NuPow: Managing Power on NUMA Multiprocessors with Domain-Level Voltage and Frequency Control

Changmin Ahn[0000−0001−6125−0386], Seungyul Lee[0000−0002−5152−6841], Chanseok Kang[0000−0003−1106−1258], and Bernhard Egger ✉[0000−0002−6645−6161]

Seoul National University, Republic of Korea
{changmin,seungyul,chanseok,bernhard}@csap.snu.ac.kr

Abstract. Power management and task placement pose two of the greatest challenges for future many-core processors in data centers. With hundreds of cores on a single die, cores experience varying memory latencies and cannot individually regulate voltage and frequency, therefore calling for new approaches to scheduling and power management. This work presents NuPow, a hierarchical scheduling and power management framework for architectures with multiple cores per voltage and frequency domain and non-uniform memory access (NUMA) properties. NuPow considers the conflicting goals of grouping virtual machines (VMs) with similar load patterns while also placing them as close as possible to the accessed data. Implemented and evaluated on existing hardware, NuPow achieves significantly better performance per watt compared to competing approaches.

Keywords: Many-core processors · Power management · NUMA

1 Introduction

The past decade has brought a shift from high-performance single-core processors to chip multiprocessors (CMPs) integrating from a few tens up to a thousand cores into one processor die [6,10,30,2]. Increasing the number of cores leads to larger memory bandwidth requirements; CMPs thus support multiple memory controllers that are connected to the cores by a network-on-chip (NoC) [3]. Depending on the location of the issuing core and the accessed memory controller, large differences in access latency are observed, resulting in a NUMA architecture on a single chip.

Chip-level power and thermal constraints have become one of the primary design constraints and performance limiters [2]. To reduce overall chip energy consumption, processors support dynamic voltage and frequency scaling (DVFS) of clocked resources. Depending on the utilization of the cores, the voltage and the frequency of a core is set to minimize the power consumption while meeting given performance requirements [5]. The hardware required to allow per-core voltage regulation on CMPs with tens or hundreds of cores is becoming too costly [21]; instead, multiple-voltage multiple-frequency (MVMF) designs have been proposed that require all cores within a domain to operate at the same level [13,32,14,9]. In the following, we refer to CMPs that support per-core DVFS control as *core MVMF* CMPs and to those that only allow per-domain DVFS control as *domain MVMF* CMPs.

Managing power on CMPs has received considerable attention. Existing work foremost focuses on minimizing power consumption or optimizing performance for a given power budget [17,27,31,11,24]. Solutions for domain MVMF CMPs combine DVFS with thread migration [16,7,18,31,20] to allow for better tailored DVFS settings by

co-locating threads with similar performance requirements in the same domain. Power management techniques for existing CMPs fall short for a number of reasons when applied to future CMPs with hundreds or thousands of cores. Most works assume core MVMF CMPs which limits their applicability to domain MVMF CMPs. In addition and to the best of our knowledge, no work considers the NUMA properties of CMPs when managing power for domain MVMF CMPs, resulting in core mappings that are not optimal with respect to the locality of the data accessed by individual threads.

This paper presents NuPow, a hierarchical power management technique that has been built from ground up for domain MVMF CMPs with NUMA properties. NuPow can be applied to SMP systems as well as non-coherent memory architectures running individual VMs. We demonstrate the feasibility of the technique by providing and evaluating a working implementation on the Intel Single-Chip Cloud Computer (SCC) [14]. Even though this prototype chip is a decade old, its architecture resembles that of proposed CMPs with hundreds of cores; the Intel SCC can thus serve as a concept vehicle to demonstrate the effectiveness of the presented approach. All experiments and measurements are performed on the architecture itself; i.e., are not simulated and include all overhead incurred by DVFS transitions, cold cache misses, VM migration, and the different power management controllers. We compare the proposed technique to a DVFS-only approach [16] and a method that combines DVFS with VM migration [20]. Executing load patterns observed in Google's data centers [33], we achieve, on average, a 51, 33, and 10% higher performance-per-watt ratio over standard Linux, DVFS-only, and NUMA-unaware DVFS with VM relocation at no performance degradation.

2 Motivation and Related Work

2.1 Characteristics of Chip Multiprocessors

Technology scaling, thermal limitations, and the insight that doubling the logic in a processor core only delivers about 40% more performance have led to the introduction of chip multiprocessors with tens or hundreds of cores on one processor die [3,4]. Architectural characteristics of today's and future many-core CMPs impose new restrictions on the design and implementation of operating systems, in particular, with respect to task scheduling and power management.

The cores of a CMP are typically organized in a two-dimensional array. The Kilocore processor, for example, arranges its 1000 cores on a 32x32 grid [2]. A network-on-chip connects the cores and is used both for inter-core communication and accesses to memory and external devices such as network or storage controllers. The flow of data packets through the NoC is controlled by routers; this routing comes with a small delay. As a consequence, the distance and topology of the NoC between the source and the destination can have a significant effect on the access latency of individual cores to memory. Figure 1 shows the results of measuring the relative memory throughput on the Intel SCC in dependence on the number of hops between the issuing core and the accessed memory controller for various core operating frequencies. At the highest frequency of 800MHz, memory throughput drops by 33% when the core is located farthest away from the memory controller, demonstrating the necessity of NUMA-aware task placement.

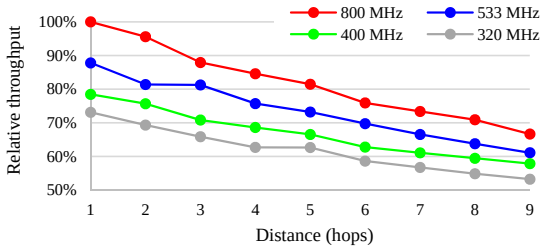

Fig. 1: Measured normalized memory throughput in dependence on the distance (hops) from the accessed memory controller for different core frequencies on the Intel SCC.

2.2 Power Management on CMPs

A vast body of research has shown DVFS to be an effective technique to limit power dissipation on core MVMF CMPs. The core observation behind DVFS is that the clock frequency of a core can be lowered without affecting performance when the core is idle or observes stalls caused by frequent memory accesses. Equipping every core with a voltage regulator, however, is becoming too costly [21], so domain-level multiple-voltage/multiple-frequency CMP designs are being proposed. In such domain MVMF designs, all cores in the same domain share the same voltage and frequency.

The NUMA properties of modern CMP architectures and the constraints with regards to power management require new approaches that combine task scheduling with domain MVMF-aware power management. From a power management perspective only, VMs with similar performance requirements should be scheduled together on cores in the same voltage and frequency domains in order to achieve optimal power savings. On the other hand, the NUMA characteristics of the chip require the scheduler to place VMs as close as possible to the accessed memory controllers. Furthermore, the scheduler and the power manager may need to adhere to user-defined performance goals such as minimizing power under constant performance, maintaining Quality of Service (QoS), even heat dissipation, or maximizing throughput for a given power budget. This paper introduces NuPow, a cooperative and hierarchical power management framework for domain MVMF CMPs that balances the independent goals of NUMA-aware scheduling and MVMF power management to achieve better energy efficiency.

2.3 Related Work

There exists a large body of work focusing on NUMA-aware work scheduling and the design and implementation of power management techniques for CMPs. One line of related work considers heterogeneous CMP. Kumar [22] proposes CMPs composed of cores different power characteristics. Ghiasi [12] proposes CMPs with cores executing at different frequencies. Both works show that such systems offer improved power consumption and thermal management. NuPow modifies the voltages and frequencies of the cores dynamically, without being bound to a specific hardware heterogeneity. Another line of research has focused on exploiting idle periods. Meisner proposes PowerNap [25] and DreamWeaver [26] that require hardware support for quick transitions between on- and off-states; the latter improves the former by batching wake-up events to increase the sleep periods. NuPow is orthogonal to such approaches.

Several power management techniques have been presented for existing core MVMF CMPs [7,8,15,16,17,19,23,24,27,31,34]. Li [23] analytically models to what extent parallel applications can be parallelized under a given power-budget. Isci [17] applies different DVFS policies under a power budget and shows that the best policy performs almost on par with an oracle policy. Meng [27] presents an adaptive power saving strategy that adheres to a chip-wide power budget by run-time adaptation of configurable processor cores. Rangan [31] proposes ThreadMotion, a technique for global shared-memory processors that moves threads between cores with different power and performance characteristics in order to improve power consumption. Hardware support is required to make thread migration more beneficial than DVFS. Cai [7] presents Thread Shuffling that migrates the hardware contexts of a single parallel application to exploit non-critical threads; non-critical threads can then be executed at reduced speed. Ma [24] proposes a solution aiming at a mixed group of single-threaded and multi-threaded applications and ignores NUMA properties of the CMP. Imamura [15] uses artificial neural networks to control power management and task placement, and Deng [8], finally, applies DVFS to multiple memory controllers. Their work is orthogonal to NuPow.

Techniques for domain MVMF CMPs typically propose hierarchical power management techniques [1,19,28]. Jha [19] classifies and migrates tasks based on DVFS sensitivity and cache behavior. That work and the approach of Yang [34], who focuses on multi-stage applications, aim at obtaining the best performance for a given power budget and ignore NUMA properties. Ali [1] exploits the low-power states of Intel processors to schedule virtual machines in a NUMA-aware manner. Unlike NuPow, these approaches assume global shared memory and a shared-memory kernel.

The works most closely related to NuPow are Ioannou [16] and Kang [20]. The former work applies DVFS to a static task assignment, the latter combines task migration with DVFS to obtain a significantly higher performance/watt ratio. Neither work considers the NUMA properties of CMPs. Both techniques are evaluated on the Intel SCC which allows for a direct comparison with NuPow on identical hardware.

3 Hierarchical Power Management

NuPow's cooperative hierarchical power manager combines VM relocation with DVFS to achieve optimal power efficiency for domain MVMF CMPs with NUMA properties.

3.1 VM Relocation

Virtual machines with similar performance characteristics need to be grouped together in frequency/voltage domains to allow for optimal DVFS settings and, as a consequence, improved power efficiency. If a VM with a high load is executed on a core co-located with lightly-loaded VMs in one domain, the entire domain needs to run at a higher power setting to satisfy the performance requirements of the busy VM. NuPow relocates VMs to group VMs with similar computational requirements such that individual domains can run at an optimal voltage/frequency setting. Taking data locality into consideration complicates the situation. Placing a VM away from the memory controllers where its data resides increases access latency and, in turn, requires a higher operation frequency to maintain throughput. In the opposite case, moving a VM closer to its data leads to lower access latency and may allow to run the core at a lower frequency.

3.2 Distributed Power Management

NuPow distributes the power management to a hierarchy of controllers that match the structure of the underlying MVMF architecture. Located at the lowest level are *core controllers* that manage a single core. Each frequency and voltage domain is controlled by a *frequency* or *voltage controller*, respectively, and a global *chip controller* sits on top of the hierarchy. The hierarchical structure and communication pattern improves the NuPow's scalability and is a natural fit to the task to be performed at each level.

- The **core controllers** monitor the performance characteristics of their core, predict performance requirements, and periodically report this data to the superordinate frequency controllers. Measured are IPC (Instructions Per Cycle) and the number of memory operations to determine the memory boundness of a VM. The required performance is extrapolated based on weighted collected data.
- The **frequency controllers** gather the performance data from the core controllers within their domain and forward the processed data to their superordinate voltage controllers. The frequency controllers also set the clock frequency of their domain.
- The **voltage controllers** collect, process, and forward data from the subordinate frequency controllers to the chip controller. The voltage controllers are also in charge of setting the operating voltage of their domains.
- The **chip controller** aggregates the data from the subordinate voltage controllers to compute an VM placement that considers the NUMA affinity of the VMs and allows for more optimal DVFS settings at the voltage and frequency domain levels.

4 DVFS and VM Relocation Policies

To allow cloud service providers to offer an undiminished quality of service while minimizing energy consumption, the focus of NuPow in this paper lies on optimizing the *performance per watt* ratio of the overall chip while maintaining throughput. The power management is implemented in the chip controller. DVFS and relocation algorithms are periodically invoked. Though the former depends on the latter, the DVFS and relocation policy are separated to support different combinations of relocation and DVFS policies.

4.1 DVFS Policies

NuPow supports all DVFS policies proposed for hierarchical power management by Ioannou [16] and Kang [20]. We compare NuPow against the two works using the `Tile` DVFS policy. `Tile` DVFS sets the voltage to the highest voltage requested by any of its frequency domains but allows each frequency domain to run at the optimal frequency that does not sacrifice performance.

4.2 Phase Ordering and Frequency Considerations

To achieve maximum power savings, VM relocation should occur before applying DVFS because a good placement of VMs allows for better voltage/frequency settings. The frequency of VM relocation and voltage/frequency changes depends on the cost of the individual operations. The total relocation time is not affected by the number of relocated VMs because the relocations occur in parallel. Voltage changes incur a not insignificant overhead because all cores in the affected domain are stopped during the voltage adjustment. Frequency changes, on the other hand, are almost instantaneous and can be performed more often.

Algorithm 1 Power-optimizing VM Placement

```
1: function COMPUTERELOCATIONMAP(Δm)
2:   vms ← set of all VMs with required voltage/frequency
3:   for each v ∈ {v_high, ··· , v_lowest} do
4:     vdom_v ← ⌈|{vm ∈ vms|vm.v = v}|/#vdom⌉
5:     for each vd ∈ vdom_v do
6:       vd.vms ← add VMs that require voltage v wrt to their data placement
7:       vd.loc ← compute location on chip minimizing distance to data of VM
8:       fdom ← compute required frequencies for domains in vd
9:       for each f ∈ {f_high, ··· , f_lowest} do
10:        fdom_f ← {x ∈ vdom|voltage(x) = v}
11:        for each fd ∈ vd do
12:          fd.vms ← add VMs that require frequency f wrt to their data placement
13:          fd.loc ← compute location within vd minimizing distance to data of VM
14:      assign unplaced VMs in descending order of required frequency to free cores in vd
15:  if ENERGY(new placement)·(1 + Δm) < ENERGY(current placement) then
16:    return new placement
```

4.3 Relocation of Virtual Machines

A naïve algorithm assigns the VM in order of their performance requirements to the domains. While the resulting placement is optimal in terms of power savings, it fails to consider the NUMA-properties of the NoC and the overhead of relocation. The relocation of a VM is very quick; measurements on a real system yield an overhead of $\leq 3ms$ [20]. Each time a VM is relocated to a different core, however, the VM will experience cold misses in the local caches that, in turn, lead to a loss of performance as well as increased memory traffic. A good algorithm has to balance the benefit of relocating a workload against the overhead incurred by relocation.

NuPow's relocation algorithm computes an optimized placement of VMs onto the cores that allows for an overall lower chip power consumption. Since we do not trade performance for power savings, the computational load of a VM determines the minimally required voltage and frequency of the core it is placed on. The goal of the algorithm is to minimize the number of domains that run at each domain/frequency and placing these domains on the chip in a NUMA-optimal way. Algorithm 1 shows the pseudo-code of the placement algorithm. The algorithm iterates over all available voltages and frequencies (lines 3, 9). For each voltage v, the VMs requiring that voltage are assigned to a voltage domain (line 6) and that domain is placed on a free domain such that the total distance of all VMs contained within to their data is minimized (line 7). The same step is then repeated within the voltage domain for all frequencies (lines 8–13). If there are free cores within the domain after this process, yet unplaced VMs are assigned in descending order of their required frequency (line 14). After all VMs have been placed, the energy consumption (Section 4.4) of the new placement is predicted and compared to that of the current placement. If the difference is larger than a given threshold Δm, the new placement is enacted (lines 15–16). The threshold Δm ensures that the expected energy savings are significant to avoid relocating VMs with little expected benefit.

4.4 Energy Model

The power consumption of CMOS logic comprises dynamic, short-circuit, and leakage power [29]. Voltage and frequency have a significant effect on dynamic power:

$$P \propto V^2 f \tag{1}$$

For a VM assignment with given voltage/frequency levels for the different domains, the energy consumption over the next epoch t can be approximated by

$$E_{status\_quo} = t \cdot \sum_{v \in vd} \sum_{f \in fd(v)} P(V_{current}(v), F_{current}(f)) \tag{2}$$

where $V_{current}(v)$ and $F_{current}(f)$ return the current voltage and frequency of a given voltage or frequency domain.

Relocating VMs may allow for better DVFS settings but incurs an overhead. The relocation overhead, O_{reloc}, is the overhead caused by the actual relocation and the (worst-case) time required to fill the empty cache on the newly assigned core. The memory overhead, O_{mem}, captures the sensitivity of a workload to the location of the assigned core on the CMP. The expected energy consumption to perform the same work after migration is then given by

$$E_{reloc} = P_{reloc} \cdot (t + O_{reloc} + O_{mem}) \tag{3}$$

$$P_{reloc} = \sum_{v \in vd} \sum_{f \in fd(v)} P(V_{relocated}(v), F_{relocated}(f)) \tag{4}$$

$$O_{reloc} = t_{relocation} + t_{cache\_fill}(F_{relocated}(f)) \tag{5}$$

$$O_{mem} = t \cdot \frac{throughput_{status\_quo}}{throughput_{relocated}} \tag{6}$$

where P_{reloc} is computed from offline power consumption data for each frequency level. The maximum throughput at each frequency and core location is profiled once offline; the actually required throughput of an application depends on the core's last-level cache misses and is measured by the core controllers.

5 Implementation

5.1 The Intel Single-chip Cloud Computer

NuPow is implemented and evaluated on the Intel Single-chip Cloud Computer (SCC). The Intel SCC consists of 48 independent cores interconnected by a routed NoC. No cache coherence is provided for the core-local L1 and L2 caches. Each pair of cores forms a *tile*; the 24 tiles are organized on a 6x4 grid. Four memory controllers in the four corners of the chip provide access to up to 64 GB of memory. An FPGA provides the interface between the CMP and the management PC (MCPC). Figure 2 shows a block diagram of the SCC.

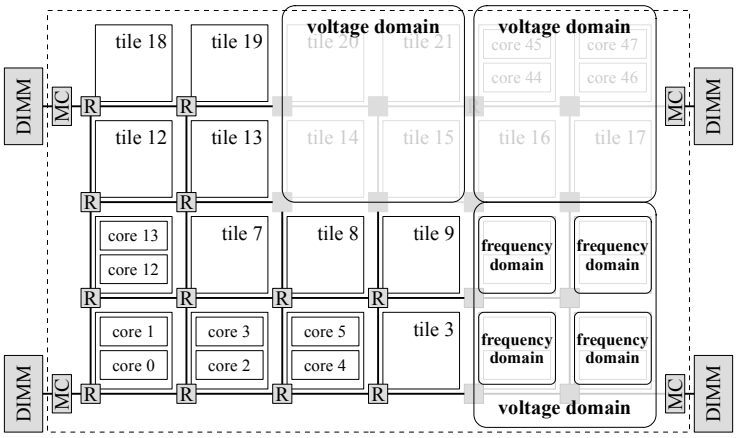

Fig. 2: Intel SCC block diagram

DVFS Capabilities. The SCC allows control over voltage and frequency for cores and the NoC. The right upper hand of Figure 2 illustrates frequency and voltage domains on the SCC. In total, there are six voltage domains comprising four frequency domains à two cores each. The SCC supports seven different supply voltage levels, however, only four are of practical interest: $1.1V$ to run at a frequency of 800MHz, $0.9V$ to run at 533MHz, $0.8V$ for 400MHz, and $0.7V$ for frequencies between 320 and 100MHz.

Power Measurement. The SCC provides a number of on-chip voltage and ampere meters. The total consumed power is computed by multiplying the supply voltage with the supply current for the entire SCC chip. Experimental results always report total chip power, i.e., include the overhead caused by the different domain controllers.

5.2 Virtual Machine Relocation

On a cache-coherent shared memory CMP, VMs can be migrated simply by pinning them onto a specific core. The Intel SCC does not support a cache-coherent shared memory space; instead all 48 cores are assigned separate memory spaces. Copying the volatile state of a VM to the memory of the designated core would incur a prohibitively large overhead; instead NuPow employs zero-copy migration by changing the core's memory mappings to point to the VM's data. This, in effect, relocates not just the VM but also the hypervisor running on the core. The interested reader is referred to Kang [20] for technical details on the relocation process.

5.3 Domain Controller Implementation

The domain controllers (core, frequency, voltage, and chip) are present in the system software running on core; the physical core ID determines which controllers are (de-)activated in a running kernel. After relocation, the kernels check if the core they are running on requires activation/deactivation of one of the four controllers. Core controllers are active on every kernel. The 24 frequency controller are activated on the cores with an odd core ID. The six voltage controllers run on the lower-left core of each domain. The chip controller is located on core 30.

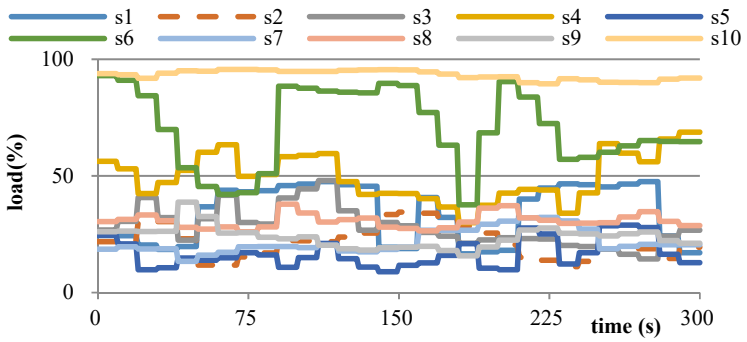

Fig. 3: Example of ten distinct workload patterns.

6 Experimental Setup

6.1 Hardware

All experiments are conducted on the Intel SCC [14]. Each core runs a modified version of the Intel SCC Linux. The on-chip voltage and ampere meters are queried with a frequency of 10'000Hz. All results report the power consumed by all 48 cores and the NoC. In particular, since NuPow is implemented entirely in software and is executed on the cores of the SCC, the result include the power consumed by NuPow.

6.2 Benchmarks

A benchmark scenario is defined by (1) a number of workload patterns and (2) a distribution of the workloads to cores.

(1) A **workload pattern** represents a load pattern experienced by a single VM and is composed of CPU load and memory load. Figure 3 shows an example of 10 different CPU workloads patterns `s1-s10`. The workload patterns of the datacenter scenarios are based on data gathered in Google data centers [33]. We have extracted the workloads of 100 randomly selected physical nodes in the system over a period of 3,000 seconds. Time is scaled by factor 10, i.e., a workload pattern runs for 300 seconds. This is not in our favor since rapidly changing load patterns put more stress on relocation. We also employ three synthetically generated workload patterns to demonstrate the potential of the presented approach.

(2) A **workload distribution** defines the initial placement of the VMs to the cores on the SCC. The data is always placed in the memory located closest to the initial placement. Depending on the scenario, between 2 to 40 different patterns are mapped onto the cores of the SCC.

Benchmark scenarios. We have generated 26 random scenarios from the Google cluster data. Each scenario comprises of w distinct workload patterns that are selected randomly from the 100 Google cluster data workload patterns. The initial placement of the VMs to physical cores can have a significant effect on the effectiveness of power management techniques that do not support workload migration (i.e., the DVFS only technique the proposed approach is compared against), so each reported result is the arithmetic average of three individual runs with three distinct initial random placements.

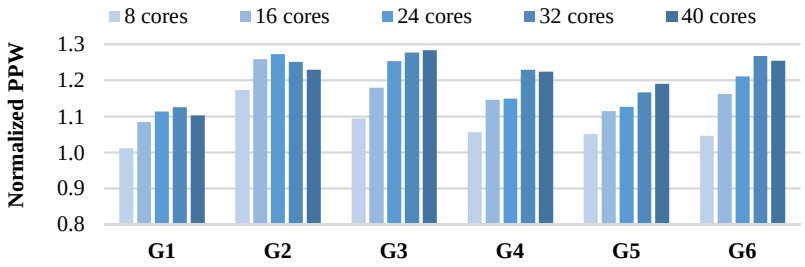

Fig. 4: Normalized performance per watt of `NuPow` relative to `DVFS only`.

6.3 Comparison of Results

The baseline of the experiments is obtained by running the benchmark scenario on the SCC at full speed (800MHz) with no power management. NuPow is compared against the `DVFS only` approach of Ioannou [16] and the DVFS+migration technique with its locality-unaware buyer-seller algorithm described by Kang [20] denoted `Buyer-Seller`. The hierarchical framework and the DVFS policies for all three methods are identical. We evaluated the different core migration algorithms using the `Tile` DVFS policies (Section 4.1). For all methods and benchmarks scenarios, the migration benefit threshold Δm is set to 10%. Migrations are evaluated and performed once every 3 seconds. All benchmark scenarios are executed to completion.

7 Results

7.1 Varying Number of Workloads

We first evaluate six distinct real-world datacenter scenarios `G1-G6` with respect to a varying number of assigned workloads from 8 to 40. The workload pattern `G1` contains four, `G2-G5` seven, and `G6` 10 distinct workload patterns that are randomly assigned to the number of workloads (i.e., for the 8-workload case and `G1` we make 8 random selections from the pool containing the four workload patterns). The initial location of the workloads on the chip can affect the result; we create three different random assignments and report the average of running each of the three assignments three times; i.e., each individual result represents the average of nine runs.

Figure 4 displays the results for the datacenter scenarios `G1` to `G6` with 8, 16, 24, 32, and 40 workloads running simultaneously. The y-axis shows the performance per watt of NuPow relative to `DVFS only`. We observe that `NuPow` shows better relative improvements if the number of active workloads (i.e., active cores) is between 16 and 32 cores. In the case of 8 workloads, `DVFS only` manages to do quite a good job (51% over no power management) despite its inability to migrate workloads because the low occupation still provides sufficient opportunities to apply DVFS. On the other end of the spectrum with 40 cores there are less opportunities for power savings with or without migration. The best case are moderately loaded CMPs where NuPow outperforms `DVFS only` by 25% on average.

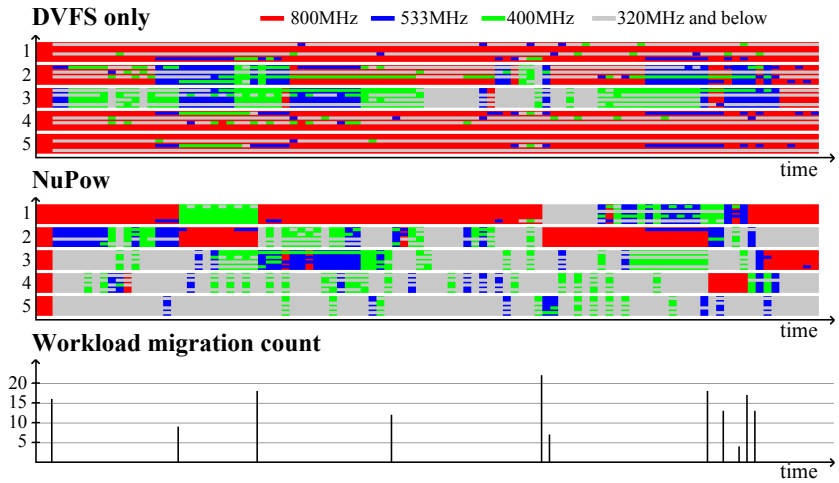

Fig. 5: Frequency map comparing `DVFS only` with `NuPow` for the G6 workload.

The effect of workload migration is visualized in Figure 5. The topmost graph shows the frequency map of `DVFS only` with the `Tile` policy for the different voltage domains. The middle graph shows the frequency map for the same workload with the proposed `Greedy` algorithm. While `DVFS only` is required to run most domains at a high frequency for most of the time, we observe that `NuPow` is able to group workloads with similar utilization into a few domains and apply aggressive DVFS on the lightly loaded domains. The bottom graph in Figure 5, finally, shows the number of workload migrations over time.

7.2 Independent Workloads

Figure 6 shows the normalized performance-per-watt over the baseline for 20 datacenter scenarios. Each scenario is composed of 40 independent randomly selected workloads patterns. NuPow's NUMA-aware algorithm outperforms `DVFS only` by 20 to 40 percent for each scenario, once again emphasizing the importance of workload migration for MVMF CMPs. The importance of NUMA-awareness is visible by comparing the NUMA-unaware `Buyer-Seller` to the proposed NUMA-aware `Greedy` algorithm: the latter achieves a between 7 and 12 percent better performance per watt with an average of a 10% better energy efficiency.

7.3 Evaluation of NUMA Affinity

Figure 7 compares the NUMA-unaware `Buyer-Seller` algorithm against `NuPow` in terms of the weighted distance of each workload's memory load to its memory controller. The data is show for the same 20 scenarios and is normalized to the best possible allocation considering *only memory affinity*. The whiskers show the standard deviation of the allocations over the entire run. We observe that the NUMA-aware `NuPow` algorithm places the workloads significantly closer to the accessed data than `Buyer-Seller`.

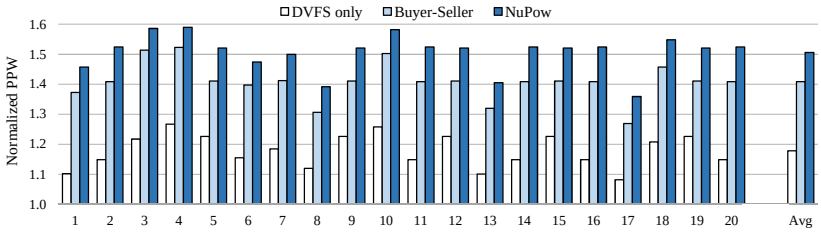

Fig. 6: Normalized performance per watt for 20 distince datacenter scenarios.

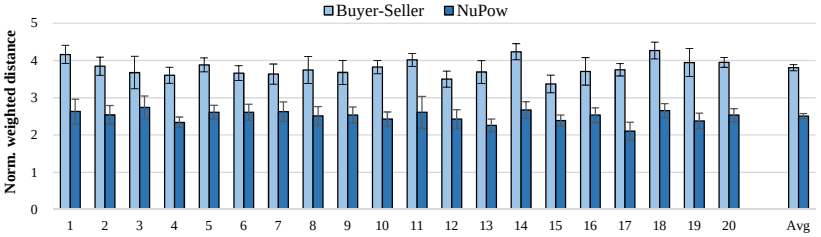

Fig. 7: Weighted memory distance comparing `Buyer-Seller` and `NuPow`.

8 Conclusion

We have presented NuPow, a NUMA-aware cooperative hierarchical power management technique for many-core systems with multiple-voltage/multiple-frequency islands. Combined with dynamic monitoring of each core's performance metrics, this technique allows the power manager to group virtual machines with similar performance requirements together so that traditional DVFS policies can apply DVFS settings closer to the optimal setting while at the same time locate memory-bound workloads closer to their data. In order to remain scalable, the power manager is implemented in a hierarchical fashion, logically re-creating the hierarchy imposed by the hardware through the different power management domains. NuPow has been implemented and evaluated on the Intel Single-Chip Cloud Computer. Experiments with a wide range of real world workload benchmark scenarios show that, on average, the proposed technique outperforms existing DVFS policies by 33% and by 10% compared to a NUMA-unaware approach.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) funded by the Korean government, in part, by grants NRF-2015K1A3A1A14021288, 2016R1-A2B4009193, by the BK21 Plus for Pioneers in Innovative Computing (Dept. of Computer Science and Engineering, SNU, grant 21A20151113068), and by the Promising-Pioneering Researcher Program of Seoul National University in 2015. ICT at Seoul National University provided research facilities for this study.

References

1. Ali, Q., Zheng, H., Mann, T., Srinivasan, R.: Power aware numa scheduler in vmware's esxi hypervisor. In: Proceedings of the 2015 IEEE International Symposium on Workload Characterization. IISWC '15, Washington, DC, USA (2015)
2. Bohnenstiehl, B., Stillmaker, A., Pimentel, J.J., Andreas, T., Liu, B., Tran, A.T., Adeagbo, E., Baas, B.M.: KiloCore: A 32-nm 1000-Processor Computational Array. IEEE Journal of Solid-State Circuits **PP**(99) (2017)
3. Borkar, S.: Thousand core chips: A technology perspective. In: Proceedings of the 44th Annual Design Automation Conference. DAC '07 (2007)
4. Borkar, S., Chien, A.A.: The future of microprocessors. CACM **54**(5) (2011)
5. Burd, T.D., Brodersen, R.W.: Energy efficient cmos microprocessor design. In: Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences. vol. 1 (1995)
6. Butts, M.: Synchronization through communication in a massively parallel processor array. IEEE Micro **27**(5) (2007)
7. Cai, Q., González, J., Magklis, G., Chaparro, P., González, A.: Thread shuffling: Combining DVFS and thread migration to reduce energy consumptions for multi-core systems. In: Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design. ISLPED '11 (2011)
8. Deng, Q., Meisner, D., Bhattacharjee, A., Wenisch, T.F., Bianchini, R.: Multiscale: Memory system dvfs with multiple memory controllers. In: Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design. ISLPED '12 (2012)
9. Dighe, S., Vangal, S.R., Aseron, P., Kumar, S., Jacob, T., Bowman, K.A., Howard, J., Tschanz, J., Erraguntla, V., Borkar, N., De, V.K., Borkar, S.: Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core teraflops processor. Solid-State Circuits, IEEE Journal of **46**(1) (2011)
10. Duran, A., Klemm, M.: The intel many integrated core architecture. In: High Performance Computing and Simulation (HPCS), 2012 International Conference on (2012)
11. Fu, X., Wang, X.: Utilization-controlled task consolidation for power optimization in multi-core real-time systems. In: 2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications. vol. 1 (2011)
12. Ghiasi, S.: Aide De Camp: Asymmetric Multi-core Design for Dynamic Thermal Management. Ph.D. thesis, Boulder, CO, USA (2004), aAI3136618
13. Herbert, S., Marculescu, D.: Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In: 2007 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED) (2007)
14. Howard, J., Dighe, S., Hoskote, Y., Vangal, S., Finan, D., Ruhl, G., Jenkins, D., Wilson, H., Borkar, N., Schrom, G., Pailet, F., Jain, S., Jacob, T., Yada, S., Marella, S., Salihundam, P., Erraguntla, V., Konow, M., Riepen, M., Droege, G., Lindemann, J., Gries, M., Apel, T., Henriss, K., Lund-Larsen, T., Steibl, S., Borkar, S., De, V., Van der Wijngaart, R., Mattson, T.: A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In: Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International (2010)
15. Imamura, S., Sasaki, H., Inoue, K., Nikolopoulos, D.S.: Power-capped dvfs and thread allocation with ann models on modern numa systems. In: 2014 IEEE 32nd International Conference on Computer Design (ICCD) (2014)
16. Ioannou, N., Kauschke, M., Gries, M., Cintra, M.: Phase-based application-driven hierarchical power management on the single-chip cloud computer. In: Proceedings of the 2011 Int'l Conference on Parallel Architectures and Compilation Techniques. PACT '11 (2011)
17. Isci, C., Buyuktosunoglu, A., Cher, C.Y., Bose, P., Martonosi, M.: An analysis of efficient multi-core global power management policies: Maximizing performance for a given power

budget. In: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. MICRO 39 (2006)
18. Jain, V.: Fast Process Migration on Intel SCC using Lookup Tables (LUTs). Tech. Rep. Masters Thesis, Arizona State University (May 2013)
19. Jha, S.S., Heirman, W., Falcón, A., Tubella, J., González, A., Eeckhout, L.: Shared resource aware scheduling on power-constrained tiled many-core processors. Journal of Parallel and Distributed Computing **100** (2017)
20. Kang, C., Lee, S., Lee, Y.J., Lee, J., Egger, B.: Scheduling for better energy efficiency on many-core chips. Job Scheduling Strategies for Parallel Processing: 19th and 20th International Workshops, JSSPP 2015, Hyderabad, India, May 26, 2015 and JSSPP 2016, Chicago, IL, USA, May 27, 2016, Revised Selected Papers (2017)
21. Kim, W., Gupta, M.S., Wei, G.Y., Brooks, D.: System level analysis of fast, per-core dvfs using on-chip switching regulators. In: IEEE 14th International Symposium on High Performance Computer Architecture (HPCA 2008) (2008)
22. Kumar, R., Tullsen, D.M., Ranganathan, P., Jouppi, N.P., Farkas, K.I.: Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In: Proceedings of the 31st Annual International Symposium on Computer Architecture. ISCA '04 (2004)
23. Li, J., Martínez, J.F.: Power-performance implications of thread-level parallelism on chip multiprocessors. In: IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2005) (2005)
24. Ma, K., Li, X., Chen, M., Wang, X.: Scalable power control for many-core architectures running multi-threaded applications. In: Proceedings of the 38th Annual International Symposium on Computer Architecture. ISCA '11 (2011)
25. Meisner, D., Gold, B.T., Wenisch, T.F.: Powernap: Eliminating server idle power. In: Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS XIV (2009)
26. Meisner, D., Wenisch, T.F.: Dreamweaver: Architectural support for deep sleep. In: Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS XVII (2012)
27. Meng, K., Joseph, R., Dick, R.P., Shang, L.: Multi-optimization power management for chip multiprocessors. In: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. PACT '08 (2008)
28. Mishra, A.K., Srikantaiah, S., Kandemir, M., Das, C.R.: Cpm in cmps: Coordinated power management in chip-multiprocessors. In: Proceedings of the 2010 ACM/IEEE Int'l Conference for High Performance Computing, Networking, Storage and Analysis. SC '10 (2010)
29. Mudge, T.: Power: A first-class architectural design constraint. Computer **34**(4) (2001)
30. Olofsson, A.: Epiphany-V: A 1024 processor 64-bit RISC System-On-Chip. `https://arxiv.org/abs/1610.01832` (2016), online, accessed July 2020
31. Rangan, K.K., Wei, G.Y., Brooks, D.: Thread motion: Fine-grained power management for multi-core systems. In: Proceedings of the 36th Annual International Symposium on Computer Architecture. ISCA '09 (2009)
32. Rotem, E., Mendelson, A., Ginosar, R., Weiser, U.: Multiple clock and voltage domains for chip multi processors. In: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO 42 (2009)
33. Wilkes, J.: More Google cluster data. Google Research Blog at `http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html` (2011), online, accessed July 2020
34. Yang, H., Chen, Q., Riaz, M., Luan, Z., Tang, L., Mars, J.: Powerchief: Intelligent power allocation for multi-stage applications to improve responsiveness on power constrained cmp. In: Proc. of the 44th Annual Int'l Symposium on Computer Architecture. ISCA '17 (2017)