PowerPrep: A power management proposal for user-facing datacenter workloads

Vineetha Govindaraj
Pennsylvania State University
University Park, PA, USA
vineetha.govindaraj@gmail.com

Sumitha George
North Dakota State University
Fargo, ND, USA
sumitha.george@ndsu.edu

Mahmut Kandemir
Pennsylvania State University
University Park, PA, USA
mtk2@psu.edu

John Sampson
Pennsylvania State University
University Park, PA, USA
jms1257@psu.edu

Vijaykrishnan Naryanan
Pennsylvania State University
University Park, PA, USA
vxn9@psu.edu

2021 IEEE International Conference on Networking, Architecture and Storage (NAS) | 978-1-7281-7744-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/NAS51552.2021.9605364

***Abstract*—Modern data center applications are user facing/latency critical. Our work analyzes the characteristics of such applications i.e., high idleness, unpredictable CPU usage, and high sensitivity to CPU performance. In spite of such execution characteristics, datacenter operators disable sleep states to optimize performance. Deep-sleep states hurt performance mainly due to: a) high wake-latency and b) cache warm-up after exiting deep-sleep. To address these challenges, we quantify three necessary characteristics required to realize deep-sleep states in datacenter applications: a) low wake-latency, b) low resident power, and c) selective retention of cache-state. Using these observations, we show how emerging technological advances can be leveraged to improve the energy efficiency of latency-critical datacenter workloads.**

***Index Terms*—Non-volatile Memory, Sleep States, Datacenter**

I. Introduction

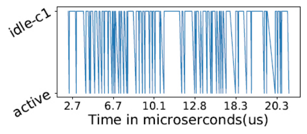
The growth of cloud and internet services has induced a paradigm shift in the nature of applications executing in datacenter servers. Application footprints are now dominated by sub-millisecond latency-critical/user facing workloads rather than traditional, long running batch jobs. Google claims this as the *"era of the killer microsecond"* [1] as the technology stack in existing datacenters is not well suited to serve requests demanding sub-millisecond service times.

The primary performance metric that characterizes the execution efficiency of user facing workloads is tail latency, rather than average request latency. Most applications utilize a metric like 99^{th} percentile tail latency, i.e., the service time of the 99^{th} slowest request out of every 100 in a small window of execution time (typically one second). Due to such strict Quality of Service (QoS) targets, datacenter servers are kept lightly loaded with utilization between 10% to 50% [2], [3]. As a result, the energy efficiency of servers take a huge hit injecting long periods of idleness in between subsequent requests.

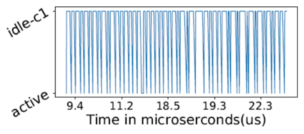
This work was partially sponsored by the National Science Foundation (NSF) under grant number 1822923.

To study the idleness existing in between requests in datacenter workloads, we created an experimental setup executing `memcached` a popular key value store application in a server for high load(500k qps) and low load scenarios(10k qps). Our experimental setup allowed us to surgically enable/disable processor C-states in different CPU cores. The goal of this experiment was to find out if there is idleness and the nature of residencies in shallow and deep C-states. We studied 3 scenarios: a) enable C1 shallow sleep states only b) enable shallow (C1&C3) sleep states only, c) enable shallow and deep sleep states (C1&C3&C6). Figure 1 illustrates that user facing data center workloads, as we expected, contain a lot of idleness.

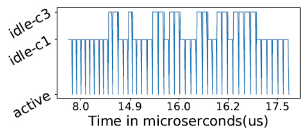
More importantly the observed idleness is short for high load scenarios due to frequent request arrivals. For low loads, there are enough idleness to harness deeper sleep states.As the load increases, the static power becomes less dominant because of the decrease in idle time.



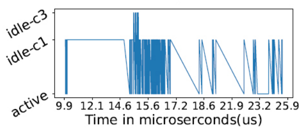
(a) C1 at low load



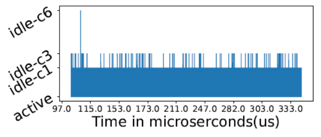
(b) C1 at high load



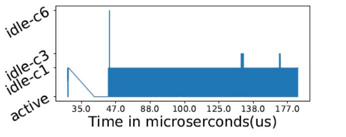
(c) C1&C3 at low load



(d) C1&C3 at high load



(e) C1,C3&C6 at low load



(f) C1,C3&C6 at high load

Fig. 1: Characterization of CPU sleep states for memcached under low and high loads

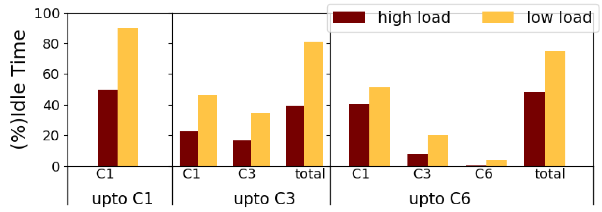


Fig. 2: % residency in a C-State when maximum enabled C-states are C1,C3 and C6

To improve energy efficiency, modern CPUs are designed to support many advanced power management features such as dynamic voltage frequency scaling (DVFS), many different deep sleep states, and on-die voltage regulation. These techniques work very well for predictable periods of short and long idleness or in situations with forgiving latency requirements. However, user facing data center workloads' unique characteristics such as high but unpredictable idleness and the desire for low latency to the 99^{th} percentile while consuming low power cannot be met adequately by modern CPUs. Specifically, deep sleep states have low resident power but they also have high wake latency (i.e. time taken to transition from deep sleep state to compute state). As a result, datacenter operators disable the deeper sleep states above C1, trading off energy efficiency for meeting QoS and SLA targets. However, this sub-optimal approach is wasteful and more importantly expensive - power costs are a key contributor to total cost of ownership [3].

To tackle the observed gap between energy efficiency and QoS, this work proposes three key attributes that the current power management algorithm needs to have to improve energy efficiency of latency critical data center workloads namely: a) low resident deep sleep state power, b) low wake latency (fast transition from deep sleep to compute state) and c) selective retention of cache-state. Our extensive analyses show that the time to warm-up the instruction cache is one of the key contributors to degrading 99^{th} percentile tail latency in user facing datacenter workloads. This work also leverages recent advances in on-die per-core voltage regulation and shows that wake latencies from deep sleep states on the order of 10us is possible [4] - thereby allowing the realization of deep sleep states with low wake latency. Furthermore, in this work, we show how emerging technologies, such as embedded non-volatile memories can further reduce energy consumption if used along with the deep sleep state.

Specifically, this paper makes the following contributions:

1) We portray the inefficiencies of modern CPU power management techniques towards exploiting idleness in datacenters while executing user facing workloads. This enabled us to illustrate several fundamental observations as to why state-of-the-art CPUs cannot achieve optimal energy consumption for user facing workloads.
2) To the best of our knowledge, our work is the first of its kind that identifies key characteristics of `CPU sleep states` that modern datacenters should posses in order to support peak performance under optimal power constraits.
3) Additionally, we show that leveraging the advances present in on-die voltage regulation, retention cache circuits can help realize a low resident power and low wake latency deep sleep state. Furthermore, we show how embedded non-volatile memory technology can further improve energy efficiency if used along with the deep sleep state.
4) Finally, we show the impact of our proposed methods on real datacenter workloads and how they significantly outperforms existing techniques. Leveraging these observations, we were able to draw inferences towards handling energy dis proportionality, by enabling micro-architectural modifications instead of existing software based techniques.

II. Background And Motivation

A. Modern Datacenter Workloads

Datacenters house a wide variety of applications of varying nature. These applications are broadly classified into two types: (1) batch applications and (2) user-facing applications. Traditional datacenter applications were predominantly of batch type, throughput-oriented, and not user-facing. Their execution performance/Quality of Service (QoS) is determined by metrics like processor Instructions per Cycle(IPC), operations per second etc. However, modern cloud-based online data intensive applications are user facing and latency critical. These applications serve millions of users' requests, each of which may require efforts spanning from several microseconds to a few milliseconds. For such applications, QoS is predominantly determined by service times for these requests.

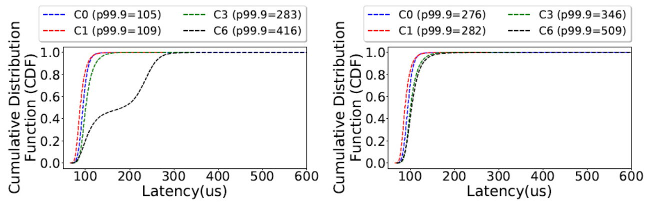
One such important user-facing application that executes in modern datacenters is `Memcached` [5] – a key value store application. The primary metric that quantifies memcached's performance is 99th percentile tail latency. For a given sample of time during which the application is executing, its 99th percentile tail latency is defined as the time within which 99% of the requests in that sample were able to complete execution.

B. Processor Power States Transition

Advanced power management allows modern CPUs to achieve high performance under a power envelope. While the specifics of power management differ, in this sub-section we explain the fundamentals of CPU power management. A CPU can exist in one of 3 power states - compute (C0), clock-gated (C1), power-gated (C6). C0 is the only state in which a CPU core can run a workload. In C1 the clock going to a CPU core is gated to save dynamic power. Cutting off the clocks to a core that has no work to do helps save power and improves energy efficiency. In C6, the power to a CPU core is cut-off, saving leakage power and hence significantly improving energy efficiency. The algorithm running on the power management unit (PMU) of a CPU orchestrates state transitions from $C0 -> C1$, $C0 -> C6$, $C6 -> C0$ etc.

Low power states like C1 or C6 are characterized by two essential attributes - wake up latency and resident power. Resident power of a core is the power consumed by a core when it is in C1. Wake latency is the time it takes for a core to exit a low power state (like C1 or C6) and become available to serve a request. As it may be apparent, in an ideal world, we want a low power C state to have extremely low resident power and wake latency. However, in the real world, bound by the laws of physics, its very difficult to achieve low resident power and wake latency for a given C-state. For instance, the power-gated state C6 has a low resident power (C6 power $<<$ C1 power), but it takes a much longer time to charge its power rails back to VDD, undergoing a soft reset to be available to serve to compute (C6 wake latency $>>$ C1 wake latency). Keeping this in mind, there have been significant advances in power management over the last decade that continue to lower resident power while reducing wake latency. A few methodologies used for this purpose include utilizing on-die voltage regulators, designing glitchless PLLs, including snoop caches etc.

Another key metric in power management is *residency time* – the time spent by a core in a particular low power state. In an ideal world we want a core to spend as little time possible in compute state (C0) to finish its work, transitioning into a deep sleep state like (C6) when idle. However, the high wake up latencies of C6 make it difficult to implement power gating on servers executing user facing applications that possesses sub-millisecond latency targets as shown in Figure3. This is because requests expect CPU cores to be readily available upon their arrival and hence, transitioning from deep sleep states to an active state will take a huge toll on their service times. User facing data center workloads like `memcached` exhibit such behavior and hence servers hosting such applications disallow CPU cores to transition into deep sleep states. While this meets latency requirements - it comes at the expense of high power consumption.



(a) Latency CDF at low load(10kqps).

(b) Latency CDF at high load(500kqps)

Fig. 3: Cumulative distribution of query latency at different C-States for memcached. Deeper C-States induce higher tail latencies.

III. Observations

In this work, we strive to lay down the requirements for a meaningful low power state that enables maximum power savings while minimizing performance impact.

In a typical CPU, there are multiple steps involved in placing a core in a deep sleep state, such as C6. When the core has been idle and clock-gated for a certain length of time, the power management algorithm determines to place the core in the C6 state. At this time, the modified data in the private caches are flushed to the shared caches. Besides flushing the caches, the core's architectural state, including the machine state registers (MSRs) are copied into the shared cache or some other persistent structure to retain state. At this point, the core is in the C6 state [6]. When the CPU receives an interrupt to resume processing, the core is brought out of C6 by ramping up power and restoring the core's architectural state. From a power standpoint, a deep sleep state such as C6 enables significantly low power consumption. However, two factors impact performance when the core exits a deep sleep state such as C6: a) cold caches resulting from the cache flush before C6 entry, and b) high latency to ramp up the power on the core and restore architecture state, i.e., wake latency. We make three observations that enable us to define a meaningful deep sleep that can maximize power savings while minimizing performance impact for user-facing datacenter workloads.

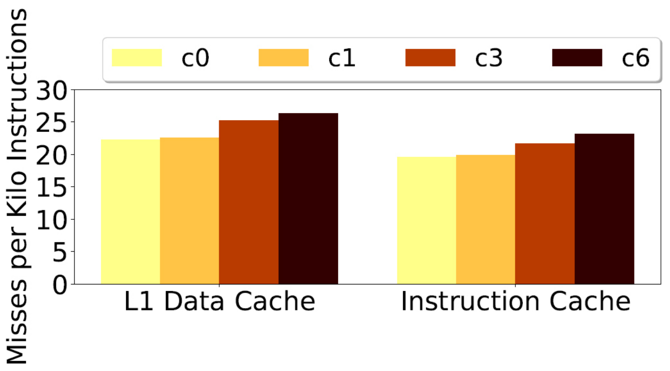
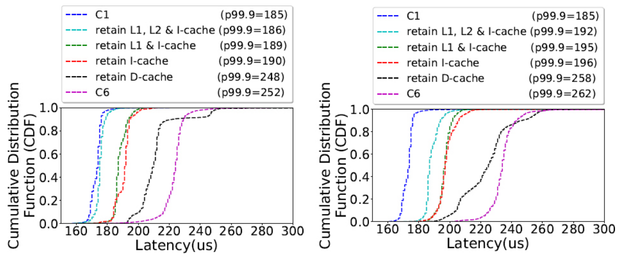


Fig. 4: L1 Data and Instruction Cache MPKI.



(a) 10us wakeup.

(b) 40us wakeup.

Fig. 5: Cumulative Distributions of query latency in memcached for different wakeup latency.

A. Observation 1: Workload dependent retention of cache state can enable significant power and performance gains

To quantify the performance impact of cold cache misses, we study the cache miss rate (misses per kilo instruction) on a modern data center CPU under 4 different scenarios: a) C states disabled, b) C1 enabled, c) C1 and C3 enabled, and d) C1, C3 and C6 enabled. From Figure 4, we observe that the deeper the sleep state enabled, the greater the cache miss rate for both the instruction and data caches. However, we also observed that significant portion of this cold cache miss impact can be mitigated by retaining *only the Icache* state for user-facing datacenter workloads like Memcached.

To demonstrate the impact of retaining the contents of different caches, we studied 6 different scenarios in an in-house simulator based on Dist-Gem5 [7]: 1) C1 is enabled When C6 is triggered(baseline), 2) Contents of L1 I, D and L2 caches are retained when C6 is triggered, 3) L1 I and D cache contents alone are retained when C6 is triggered, 4) L1 Icache contents alone are retained when C6 is triggered, 5) L1 Dcache contents alone are retained when C6 is triggered, and 6) no cache contents are retained, when C6 is triggered (baseline C6).

Figure 5b, shows the impact of each scenario, on average as well as 99^{th} percentile tail latency. It can be observed that the 99^{th} percentile tail latency increases by less than 4%, as compared to enabling C1. This essentially enables C1-like latency characteristics, while enabling C6-like power characteristics.

B. *Observation 2: Low Wakeup latency is critical to minimize performance impact*

To study the impact of wakeup latency on performance, the aforementioned scenarios were run with different wakeup latencies. From Figure 5a, we can observe that, for scenarios 2, 3 and 4, the tail latency improves to a great extent and is close to the baseline. As per Figure 5, as we retain more private cache content, we can see latencies similar to baseline. Generally, workload developers and system operators only care about avoiding the SLO latency violations than average latency. Retaining the contents of all the private caches can increase the resident power of the deep sleep state (unless we use NV-SRAMs), as shown in Figure 6. As can be observed, of all the cache contents, preserving the L1 Icache provides the maximum performance benefit for Memcached in terms of retention power and tail latency. Thus, based on our previous observation and Figure 6, we can conclude that retaining the Icache contents is sufficient for user-facing workloads like Memcached.

C. *Observation 3: Low resident power in a deep-sleep state is critical to maximizing power savings*

Power consumed in the deep sleep states should be as low as possible. As shown in Figure 7, as we transition to deeper power states, we save more power. It is to be noted that, the power-gating efficiency should be as high as possible to minimize leakage power. Hence, low resident power for a deep sleep state is key to improving energy efficiency.

Collectively, we observe that low wakeup latency and low resident power in a deep sleep state along with cache retention are necessary conditions to reach an energy efficient system.

IV. Mechanism

We propose a *power management state*, c1',which does a complete power gate of the core like c6 but also adheres to three key observations: a) low wake latency, b) low resident power, and c) workload-dependent retention of cache state. We evaluate the plausible energy savings if such a power management technique were to be implemented in a modern CPU. For

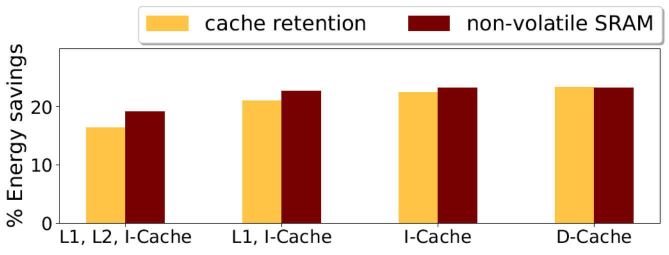


Fig. 6: %Energy saved in different cache retention scenarios normalized to baseline C1.

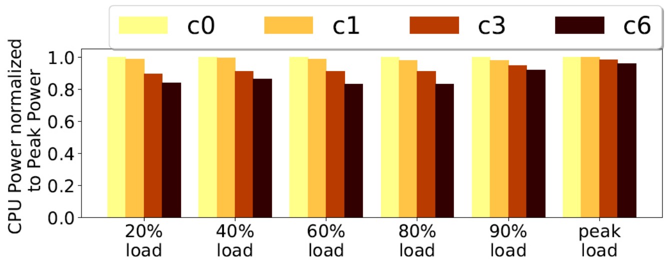


Fig. 7: CPU power when different C-states are enabled across loads for memcached.

this, we first discuss the current state of art technologies which could help us realize such a technique.

a) Wakeup-latency is largely governed by the time taken to ramp up the voltage to the designated level. Continuous innovations are being made to reduce entry and exit latencies from a given sleep state. For example, Intel Haswell achieves 20x more power savings, due to reduced wakeup latency [8]. Recent advances in on-chip voltage regulators are reducing wake latency to 10's of ns.

b) With the inception of voltage islands [9], it has become feasible to independently manage the power-performance of different cores in a CPU. For instance, it is a common practice today to power gate idle cores in a CPU, while having other cores actively serve requests. This matters greatly because the power-grid capacitance of the power island for one core is an order of magnitude smaller than the power grid capacitance of an 8-core CPU. The smaller power-grid capacitance presented by a per-core power island reduces the load on the voltage regulator. This allows for rapid replenishment of charge and allows for shorter wake latency (from C6 to C1).

c) Caches are primarily implemented today using CMOS static random access memories (SRAMs) or register files (RFs). A conventional SRAM is volatile, i.e., it loses its state when power is gated off. Power constrained modern SoCs implement CMOS caches with retention capability; these caches can retain state if the main power supply for a CPU core is gated off [10]. Typically a retention RAM [4] uses an additional always-ON power supply to retain state and operates in the near threshold region. The advantage of the retention rail is that it is implementable today, but it has the shortcoming that, even when it is in retention state, the SRAM still consumes leakage power (though the retention RAM leakage is 50% lower than the compute state leakage). It is noteworthy that retention SRAMs/RFs are a suitable implementation when sleep state residencies are of the order of us (micro-seconds). They are not suitable when sleep state residency exceeds 10ms or seconds, as their leakage power will dominate the total energy consumption. RAMs implemented with emerging non-volatile memory technologies (NV-SRAM or NV-RF) can retain state while consuming no retention power. This makes them an ideal candidate for state retention in deep sleep states.

NV-SRAMs implemented with MRAM and RRAM typically suffer from the challenge of high write latency. The cost to backup and restore data are also high due to static current.However, recent work with FeFET-NVSRAMs [11] addresses these limitations. FeFET is an emerging CMOS transistor which has the capability to completely eliminate static current in the NV-SRAMs since the conducting route of drain-source is separated from the gate control signal.Since this type of memory storage would be beneficial for our power management system in terms of low resident power at our proposed c1' state , we use FeFet-NVSRAMS [11] for retaining data. One of the major tradeoffs of using an non volatile SRAM is the extra write energy spent for backup and restore. Energy to backup and restore is only 58.7 aJ which corresponds to a break even time of 33.8 us at 0.3V standby CMOS [11]. Since, we are targeting sleep state residency in the order of ms, we can get a positive energy gain from FeFet-NVSRAM [11].

In Figure 8, we delineate the series of stages pertaining to normal clock-gating state c1 and our proposed state c1'. Given small wakeup latency, the sub-us delay added to the total access latency will be negligible, compared to the tail latency from Figure 5.

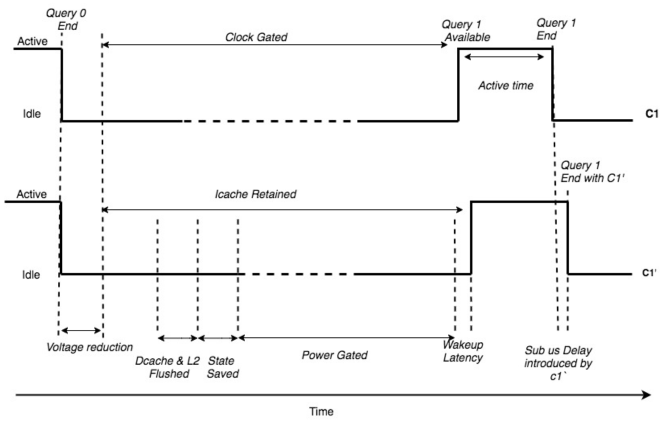


Fig. 8: CPU behaviour in C1 and C1'.

V. Results and Analysis

In this section, we first describe the ßreal system experimental setup that we utilized to determine the impact of processor sleep states on the tail latency of `memcached` and `apache web server`. Following that, we illustrate our simulation infrastructure using which we quantify the energy gains that we obtain by using PowerPrep.

A. Evaluation Methodology

Hardware Infrastructure. Our workloads are executed on server grade Intel Xeon CPU E5-2670 nodes. Each node consists of 24 physical cores. We switch off hyper-threading for all our experiments.

Performance Measurement tools. We use Intel VTune amplifier [12] to obtain and visualize idle CPU cycles that exist in between subsequent requests. Additionally, we utilize `perf` to measure microarchitectural events (cache hits, cache misses) happening in the server.

Workloads. We evaluate our technique on two important real system workloads – `memcached` and `apache web server`. `Memcached` is an in-memory key value store infrastructure, and `apache web server` is an HTTP server that is widely used for hosting websites.

Simulation Infrastructure. To simulate the performance behavior of these two workloads under our proposed cache retention and non-volatile SRAM schemes, we use a distributed cycle-accurate microarchitectural simulator called dist-gem5 [7]. Dist-gem5 [7] simulates a distributed setup of a gem5 client, which issues requests to a gem5 server that hosts the applications `memcached` and `apache web server`. Additionally, we use an in-house power model based on McPAT [13] and CACTI to calculate static and runtime power for processors containing an SRAM cache at 10nm technology. We also utilize the same tool to obtain power values for NVSRAM FeFET technology at 10nm.The energy measures are obtained through the weighted sum of the duration idle, busy and transition period with their corresponding power consumption.

B. Experimental Results

To show the effectiveness of our scheme, we evaluate energy gain and QoS (99th percentile tail latency). Further, we compare the results that we have obtained with state-of-the-art techniques DynSleep [14] and μDPM [2] respectively. Additionally, we assume that the maximum energy that can be saved for a given load is depicted by C6.

1) Power Savings: DynSleep [14] takes advantage of the fact that tail latency is an order of magnitude higher that the service time of 95th percentile request. It enters deep sleep state during this latency slack to save power. However, there are two major drawbacks in this technique. First, there are very few scenarios where continuous chunks of idleness that exceeds the residency time of C6 exists. Hence the effective power sad by DynSleep [14] is very low. Second, this scenario gets worse as the loads increase, reducing the amount of power saved to a great extent.

Figure 10 illustrates this scenario. The vertical axis represents %energy saved for memcached and apache, *normalized* to the energy consumption of baseline. The horizontal axis compares % of energy saved by the techniques DynSleep,

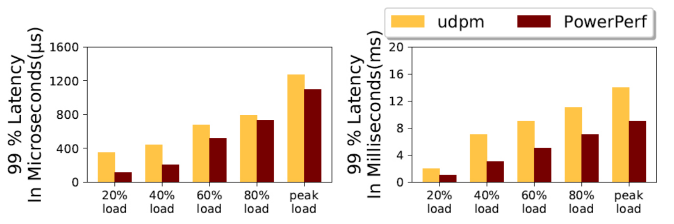


Fig. 9: 99^{th} percentile latency between PowerPerf and μDPM.

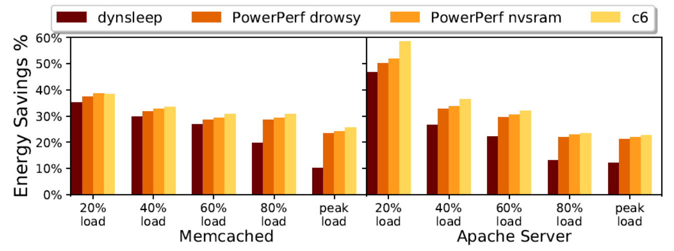


Fig. 10: Energy savings across different power management schemes.

PowerPrep with drowsy cache and PowerPrep with NVSRAM, across different loads for the applications memcached and apache web server. We can see from Figure 10 that PowerPrep outperforms DynSleep in every load scenario. PowerPrep's energy savings are extremely significant compared to DynSleep in high load scenarios. Also, we can see from Figure 10 that the energy saved by PowerPrep NVSRAM is close to C6. Hence, FeFET NVSRAM has great potential to save leakage power.

2) Tail Latency Reduction: μDPM overcomes the shortcoming of DynSleep by utilizing fine-grained power management schemes and relaxing SLO constraints. For that purpose, μDPM leverages DVFS, clock-gating as well as power-gating. Such an aggressive power management scheme takes a toll on the 99th percentile latency, which is plotted in Figure 9. The horizontal axis in Figure 9 evaluates μDPM and PowerPrep under different load scenarios, while the vertical axis indicates the 99th% tail latencies of applications `memecached` and `apache web server`, respectively. From Figure 9, we can observe that PowerPrep outperforms μDPM in every execution scenario.

VI. Related Work

There has been a wide body of prior work that tries to save power for datacenter workloads. We broadly divide them into the following categories.

Dynamic Voltage Frequency Scaling (DVFS). Dynamically scaling CPU voltage and frequency to save power is a widely studied research topic. For this purpose, Adrenaline [15] selectively pinpoints requests with long tail latencies, which are then boosted by increasing the clock frequency at which they operate. On the other hand, Pegasus [16] constantly monitors workloads and the performance statistics of recent requests within a sliding window. Using this information it leverages a feedback controller to selectively execute queries under lower operating frequency without violating the QoS requirement. Rubik [17], on the other hand utilizes the queuing delay of requests to dynamically reduce/increase voltage and frequency to save power. Most prior works [15]–[19] only look at voltage and frequency scaling techniques for which the energy savings obtained are much lower than for techniques that utilize clock gating, power gating, or otherwise deactivate entire components.

Clock Gating and Power gating techniques. A slightly orthogonal body of prior work utilizes fine grained power management schemes for datacenter workloads through the use of per-core sleep states (C-states). Dynsleep [14] reorders requests by postponing requests with long latency slacks. This introduces wide gaps of idleness during which shallow/deep sleep states can be reached in order to save power. μDPM [2], on the other hand, performs power management at a finer granularity by utilizing both per-core sleep states and DVFS. μDPM [2] performs statistical predictions to create idleness in the core by postponing the requests. However, in our experimental setup, Dynsleep [14] performed poorly at higher loads due to the reduction of idleness of the core and, in Figure 9 we can see that the 99% tail latency of μDPM [2] is consistently higher than PowerPref. Thus,μDPM [2] was not able to guarantee SLO in every scenario. These differences highlight the importance of considering the impact of nonvolatile memory technologies on datacenter power management approaches.

VII. Conclusion

We address the issue of energy proportionality for latency critical workloads in datacenters by presenting PowerPrep. In this regard, we enumerate key requirements that should drive sleep state transitions. Namely, a)low wakeup-latency, b) low resident power and c) selective retention of microarchitectural states. Using these observations, we illustrate that when such requirements are met, we can seamlessly transition into deep sleep states and save significant amount of datacenter energy while guaranteeing user-defined SLOs. From our experiments, we were able to save up to 50% of energy for widely used latency-sensitive datacenter applications like `memcached` and `apache web server`, without negatively impacting QoS. We believe that the PowerPrep technique would contribute towards saving significant amounts of power in underprovisioned datacenters and would better enable them to execute "killer microsecond workloads" without sacrificing efficiency.

Acknowledgements

We thank our anonymous reviewers for their feedback and suggestions.This work was partially sponsored by the National Science Foundation (NSF) under grant number 1822923.

References

[1] L. A. Barroso, M. Marty, D. A. Patterson, and P. Ranganathan, "Attack of the killer microseconds." *Commun. ACM*, vol. 60, no. 4, pp. 48–54, 2017.

[2] C.-H. Chou, L. N. Bhuyan, and D. Wong, "μdpm: Dynamic power management for the microsecond era," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2019, pp. 120–132.

[3] L. A. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 4, no. 1, pp. 1–108, 2009.

[4] P. A. Meinerzhagen, C. Tokunaga, A. Malavasi, V. Vaidya, A. Mendon, D. Mathaikutty, J. Kulkarni, C. Augustine, M. Cho, S. T. Kim *et al.*, "An energy-efficient graphics processor in 14-nm tri-gate cmos featuring integrated voltage regulators for fine-grain dvfs, retentive sleep, and v_{MIN} optimization," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 144–157, 2018.

[5] B. Fitzpatrick, "Distributed caching with memcached," *Linux J.*, vol. 2004, no. 124, pp. 5–, Aug. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1012889.1012894

[6] M. Arora, S. Manne, I. Paul, N. Jayasena, and D. M. Tullsen, "Understanding idle behavior and power gating mechanisms in the context of modern benchmarks on cpu-gpu integrated systems," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, 2015, pp. 366–377.

[7] A. Mohammad, U. Darbaz, G. Dozsa, S. Diestelhorst, D. Kim, and N. S. Kim, "dist-gem5: Distributed simulation of computer clusters," in *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2017, pp. 153–162.

[8] N. Kurd, M. Chowdhury, E. Burton, T. P. Thomas, C. Mozak, B. Boswell, P. Mosalikanti, M. Neidengard, A. Deval, A. Khanna *et al.*, "Haswell: A family of ia 22 nm processors," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 49–58, 2014.

[9] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn, "Managing power and performance for system-on-chip designs using voltage islands," in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*. ACM, 2002, pp. 195–202.

[10] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: simple techniques for reducing leakage power," in *Proceedings 29th Annual International Symposium on Computer Architecture*, 2002, pp. 148–157.

[11] X. Li, K. Ma, S. George, W. Khwa, J. Sampson, S. Gupta, Y. Liu, M. Chang, S. Datta, and V. Narayanan, "Design of nonvolatile sram with ferroelectric fets for energy-efficient backup and restore," *IEEE Transactions on Electron Devices*, vol. 64, no. 7, pp. 3037–3040, July 2017.

[12] J. Reinders, *VTune performance analyzer essentials*. Intel Press, 2005.

[13] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2009, pp. 469–480.

[14] C.-H. Chou, D. Wong, and L. N. Bhuyan, "Dynsleep: Fine-grained power management for a latency-critical data center application," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. ACM, 2016, pp. 212–217.

[15] C. Hsu, Y. Zhang, M. A. Laurenzano, D. Meisner, T. Wenisch, J. Mars, L. Tang, and R. G. Dreslinski, "Adrenaline: Pinpointing and reining in tail queries with quick voltage boosting," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2015, pp. 271–282.

[16] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis, "Towards energy proportionality for large-scale latency-critical workloads," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. IEEE, 2014, pp. 301–312.

[17] H. Kasture, D. B. Bartolini, N. Beckmann, and D. Sanchez, "Rubik: Fast analytical power management for latency-critical systems," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2015, pp. 598–610.

[18] T. Kolpe, A. Zhai, and S. S. Sapatnekar, "Enabling improved power management in multicore processors through clustered dvfs," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[19] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, "Multiscale: Memory system dvfs with multiple memory controllers," in *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '12. New York, NY, USA: ACM, 2012, pp. 297–302. [Online]. Available: http://doi.acm.org/10.1145/2333660.2333727