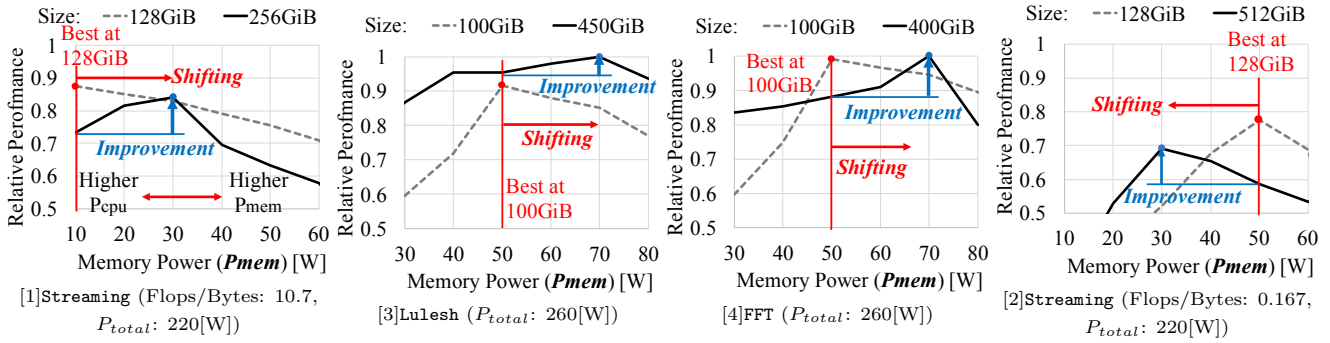

Figure 3: Power shifting result. The X-axis represents P_{mem}, and the Y-axis shows the performance which is normalized to that without power capping for each footprint size. The larger P_{mem} goes, the smaller P_{cpu} becomes as we set the constraint: $P_{total} = P_{pcu} + P_{mem}$.

tive bandwidth decreases as the footprint size scales. As a result, the slope line in Figure 1 moves toward the downside. Because of this effect, *the performance bottleneck shifts from the CPU to the memory system* even for CPU intensive workloads when we increase the footprint size.

3. APPROACH

Driven by the observation, we propose *footprint-aware power shifting* that shifts power among components *depending on the footprint size*. The concept can be formulated as follows.

$$\begin{aligned} \max \quad & Perf(S, P_{cpu}, P_{mem}) \\ s.t. \quad & P_{cpu} + P_{mem} \leq P_{total} \end{aligned}$$

The objective is to maximize performance ($Perf$), which is a function of the footprint size (S) and the power budgets of the CPU (P_{cpu}) / memory system (P_{mem}). The second equation describes the constraint given by the total power budget of the node (P_{total}). We attempt to optimize P_{cpu} and P_{mem} when the other two parameters are given. To maximize performance, *we must choose the best combination of* $\{P_{cpu}, P_{mem}\}$ *by identifying the bottleneck component, which highly depends on the footprint size* S.

4. EVALUATION

In our evaluation, we changed the combination of $\{P_{cpu}, P_{mem}\}$ under a given power constraint (P_{total}) for several workloads. To do so, we set various power cap values to each component through RAPL (Running Average Power Limit) [1]. Note that the power budget of the NVRAM was not considered, which was regarded as 0 in this evaluation. We are going to cover this in the future work. As for the workloads, we chose `FFT` from the HPCC benchmarks, `Lulesh` from the CORAL benchmark, and the synthetic code (`Streaming`) shown in Figure 2.

Table 1: System configuration

CPU Package	Xeon Gold 6154 Processor (Skylake), 18 cores, 3.0GHz, TDP 200W x2 sockets
Memory System	**DRAM:** DDR4-2666 x12 DIMMs, 12ch, 192GiB **NVRAM:** Intel Optane SSD P4800X, 375GB, 2.4GB/s(read), 2.0GB/s(write) x2 cards **Data management:** IMDT [2]
OS	Cent OS 7.4
Compiler	Intel C++/Fortran Compiler 17.0.4, **Options:** -O3 -qopenmp

Figure 3 shows the results. As shown in the graphs, the optimal combination of $\{P_{cpu}, P_{mem}\}$ changes as the data size scales for all of those workloads. For `Streaming` (Flops/Bytes: 10.7), `FFT` and `Lulesh`: shifting power from the CPU to the DRAM memory improves performance when we scale the problem size, as the CPU becomes less critical due to the bandwidth degradation (Figure 1).

An exception is the code `Streaming` (Flops/Bytes: 0.167), which is a memory intensive workload. Consequently, the DRAM cannot convert the allocated power into performance effectively due to the frequent accesses to the NVRAM when the footprint size is larger than the DRAM capacity. In contrast to this, the DRAM can utilize larger power to improve performance at 128GiB as the memory system is still the bottleneck and the NVRAM is rarely accessed. Therefore, we need to consider (1) *the bottleneck shifting between CPU and memory for CPU intensive workloads* and (2) *the DRAM/NVRAM access rate especially for memory intensive workloads* when allocating power.

5. CONCLUSION AND FUTURE WORK

In this article, we proposed a power management concept called *footprint-aware power shifting*. According to our evaluation, the approach is promising to improve the performance of power-constrained hybrid memory based systems. In the future work, we are going to develop a software framework, a performance-power model, and a power allocation algorithm to realize our proposal.

Acknowledgements

This work is partly supported by JSPS Grant-in-Aid for Research Activity Start-up (JP16H06677), JSPS Grant-in-Aid for Early-Career Scientists (JP18K18021), and Research on Processor Architecture, Power Management, System Software and Numerical Libraries for the Post K Computer System of RIKEN.

6. REFERENCES

[1] Intel. Intel® 64 and ia-32 Architectures Software Developer's Manual, System Programming Guide, 2017.

[2] Intel. Intel® Memory Drive Technology, Set Up and Configuration Guide, 2017.

[3] Jeffers, J., et al. *Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2016.

[4] Lefurgy, C., et al. Power Capping: A Prelude to Power Shifting. *Cluster Computing 11*, 2 (2008), 183–195.

[5] Williams, S., et al. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Communications of the ACM 52*, 4 (2009), 65–76.