EPFL-REPORT-175359

Scale-Out Processors

Pejman Lotfi-Kamran[†], Boris Grot[†], Michael Ferdman[‡†], Stavros Volos[†], Onur Kocberber[†], Javier Picorel[†], Almutaz Adileh[†], Djordje Jevdjic[†], Sachin Idgunji[*], Emre Ozer[*], Babak Falsafi[†]

[†]EcoCloud
École Polytechnique Fédérale de Lausanne

[‡]CALCM
Carnegie Mellon University

[*]ARM

ABSTRACT

The emergence of global-scale online services has galvanized scale-out software, characterized by splitting vast datasets and massive computation across many independent servers. Datacenters housing thousands of servers are designed to support scale-out workloads, with per-server throughput dictating the overall datacenter capacity and cost. However, today's processors do not use the die area efficiently, limiting the per-server throughput. We find that existing processors over-provision cache capacity, leading to designs with sub-optimal performance density (performance per unit area). Furthermore, as these designs are scaled up with technology, the increasing number of cores leads to further performance density reduction due to increased on-chip latencies.

We use a suite of real-world scale-out workloads to investigate performance density and formulate a methodology to design optimally-efficient processors for scale-out workloads. Our proposed architecture is based on the notion of pods, an optimal performance-density building block that tightly couples several cores with a small last-level cache using a fast interconnect. Replicating the optimal performance-density pod to fill the die area necessarily yields processors which themselves have optimal performance density, leading to maximum per-server throughput. Moreover, replicating the pod design to fill the additional die area in future technology generations overcomes the performance penalties of traditional design scale-up. We use a combination of model-driven analysis and cycle-accurate simulation to demonstrate our design methodology, which, in 40nm technology, achieves 6x throughput improvement over conventional datacenter processors and 13% over tiled processors. When scaled to 20nm technology, throughput gains increase to 42% over a tiled processor configuration.

1. Introduction

Cloud computing has emerged as the foundation for scalable online services. Cloud operators such as Google, Microsoft, and Facebook rely on networks of data centers to deliver search, social connectivity, and a growing number of other offerings. Underlying the cloud's ability to adapt to a growing dataset and user base, the scale-out software architecture allows to accommodate load increases simply by adding more servers to the cloud, because servers handle independent requests that do not share any state. With typical scale-out applications operating on peta-scale datasets distributed across tens of thousands of servers, at the datacenter level, per-server throughput dictates the datacenter capacity and cost.

Today's volume servers comprise two-socket systems populated with processors designed for the general purpose market. These conventional processors combine a handful of aggressively speculative and high clock-frequency cores, supplemented by a large shared on-chip cache, a culmination of more than 30 years of research and development targeting the absolute peak single-thread performance. Peak single-thread performance minimizes the time spent on processing each request, enabling high throughput. As manufacturing technology provides higher transistor density, conventional processors use the additional transistors to increase on-chip core count and cache capacity, allowing new processors to achieve higher throughput.

Recently, tiled processors have emerged as competition to volume processors in the scale-out server space [7, 46]. Recognizing the importance of per-server throughput, these processors use the same die area as the conventional designs, but implement a large number of *tiles* interconnected by an on-chip network, each tile comprising a simple core and a piece of the aggregate cache. The single-thread performance of each tile

is lower than the performance of a conventional core, but the increased parallelism from a larger number of on-chip cores yields higher per-server throughput on scale-out applications. Although radically different in the on-chip organization, the technology scaling trends of tiled processors are similar to conventional processors; each technology generation affords more tiles, which increases the core count and cache capacity.
We observe that, in the context of processors for scale-out applications, both architectures make sub-optimal use of the on-chip area. Conventional and tiled processors organizations dedicate a large fraction of the die area to the shared last-level cache. Although large caches can improve single-thread performance, scale-out workloads observe limited benefit due to their enormous data footprints, leading to a marginal improvement in throughput. Maximizing throughput necessitates a careful choice in the size of the cache, reducing cache capacity while maximizing the die area used for cores and parallelism. Moreover, the latency incurred by the on-chip interconnect limits the benefits of integration in existing processors, slowing the gains in throughput despite increasing core counts.
In this work, we seek to develop a technology-scalable server chip architecture for scale-out workloads that makes optimal use of the on-chip real-estate. We use performance density, defined as throughput per unit area, to quantify how effectively an architecture uses the die real-estate, enabling us to design a building block called a *pod*, which has an optimal performance density with respect to the core-to-cache area ratio and the interconnect latency. As technology scales to allow more on-chip cores, our methodology calls for keeping the design of the pod unchanged, replicating the pod to fill the available die area. The pod methodology avoids the challenges of scaling hardware to large core counts, while at the same time guaranteeing maximum throughput and optimally-efficient use of the on-chip real-estate.
We use trace-driven and cycle-accurate full-system simulation of a diverse suite of representative scale-out workloads to demonstrate that:

- the core-to-cache area ratio of existing processors results in poor processor performance density;
- scaling up of existing designs to larger core counts leads to a reduction in performance density due to increases in on-chip communication latency;
- performance density can be used to derive an optimally-efficient pod that uses a small (i.e., 2-4MB) last-level cache and benefits from a high core-to-cache area ratio and simple crossbar interconnect;
- replicating pods to fill the die results maintains performance density as core counts increase, resulting in optimal use of die real-estate and maximizing the per-server throughput.

2. Why Scale-Out Processors?

In this section, we make the case for scale-out processors by first examining the characteristics of scale-out workloads, followed by an analysis of both conventional and emerging server processor architectures. Using performance density as a metric of interest, we demonstrate that today's processors are poorly matched to the demands of scale-out applications as evidenced by a significant gap in performance density between existing designs and optimal configurations.

2.1 What do the workloads want?

We examine a representative slice of scale-out applications in order to understand the demands that they place on server chip architectures. Our workloads include search, map/reduce, web frontend, and data serving, which are the dominant applications in scale-out datacenters, such as those operated by Google and Microsoft. We also consider a simulation workload, representative of a computationally-demanding scale-out service. Overall, we find that the scale-out applications in our suite have functionally similar characteristics, namely (a) they operate on huge data sets that are split across a large number of nodes into memory-resident shards; (b) the nodes service a large numbers of completely independent requests that do not share any state; and (c) the inter-node connectivity is used only for high-level task management and coordination.

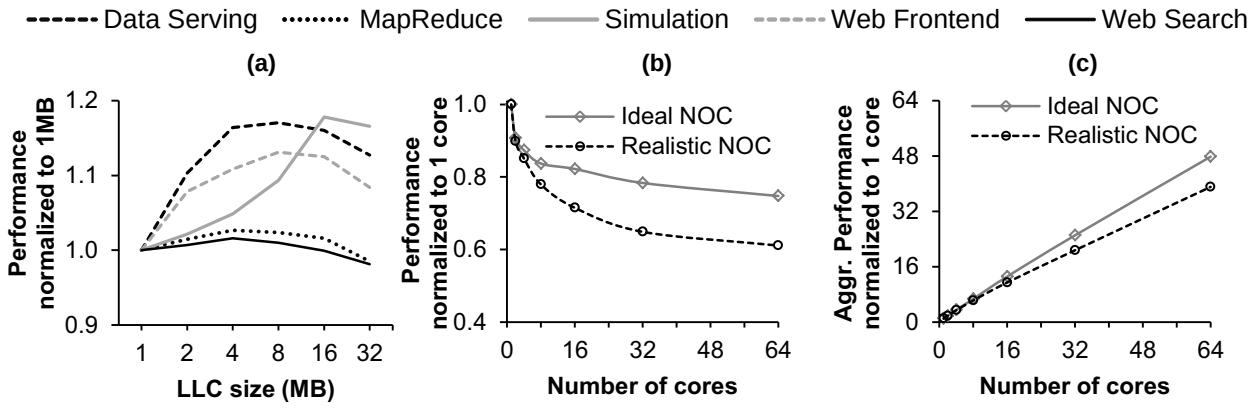

Figure 1. Performance of 4-core workloads varying the LLC size (a), average performance across the workloads with a 4MB LLC varying the number of cores (b), and aggregated performance with a 4MB LLC varying the number of cores (c).

2.1.1 Sensitivity to LLC Size

We first analyze the cache requirements of scale-out applications by sweeping the size of the last-level cache (LLC) from 1 to 32 MB. We present the results for a quad-core CMP, but note that the general trends are independent of the core count. Details of the methodology can be found in Section 4.2.

Figure 1(a) plots the performance of each cache configuration normalized to a design with a 1 MB LLC. For the four traditional datacenter workloads, we observe that LLC capacities of 2-8 MB are sufficient to capture the secondary working sets, and that cache configurations beyond this range are not beneficial due to the enormous memory footprints of the applications. The Simulation workload exhibits a different cache behavior, as larger cache capacities aid in capturing the secondary working set. However, even in the case of Simulation, a 16-fold increase in cache size from 1 to 16 MB translates into a performance gain of just 18%. Beyond 16 MB, additional cache capacity is strictly detrimental to performance, as the reduction in miss rate it provides is dwarfed by the increased access latency. Hardavellas et al. observed similar trends in the cache behavior of database workloads [20].

2.1.2 Sensitivity to core count

Next, we analyze the sensitivity of scale-out applications to the number of threads and the sharing degree. To do so, we fix the LLC size at 4 MB and examine the performance as the number of cores varies from 1 to 64. Figure 1(b) plot the performance per core and collective throughput, respectively, averaged across workloads and normalized to a single-core baseline. The two lines in the figures correspond to (a) an ideal organization with a fixed-latency interconnect between each core and the LLC, and (b) a realistic mesh-based interconnect where the physical distance between cores and cache banks affects the LLC access latency.

In the case of an ideal interconnect, we observe a one-time performance drop of around 10% going from a single- to dual-core configuration in Figure 1(b). The degradation in per-core performance associated with adding more cores is small – 17% for a 32x increase in core count (i.e., from 2 to 64 cores). As a result, Figure 1(c) demonstrates that aggregate performance can be improved by a factor of 48 by sharing a fixed-size cache among 64 cores.

In the case of a design subject to physical constraints, the negative slope of the performance curve in Figure 1(b) is much steeper. The distance to the LLC has a direct effect on performance due to a combination of (a) primary working set sizes that greatly exceed L1 capacities, and (b) the memory-intensive nature of scale-out applications, which makes them particularly sensitive to the average memory access time (AMAT). Figure 1(c) shows a 19% gap in throughput between the ideal and realistic designs at 64 cores,

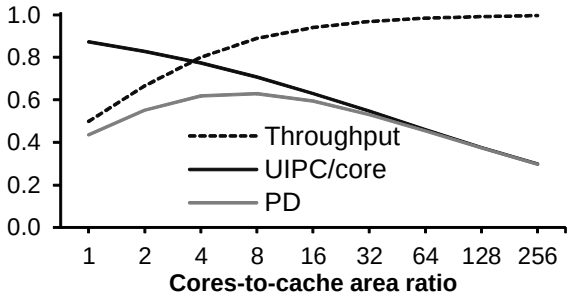

Figure 2. Throughput, UIPC per core, and PD metrics for a hypothetical workload.

underscoring how distance effects threaten the ability of server processors to reach their throughput potential.

2.2 Performance Density: A metric of architectural efficiency for Scale-Out workloads

Scale-out applications are best served by substrates that strike a delicate balance between the conflicting objectives of latency and throughput. On one hand, achieving high per-server throughput is a critical means of maximizing the hardware investment, which is a dominant expense category in today's datacenters [31]. On the other hand, latency-sensitive workloads such as Web Frontend and Web Search necessitate predictable single-threaded performance. While the former requirement calls for high core density, the latter demands greater cache capacities and small interconnect delays.

To capture these requirements into a simple metric for assessing the efficiency of datacenter server processors, we propose performance density (PD), defined as UIPC/mm^2. Given a core microarchitecture, the performance density metric provides a simple means of comparing a range of designs that differ in the number of cores, LLC size, and interconnect characteristics.

Figure 2 provides the intuition behind the metric using a hypothetical workload whose behavior is representative of scale-out applications. The x-axis plots the core-to-cache area ratio for a fixed cache size. Thus, the number of cores increases to the right of the graph, resulting in a higher core-to-cache ratio. The black solid line plots the per-core performance, which diminishes as the number of cores grows due to the combination of distance and sharing at the LLC. The dotted line shows the aggregate throughput, which scales with the additional core resources, but the growth is sub-linear in core count due to the eroding per-core IPC. Finally, the gray line plots performance density, whose peak represents an optimal configuration that maximizes performance per unit area by balancing throughput, sharing, and distance factors. As we will show in Section 5, designs based on existing core, cache, and interconnect components exhibit similar behavior to the one shown in Figure 2.

Depending on core microarchitecture, interconnect parameters, and other factors, a performance-density optimal (PD-optimal) design may consume only a fraction of the available on-die area and power budget. Simply increasing core and/or cache resources to boost aggregate throughput up to the limits dictated by the package (i.e., area, power, or pins) will necessarily diminish performance density. Instead, a scaling strategy that preserves performance density optimality while scaling aggregate throughput requires tiling at the granularity of a PD-optimal configuration.

2.3 What are the existing server architectures providing?

Figure 3 compares the relative efficiency of several server chip architectures assuming a constant 300 mm^2 area for cores and caches. For each design, the three bars show total core area, cache (LLC) area, and average performance density across the workload suite. While the aggregate performance (throughput) of each configuration is not shown, it is directly proportional to performance density due to the constant area across the designs. The number above each group of bars is the core count of the corresponding configuration.

We first examine a conventional processor architecture, representative of designs used in today's datacenters. We model an organization based on a contemporary Intel Xeon Westmere-EX-class processor, which

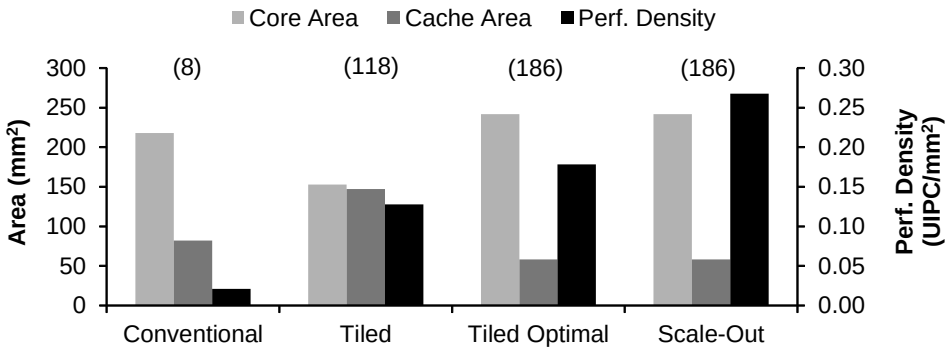

Figure 3. Performance density comparison of various designs. The number above each design corresponds to the number of cores integrated in the design.

integrates up to 10 aggressive out-of-order cores with a two-level private cache hierarchy and up to 30 MB of last-level cache. The conventional server chip organization achieves low aggregate performance and performance density due to the misappropriated core and cache budget. High core complexity carries a significant area cost, yet does not translate into a commensurate performance gain. The LLC is also over-provisioned for the few cores and modest demands placed by the secondary working sets of the scale-out applications.

As an alternative to conventional "fat-core" server chips, several start-ups have announced products that integrated dozens of light-weight cores on a die. One such emerging server processor is Tilera GX 3036, a heavily tiled design with a mesh-based NOC [46]. Each tile contains a 3-wide VLIW core, coupled with a 256 KB slice of a coherent NUCA LLC and a router. The Tiled design in Figure 3, modeled after the 3036, achieves higher performance per unit area as compared to the Conventional architecture due to the vastly expanded execution resources. However, the large LLC is grossly over-provisioned, thereby reducing the area available for the cores and limiting throughput as a result. We introduce a configuration labeled Tiled Optimal that addresses the cache inefficiency by allocating the core and cache area in an optimal manner, boosting throughput and performance density by 40% over the baseline Tiled design. With a proper core-to-cache area ratio, the chief performance-limiting factor in the Tiled Optimal chip are the delays introduced by the multi-hop interconnect that depress the performance of individual cores.

Finally, the Scale-Out configuration, based on the methodology proposed in this work, preserves the core-to-cache ratio of the Tiled Optimal design while boosting throughput by another 50% through significantly lower interconnect delays. The Scale-Out design accomplishes this through a partitioning strategy that relies on independent PD-optimal units that use a fast local interconnect to minimize the communication delays between the cores and the local LLC. In the next section, we detail the Scale-Out chip architecture and explain how such designs can be built in technology-constrained chips of today and tomorrow.

3. Scale-Out Processors

Today's server chips, such as the conventional Xeon processor and the emerging tiled designs, scale performance through the addition of cores, cache capacity, interconnect and coherence resources, and miscellaneous glue logic. This scaling strategy is a characteristic of the *scale-up* model. Such a model of increasing processor complexity is counter-productive for scale-out workloads, as additional resources do not yield a commensurate improvement in chip-level performance.

In this work, we seek to develop a technology-scalable approach for maximizing the performance of server processors targeting scale-out workloads. Our approach uses performance density as an optimization metric and builds on a simple observation that, given a configuration that is PD-optimal, the most profitable way of scaling aggregate performance is to grow the number of such PD-optimal units. Such a strategy maintains the optimal performance density while increasing aggregate throughput. In contrast, an approach that expands a PD-optimal configuration through additional core and cache resources lowers performance density and leads to a chip organization whose peak throughput is sub-optimal for a given area budget.

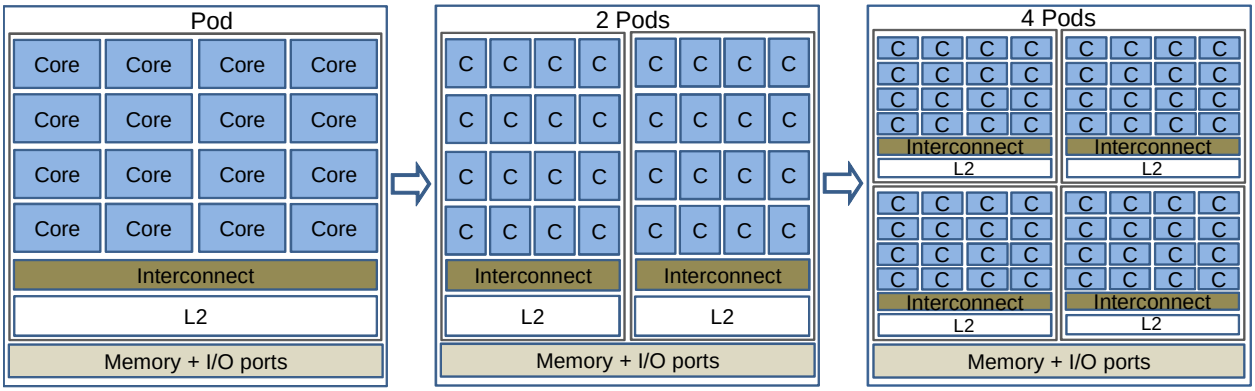

Figure 4. The Scale-Out Processor design replicates the optimal-PD pod to fully utilize the chip resources.

We define a *pod* as a PD-optimal organization of core, cache, and interconnect resources. *Podding* refers to the optimal scaling strategy based on tiling at the granularity of pods. Figure 4 captures the spirit of our approach by demonstrating how podding can be used to scale the on-die execution resources while preserving the organization of each PD-optimal pod.

3.1 Pod organization

A pod couples an optimally-ratioed set of core and cache resources via an interconnect and coherence layer. The core-to-cache area ratio and the actual performance density heavily depend on the choice of components used in designing the pod. The core microarchitecture, cache parameters, and interconnect characteristics all play a vital role in determining the PD-optimal organization. For instance, the large area of a conventional server core makes a 1MB cache bank appear relatively inexpensive. In contrast, multiple in-order cores can be squeezed into the footprint of a 1MB cache, greatly inflating the relative cost of the LLC. Our evaluation of two different core microarchitectures in Section 5 clearly shows the fact that differences in core complexity lead to different pod designs.

In our experience, several organizations that differ considerably in core count or cache size may have near-optimal performance density. In many cases, smaller pod configurations (i.e., ones that features fewer components) may be preferred due to the limits on software scalability, reduced interconnect and coherence complexity, and lower susceptibility to faults at the chip level.

Another consideration in pod design is the trade-off between cache capacity and memory bandwidth. Because caches filter accesses that would otherwise go to main memory, DRAM bandwidth demands are a function of the cache capacity. Today's on-die memory interfaces are extremely area-intensive, and the ability to jointly optimize cache and DRAM requirements is a useful capability for designing PD-optimal servers. Our results for performance density in Section 5 capture this trade-off between cache capacity and the required number of memory interfaces.

3.2 Chip organization

A Scale-Out chip is a simple composition of one or more pods and a set of memory and I/O interfaces. The number of pods on a die is dictated by physical constraints, such as area, power, and pin bandwidth. Because scale-out chips maximize performance per unit area, they are more cost-effective (i.e., require less silicon area for a given throughput target) than non-optimal designs. Furthermore, for a given core and cache budget, a scale-out architecture is more energy-efficient than a non-optimal one due to high communication locality inside each pod, which helps reduce the energy footprint of the interconnect.

Each pod is a stand-alone server that runs its own copy of the operating system. The lack of inter-dependence among pods is a valuable feature that eliminates the need for a pod-to-pod communication infrastructure and chip-wide coherence support. The absence of these mechanisms reduces the design complexity of scale-out processors and boosts their scalability.

Table 1. Area and power estimates for various system components.

Component		Area	Power
Cores	Cortex-A15	4.5mm^2	1W
	Cortex-A8	1.3mm^2	0.48W
LLC	16-way SA	5 mm^2 per MB	1W per MB

Component	Area	Power
Interconnect	0.1 - 4.5 mm^2	< 1.5W
Memory controller and single channel	(2 + 10) mm^2	5.7W
SoC components	15% of total	5% of total

We anticipate that pods will share the on-die DRAM and I/O ports to improve bandwidth utilization and reduce pin pressure. Scale-out chips that share pins among pods will necessitate a global interconnect layer to connect the individual pods to the memory and I/O ports. Fortunately, such a layer can be kept trivially simple due the limited connectivity that it must provide, since pod-to-pod connections are not needed and the number of external interfaces is typically very small.

4. Methodology

We conduct our evaluation through a combination of cycle-accurate simulation and validated CMP models [17]. We consider two types of cores: high-performance three-way out-of-order cores — similar to ARM Cortex-A15 [43] — and dual-issue in-order cores (resembling ARM Cortex-A8 [3]). Cores operate at 2 GHz, 0.9V in 40nm technology. Table 1 illustrates the area and power estimates for all the CMP components.[1]

We model a range last-level cache sizes between 1 and 8 MB. We consider three interconnect types between cores and the last-level cache: (a) an ideal crossbar in which each core accesses a cache bank with a fixed latency; (b) a realistic crossbar in which the access latency of a cache bank is a function of the physical distance between the core and the cache bank; and (c) a mesh interconnect with a delay of 3 cycles per hop (i.e., link and router). We use analytic models based on ITRS projections for interconnect area and power estimation, and CACTI 6.5 [33] for cache modeling.

For the memory interface and other SoC components, we estimate the area by scaling the micrograph of a Nehalem processor in 45nm technology [28]. We measure the power consumption of a single DDR3 memory channel operating at 1.6GHz[2] by taking actual measurements on a real system.[3] We find the power consumption of SoC components to be 5% of the total chip power using McPAT v0.8 [30] to model Sun UltraSparc T2.

To understand the effect of technology scaling on the *Scale-out Processor* design, we also model our systems in 20nm technology. We scale the area of cores, caches, and interconnects from 40nm technology by a factor of 0.25 [44]. We find that the analog circuitry in the PHYs of memory interfaces prevents them from benefitting from technology scaling [12, 15, 28, 29] and as a result memory channels scale only by a small factor across technologies. Supply voltage drops by only 0.1V to 0.8V, from 40nm to 20nm node.

For scaling the available off-chip bandwidth, we anticipate that DDR4 memory will replace the current DDR3 standard by the 20nm technology node. Based on the JEDEC specification [23], we assume that a single DDR4 memory channel will operate at 3.2 GHz.

4.1 Scale-out applications

To find the set of applications that run in today's data centers, we examined a selection of internet services based on their popularity [1]. We present a brief overview of the applications commonly found in today's datacenters:

Data Serving: Cloud operators rely on *NoSQL* data stores for fast and scalable storage with varying and rapidly evolving storage schema. Facebook uses a NoSQL data store for their inbox service, while Google

1. For the area and power of Cortex-A8, we double the area and power of a single-issue in-order Cortex-A5 core [16] to account for twice the cache size, issue width, and the existence of a floating-point unit.
2. A single DDR3 memory channel operating at 1.6GHz provides off-chip bandwidth of 12.8GB/s
3. We find the energy per data pin to be 80pJ. We assume an effective memory bandwidth utilization of 70% [10].

Table 2. System parameters for cycle-accurate full-system simulations.

CMP Size	1-16 cores for Web Search and Web Frontend 1-32 cores for Data Serving 1-64 cores for MapReduce, and Simulation
Processing Cores	UltraSPARC III ISA; 2GHz 8-stage pipeline 3-wide dispatch / retirement, 16-entry store buffer, 60-entry ROB, LSQ
L1I / D Caches	32KB 2-way, 2 cycle load-to-use, 2 ports, 32 MSHRs, 8-entry victim cache
Last-level Cache	1/2/4/8 MB,16-way set-associative Cache bank access latency: 3 cycles for 64/128KB, 4 cycles for 256/512KB, and 5/6/8/10 cycles for 1/2/4/8 MB 64-byte lines, 1 port, 64 MSHRs, 16-entry victim cache 1 bank per every 4 cores for UCA and 1 bank per LLC slice for NUCA
Interconnect	*Ideal crossbar*: 2 cycles for 2-, 4-core CMPs and 4 cycles for the rest. *Crossbar*: For CMPs up to 8 cores the same as in Ideal. For 16-, 32-, and 64-core CMPs we assume 5, 7, and 11 cycles respectively. *Mesh*: 3 cycles/hop
Main Memory	4GB for 1-, 2-, and 4-core CMPs and 2GB per core for 8-, 16-, 32-, 64-core CMPs 45ns access latency

employs it for Google Earth and Google Finance applications. The NoSQL systems split hundreds of terabytes of data into shards and horizontally scale to large cluster sizes, typically using indexes that support fast lookup and key range scans to retrieve the set of requested objects. We benchmark one node running the Cassandra 0.7.3 NoSQL storage system [41] with a YCSB dataset that exceeds the memory capacity to mimic a realistic setup [9]. Server load is generated using the YCSB 0.1.3 client [9] that sends requests following a Zipfian distribution with an equal number of reads and updates.

MapReduce: The map-reduce paradigm has emerged as a popular approach to handling large-scale analysis, farming out requests to a cluster of nodes that first perform filtering and transformation of the data (map) and then aggregate the results (reduce). We benchmark one node running the WordCount workload on a 30GB set of Wikipedia pages on top of a Hadoop 0.20.2 cluster. Each core runs one map and one reduce job.

Simulation: The ability to allocate compute resources on-demand using the cloud has created an opportunity to conduct large-scale simulations without a significant infrastructure investment. We benchmark one node running parallel symbolic execution [6] to search for programming bugs in application binaries from the GNU CoreUtils 6.10.

Web Frontend: Businesses are increasing migrating web services with both static and dynamic content into the cloud for improved scalability and resilience. In a typical system, independent client requests are accepted by a stateless web server process which either directly serves static files from disk or passes the request to a stateless middleware script which is then responsible for producing dynamic content. We benchmark one node running the Nginx 1.0.1 [34] web server with an external PHP 5.2.6 module and APC 3.0.19 PHP opcode cache serving the SPECweb2009 e-banking workload [40].

Web Search: The search workload is highly demanding and characterized by rapid load fluctuations with strict latency requirements on each request. The search service relies on a multi-terabyte index, split into shards for scalability. We benchmark one index serving node (ISN) [38] which runs the distributed version of Nutch 1.1/Lucene 3.0.1 [42]. We make sure that the search index fits in memory to eliminate page faults and minimize disk activity to mimic real-world setups [38]. We simulate the clients using the Faban workload driver. The clients are configured to achieve the maximum search request rate while ensuring that 90% of all search queries complete in less than 0.5 seconds.

4.2 Cycle-accurate simulation

We use *Flexus* [45] for cycle-accurate full-system simulation of various CMP configurations. *Flexus* extends the *Virtutech Simics* functional simulator with timing models of processing tiles with in-order and out-of-order cores, caches, on-chip protocol controllers, and an on-chip interconnect. We model CMPs with

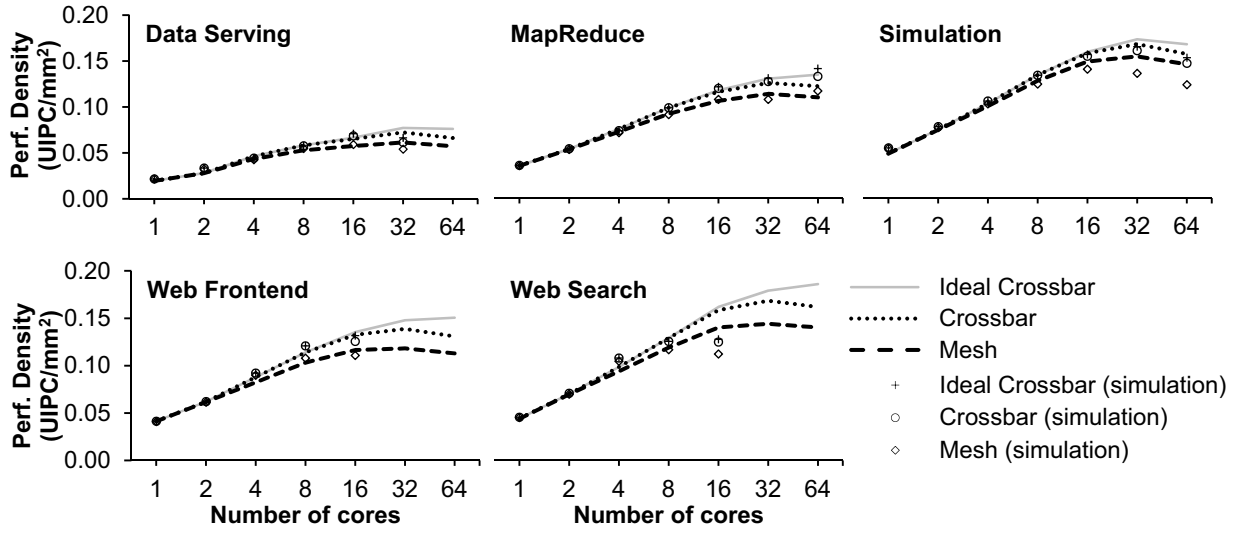

Figure 5. Results and cycle-accurate simulation results for designs with out-of-order cores and a 4MB LLC.

1 to 64 cores, various cache sizes, and three different on-chip interconnects. The details of the simulated architecture are listed in Table 2. *Flexus* models the SPARC v9 ISA and is able to run unmodified operating systems and applications. While the choice of the operating system might affect the performance scalability of our scale-out applications, it will not change the impact of physical distance on performance density. Therefore, we simulate systems running scale-out applications on the Solaris 10 operating system.

For performance evaluation, we use the SimFlex multiprocessor sampling methodology [45]. Our samples are drawn over an interval of five to ten seconds of simulated time. For each measurement, we launch simulations from checkpoints with warmed caches and branch predictors, and run 100K cycles (4M cycles for Web Frontend and Data Serving) to achieve a steady state of detailed cycle-accurate simulation before collecting measurements for the subsequent 50K cycles. As the performance metric, we use the number of user instructions committed per cycle[1] (UIPC), which is proportional to the overall system throughput [45]. We present average UIPC computed with 95% confidence and an error margin of ±4%.

5. Results

5.1 Performance density (PD) across scale-out applications

Figure 5 illustrates the performance density results for designs with out-of-order cores, a 4MB LLC, and different interconnect types across our scale-out applications. We plot one graph for each workload. Each graph consists of three lines representing the results obtained using the methodology described in Section 4 and three markers showing cycle-accurate simulation results.

Our results show that performance density of designs which integrate realistic interconnects such as *Crossbar* (dotted line) and *Mesh* (dashed line) starts diminishing above 32 cores, corroborating our intuition that physical distance between cores and the last-level cache hurts performance when integrating a high number of cores. In fact, our cycle-accurate simulation results show that Web Frontend and Web Search do not scale well beyond eight cores, thereby limiting performance density of pods that integrate higher number of cores. Furthermore, hardware complexity and various forms of contention limit performance gains for scalable workloads (e.g., Simulation), thereby causing performance density to peak at lower core counts compared to what is predicted by our model.

1. Total IPC is not representative of performance in full-system evaluation due to I/O and OS spin locks [45]. Like prior full-system work, we use User-IPC, which includes system cycles in the denominator (defined as committed user instructions divided by user+system cycles).

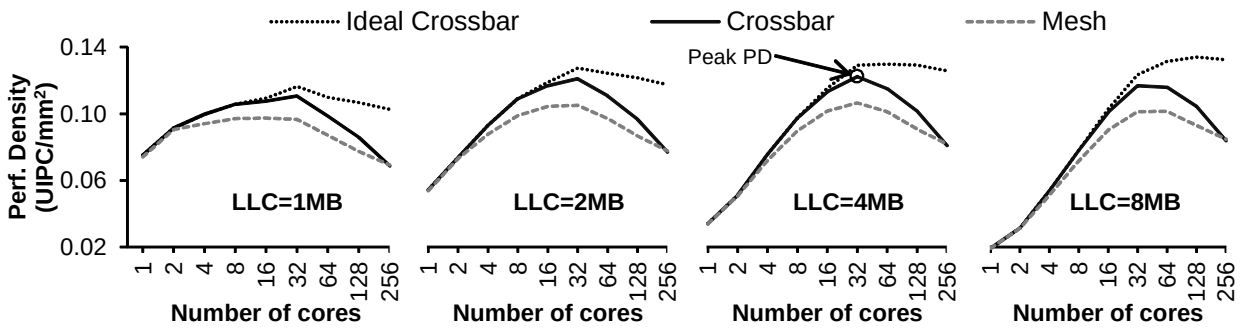

Figure 6. Performance density results for a system with out-of-order cores and various last-level cache sizes. Distance between cores and the last-level cache hurts performance as the number of cores increases. The arrow shows that the peak-PD pod employs 32 cores, tightly coupled with a LLC of 4MB.

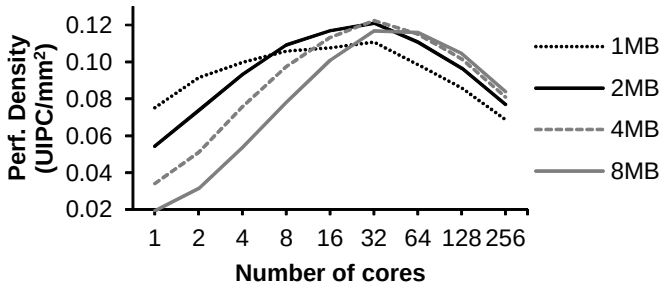

Figure 7. Performance density of pods based on a crossbar interconnect and various LLC sizes.

5.2 System with out-of-order cores

We begin our study using out-of-order cores in the 40nm technology. Figure 6 plots performance density — averaged across all applications — for four different LLC sizes. We do not consider cache sizes above 8MB, as bigger caches do not provide performance improvement (Section 2). Each graph consists of three lines, each of which corresponds to one of the three interconnects described in Section 4. The black dotted line, *Ideal Crossbar*, represents a perfectly scalable crossbar in which each core can access each cache bank with a fixed latency. The black line, *Crossbar*, represents a realistic crossbar implementation in which the cache bank access latency is a function of the physical distance between the core and the cache bank. The gray dashed line, *Mesh*, represents the mesh-based interconnect.

Across all cache configurations and realistic interconnects (i.e., *Crossbar* and *Mesh)* we observe that performance density starts diminishing above 32 cores, corroborating our intuition that physical distance between cores and the last-level cache hurts performance when integrating a large number of cores.

The PD-optimal pod employs 32 cores, tightly coupled to a 4MB LLC via a crossbar switch. However, software scalability, coherence complexity, and the difficulty of implementing a crossbar interconnect for a high number of cores are likely to shift the design toward a near-to-optimal pods with fewer cores. To explore this trade-off, Figure 7 examines performance density of pods based on a crossbar interconnect (shown with a black solid line in Figure 6) across various LLC sizes. Among designs with fewer than 32 cores, a pod which integrates 16 cores and 2MB of LLC provides the highest performance density and is within 4% of the optimum. We therefore adopt the 16-core 2MB LLC design with a crossbar interconnect as the preferred pod configuration due to a combination of high performance density and modest design complexity.

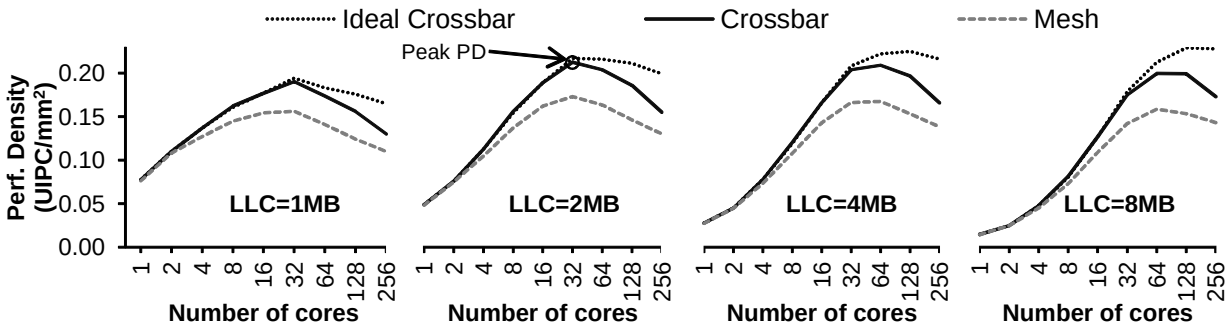

Figure 8. Performance density results for a system with in-order cores and various last-level cache sizes. Distance between cores and the last-level cache hurts performance as the number of cores increases. The arrow shows that the peak-PD pod employs 32 cores, tightly coupled with a LLC of 2MB.

5.2.1 Scale-Out Processor design

The preferred pod configuration occupies 83mm^2 and draws 18W of power for the cores, the last-level cache, and the crossbar interconnect, while achieving a performance density of 0.12 *User IPC per mm*^{2}. Off-chip bandwidth requirements for Data Serving, the most bandwidth-hungry application in our suite, are approximately 11GB/s for the 16 out-of-order cores.

Assuming a die size of 350 mm^2, representative of today's mid-range server chips, a *Scale-Out Processor* can integrate up to three pods, a quad-channel memory interface[1], and the rest of SoC components while consuming 81 W at peak power, well within limits of today's package technology. For comparison, the Nehalem processor in the 45nm technology node integrates three memory channels and consumes peak power of 80W.

The optimal design — among the practical ones which support full connectivity between cores and the LLC — employs 48 cores with 4MB of LLC and a mesh interconnect, a design which is similar to Tilera's Tile-Gx processor [46]. Compared to this design, our *Scale-Out Processor* design achieves 13% higher performance density, because the mesh-based interconnect introduces high on-chip communication delays (i.e., router and switch delays) between cores and the last-level cache slices, whereas a crossbar interconnect introduces only the switch delay.

5.3 System with in-order cores

In the recent years, there has emerged a trend towards simpler cores. Tilera [46] targets servers for cloud computing by integrating hundreds of simple in-order cores corroborating prior research which shows that designs with low-complexity cores are suited for server throughput applications [32]. Although such designs sacrifice latency, for services whose primary concern is throughput (e.g., data analysis), high-throughput designs that integrate simple in-order cores will be favored over designs which integrate fewer aggressive cores. Following this trend, we continue our study with in-order cores in 40nm technology. Figure 8 illustrates performance density results — averaged across all workloads — for four cache sizes and three different interconnects.

The figure shows that the trends are similar to those described in the previous section. In particular, our results show that a pod with 32 cores, 2MB of LLC, and a practical crossbar interconnect achieves the peak performance density across the configurations with implementable interconnects. We assume that the crossbar connecting the 32 cores and the last-level cache can be implemented by having multiple cores sharing one crossbar interface. Because the cores are simple and achieve low IPC, the impact of switch sharing is limited and in fact, Oracle's SPARC T3 uses this approach [37].

1. At 70% of memory bandwidth utilization, this memory interface provides 35.84GB/s.

5.3.1 Scale-Out Processor design

The optimal-PD pod occupies 52mm^2 and draws 17.4W of power for cores, caches, and a crossbar interconnect, and achieves a performance density of 0.21 *User IPC per mm*^{2}. Off-chip bandwidth requirements for Data Serving, the most bandwidth-hungry application, are approximately 16GB/s for 32 cores.

Compared to the optimal-PD pod which employs out-of-order cores, this pod achieves 75% higher performance density. Although an in-order core achieves approximately half the performance of an out-of-order core, it occupies 4x smaller area, thus providing higher performance density. These results corroborate prior work which shows that low-complexity cores are well-suited for throughput applications [11, 20].

Assuming a die size of 344mm^2, a *Scale-Out Processor* design can integrate up to four optimal-PD pods including seven memory channels and the rest of SoC components, yet staying within the bandwidth and power envelopes, as this design draws only 115W. For example, Oracle's Ultrasparc T2 [2] — in a die size of 342mm^2 — integrates eight memory channels and Intel's Gainestown's X5560 [22] has a maximum power density of 0.36 (i.e., in a die size of 263mm^2 the power consumption is 95W) which is higher than the maximum power density of our design in 40nm.

The optimal design — among the practical ones which support full connectivity between the cores and the LLC — employs 128 cores with an 8MB LLC and a mesh interconnect, a design which is similar to Tilera's Tile-Gx processor [46]. Compared to this design, our *Scale-Out Processor* design achieves 39% higher performance density.

Regardless of the core type, we demonstrate that designs which provide full connectivity between cores and the last-level cache are sub-optimal for scale-out workloads as router and switch delays significantly increase on-chip communication latency. However, a *Scale-Out Processor* design, that splits the available chip resources over multiple optimal-PD pods, overcomes on-chip communications delays, thus achieving maximum performance density.

5.4 Projection to 20nm technology

To understand the effect of technology scaling on the *Scale-out Processor* design, we project our systems with out-of-order and in-order cores for the 20nm technology. Wire, router, and switch delays as well as cache access latencies remain the same because the frequency does not change. As such performance per core and pod remains the same and hence the optimal-PD pod remains the same across technology generations.

while the area of memory channels does not scale across technologies, their frequency scales by a factor of two across two technology generations (i.e., 40nm => 32nm => 20nm), and hence the number of required memory channels per pod halves. While the area of the pod and the memory interface shrink by a factor of 0.25 and 0.5 respectively across two technology generations, performance density increases by ~3.5x.

Scaling our design that integrates out-of-order cores, the optimal-PD pod achieves a performance density of 0.40 *User IPC per mm*^{2} and occupies an area of 21mm^2, draws 7W of power, and requires an off-chip bandwidth of 11GB/s for 16 cores (i.e., 43% of the peak DDR4 memory channel's bandwidth). Assuming a die size of 332 mm^2 at a peak power consumption of 88W which integrates six DDR4 memory channels, the *Scale-Out Processor* design integrates 10 optimal-PD pods without hitting the bandwidth and power wall.

The optimal design among the practical ones that provide full connectivity between the cores and the LLC is a 160-core CMP with an 8MB LLC and a mesh interconnect. Compared to this design, the *Scale-Out Processor* design achieves 42% higher performance density.

Our projection shows that as future technologies allow for integrating more cores, the on-chip distance problem further becomes exacerbated. However, the *Scale-Out Processor* design maintains performance scalability across fabrication technologies by integrating multiple peak-performance-density pods to fully utilize the available chip resources.

When scaling our design with in-order cores, the optimal-PD pod achieves a performance density of 0.64 *User IPC per mm*^{2} and occupies an area of 13mm^2, draws 7W of power, and requires off-chip bandwidth of

Table 3. Performance density, area, power, and bandwidth requirements of various processor designs.

Processor design	40nm						20nm					
	PD	Area (mm^2)			Power (W)	BW GB/s	PD	Area (mm^2)			Power (W)	BW GB/s
		Core	Cache	Die				Core	Cache	Die		
Conventional	0.02	118	58	263	95	N/A	Data Not Available (N/A)					
Tiled Optimal (OoO)	0.106	216	20	329	79	33	0.28	180	10	317	82	110
Scale-Out (OoO)	0.12	216	30	349	81	33	0.40	180	25	332	88	110
Tiled Optimal (In-order)	0.15	166	40	344	115	64	*0.42*	*83*	*10*	*208*	*84*	*140*
Scale-Out (In-order)	0.21	166	40	344	115	64	*0.64*	*126*	*30*	*320*	*115*	*192*

Italicized numbers exceed available off-chip bandwidth

16GB/s for 32 cores (i.e., 63% of the peak DDR4 memory channel's bandwidth). Assuming a die size of 320 mm2 at a peak power consumption of 115W, the *Scale-Out Processor* design integrates 12 optimal-PD pods without any power constraint. However, this design requires eleven (11) DDR4 memory channels, for which the required number of pins cannot be provided without increasing the packaging cost, thereby hitting the bandwidth wall. This corroborates prior work [19] that shows that future's chips which integrate simple in-order cores will require various techniques (e.g., 3D-stacked DRAM caches) to mitigate the off-chip memory bandwidth limitations.

5.5 Summary

Table 3 summarizes performance density, die sizes, peak power consumption, and off-chip bandwidth requirements for various processor designs. We observe that a *Conventional* design[1], which uses large aggressive cores and large caches, achieves the worst performance density across all designs. Conventional designs have low performance density because, in addition to having an inappropriate core-to-cache area ratio for scale-out workloads, these aggressive cores provide only a small performance improvement over less aggressive cores, while occupying considerably larger area. Compared to the conventional design, the *Scale-Out (OoO)* design, which integrates less aggressive cores, smaller LLC, and splits the on-chip resources into multiple optimal-PD pods, achieves 6x higher performance density without a significant drop in single-thread performance.

We find that, at a cost to request latency, an *Optimal Tiled* design that integrates simple in-order cores and uses a mesh interconnect can achieve higher throughput, and therefore higher performance density, compared to the *Scale-Out (OoO).* However, the *Optimal Tiled* design loses 33% of performance density due to increased interconnect latency of its large on-chip network, when compared to a *Scale-Out (In-order)* design using the same core type.

We note that today's memory technology is over-provisioned, offering excessive off-chip bandwidth even for the highest-PD designs at 40nm. Projections of memory technologies also indicate that sufficient off-chip bandwidth exists for the 20nm *Scale-Out (OoO)* design. However, at 20nm, the *Scale-Out (In-order)* design calls for greater off-chip bandwidth than what will be available with conventional techniques, requiring innovation in memory bandwidth technologies (e.g., 3-D integrated DRAM or optical off-chip interconnects).

6. Related Work

There has been a lot of research effort trying to find the optimal CMP configuration. The prior work is either focused on finding the optimal cache architecture for a given number of cores [24, 47], or finding the optimal core configuration for certain applications [13]. Most of the prior work uses non-datacenter benchmarks [13, 14, 21, 36]. Oh et al. [36] presented a simple and effective analytic model to study the core

1. Computed by measuring the UIPC on a Nehalem-based Dell server blade; area estimated are based on a die micrograph.

count/cache capacity tradeoff in a CMP under a die area constraint, showing that for a fixed cache size increase in core count hurts the aggregate performance beyond a certain point. We show a similar trend for scale-out workloads. Huh et al. [21], investigated the area and performance trade-offs for CMP implementations to determine the optimal number of processing cores and their complexity for future server CMPs, and how big per-core on-chip caches should be. They observed that a small cache and many cores increase aggregate throughput, which we confirm for scale-out workloads. Their performance model employs private caches, which is not the right choice for scale-out workloads with huge instruction and data working set and also hides the impact of on-chip distance on performance. Hardavellas et al. [20], showed that for commercial server workloads, a modest sized cache is preferred over large caches, because larger caches suffer from longer access time latency.

In order to reduce access time to the LLC, Non-Uniform Cache Architectures (NUCA) were proposed [26]. Chishti proposed decoupling physical placement from logical organization [8] to add flexibility to the NUCA design. Taking advantage of the NUCA design, researchers aimed at further reducing the cache access latency by taking two orthogonal approaches: providing a faster and more efficient interconnect [27] and providing a better block placement in the NUCA cache [5, 18]. Similarly, our goal is to reduce on-chip distance and provide fast access to caches but we observed that pods have few cores and a small cache, which enables having a fast crossbar with short wires for core to cache communication.

There also has been related work with regard to using cores for most of the die area. Kgil et al. [25] proposed removing the last-level cache from the chip and use its area for more cores in conjunction with a 3D-stacked DRAM cache to decrease the off-chip traffic. Similarly, we find that increasing the fraction of compute unit on chip is important for the aggregate throughput but we also observed that scale-out workloads have reuse in their secondary working set and benefit from a modest cache size. The optimal configuration for scale-out workloads has a few megabytes of cache. GPUs are processors used in a different computation field and have many simple computational units [39] and a small amount of memory. As an example, Tesla C1060 GPUs have 240 cores and only 16Kb of shared memory per 8 cores [35]. Similar to GPUs, a significant fraction of die area in scale-out chips is used for cores.

There are already manufactured chips that share some of the insights or conclusions of this work. Piranha [4] is the first chip multi-processor designed for commercial server workloads that used simple cores to get higher efficiency. In this work, we also showed that using simple cores for scale-out workloads would result in having pods with higher performance density. Among today's commercial processors, Niagara III [37] is the closest to our optimal pod. Niagara III has 16 cores, 128 hardware contexts, a few megabyte of cache (i.e., 6MB) and provides fast access from cores to caches using a high-speed crossbar. Consequently, Niagara III can be used as a building block (i.e., pod) for designing scale-out processors.

7. Conclusions

Performance density is a critical metric for determining the efficiency of processor designs, directly translating to datacenter cost and capacity. In this work, we showed that today's conventional processor designs and processor designs specifically targeting the datacenter space make sub-optimal use of the die real-estate. We demonstrated that the core-to-cache area ratio of today's designs is sub-optimal, resulting in designs that over-provision cache capacity at the cost of core count. Furthermore, we demonstrated that the scale-up methodology of today's processors leads to a decline in performance density in future technology generations. To overcome both of these performance-density limiters, we proposed a processor design methodology based on pods, optimally-efficient processor building blocks. Pod-based designs maximize per-processor throughput in today's technology and provide a simple mechanism to scale the designs to future technologies without diminishing performance density. We used a combination of model-driven analysis and cycle-accurate simulation to demonstrate our design methodology, achieving 13% and 42% throughput improvement over tiled processor designs in 40nm and 20nm technologies, respectively.

8. Acknowledgements

We thank the EuroCloud project partners for advocating and inspiring the scale-out processors. This work was partially supported by EuroCloud, Project No 247779 of the European Commission 7th RTD Framework Programme – Information and Communication Technologies: Computing Systems.

9. REFERENCES

[1] Alexa, The Web Information Company. http://www.alexa.com.

[2] An 8-core, 64-thread, 64-bit, power efficient SPARC SoC. http://www.opensparc.net/pubs/preszo/07/n2isscc.pdf.

[3] M. Baron. The F1: TI's 65nm Cortex-A8. *Microprocessor Report*, 20(7):1 –9, July 2006.

[4] L. A. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese. Piranha: a scalable architecture based on single-chip multiprocessing. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, June 2000.

[5] B. M. Beckmann, M. R. Marty, and D. A. Wood. ASR: adaptive selective replication for CMP caches. In *Proceedings of the 39th Annual International Symposium on Microarchitecture*, Dec. 2006.

[6] S. Bucur, V. Ureche, C. Zamfir, and G. Candea. Parallel symbolic execution for automated real-world software testing. In *Proceedings of the 6th ACM SIGOPS/EuroSys Conference on Computer Systems*, Apr. 2011.

[7] Calxeda Launches the EnergyCore Processor. http://www.calxeda.com/company/newsroom/pressreleases/63-calxeda-launches-energycore-processor, Nov. 2011.

[8] Z. Chishti, M. D. Powell, and T. N. Vijaykumar. Distance associativity for high-performance energy-efficient non-uniform cache architectures. In *Proceedings of the 36th Annual International Symposium on Microarchitecture*, Dec. 2003.

[9] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, June 2010.

[10] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu. Memory power management via dynamic voltage/frequency scaling. In *Proceedings of the 8th ACM International Conference on Autonomic Computing*, June 2011.

[11] J. D. Davis, J. Laudon, and K. Olukotun. Maximizing CMP throughput with mediocre cores. In *Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques*, Sept. 2005.

[12] J. Dorsey, S. Searles, M. Ciraula, S. Johnson, N. Bujanos, D. Wu, M. Braganza, S. Meyers, E. Fang, and R. Kumar. An integrated quad-core Opteron processor. In *International Solid-State Circuits Conference*, Feb. 2007.

[13] M. Ekman and P. Stenstrom. Performance and power impact of issue-width in chip-multiprocessor cores. In *Proceedings of the International Conference on Parallel Processing*, Oct. 2003.

[14] M. Farrens, G. Tyson, and A. R. Pleszkun. A study of single-chip processor/cache organizations for large numbers of transistors. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, Apr. 1994.

[15] J. Friedrich, B. McCredie, N. James, B. Huott, B. Curran, E. Fluhr, G. Mittal, E. Chan, Y. Chan, D. Plass, S. Chu, H. Le, L. Clark, J. Ripley, S. Taylor, J. Dilullo, and M. Lanzerotti. Design of the Power6 microprocessor. In *International Solid-State Circuits Conference*, Feb. 2007.

[16] T. R. Halfhill. ARM's midsize multiprocessor. *Microprocessor Report*, 23(10):17 –24, Oct. 2009.

[17] N. Hardavellas. *Chip-multiprocessors for server workloads*. PhD thesis, Carnegie Mellon University, July 2009.

[18] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Reactive NUCA: near-optimal block placement and replication in distributed caches. In *Proc. of the 36th Annual International Symposium on Computer Architecture*, June 2009.

[19] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Mi-*

cro, 31(4):6 –15, Jul-Aug 2011.

[20] N. Hardavellas, I. Pandis, R. Johnson, N. Mancheril, A. Ailamaki, and B. Falsafi. Database servers on chip multiprocessors: limitations and opportunities. In *The 3rd Biennial Conference on Innovative Data Systems Research*, Jan. 2007.

[21] J. Huh, D. Burger, and S. W. Keckler. Exploring the design space of future CMPs. In *Proceedings of the 10th International Conference on Parallel Architectures and Compilation Techniques*, Sept. 2001.

[22] Intel Xeon Processor X5560. http://ark.intel.com/products/37109/Intel-Xeon-Processor-X5560-(8M-Cache-2_80-GHz-6_40-GTs-Intel-QPI).

[23] JEDEC Announces Key Attributes of Upcoming DDR4 Standard. http://www.jedec.org/news/press-releases/jedec-announces-key-attributes-upcoming-ddr4-standard, Aug. 2011.

[24] N. P. Jouppi and S. J. E. Wilton. Tradeoffs in two-level on-chip caching. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, Apr. 1994.

[25] T. Kgil, S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, T. Mudge, S. Reinhardt, and K. Flautner. PicoServer: using 3D stacking technology to enable a compact energy efficient chip multiprocessor. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2006.

[26] C. Kim, D. Burger, and S. W. Keckler. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.

[27] J. Kim, W. J. Dally, and D. Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, June 2007.

[28] R. Kumar and G. Hinton. A family of 45nm IA processors. In *International Solid-State Circuits Conference*, Feb. 2009.

[29] N. A. Kurd, S. Bhamidipati, C. Mozak, J. L. Miller, T. M. Wilson, M. Nemani, and M. Chowdhury. Westmere: a family of 32nm IA processors. In *International Solid-State Circuits Conference*, Feb. 2010.

[30] 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 International Symposium on Microarchitecture*, Dec. 2009.

[31] S. Li, K. Lim, P. Faraboschi, J. Chang, P. Ranganathan, and N. Jouppi. System-level integrated server architectures for scale-out datacenters. In *Proceedings of the 44th Annual International Symposium on Microarchitecture*, Dec. 2011.

[32] K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, and S. Reinhardt. Understanding and designing new server architectures for emerging warehouse-computing environments. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, June 2008.

[33] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *Proc. of the 40th Annual International Symposium on Microarchitecture*, Dec. 2007.

[34] NGINX, Inc. http://nginx.com/index.html.

[35] NVIDIA Tesla Computing Processor. http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_C1060_US_Jan10_lores_r1.pdf.

[36] T. Oh, H. Lee, K. Lee, and S. Cho. An analytical model to study optimal area breakdown between cores and caches in a chip multiprocessor. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, May 2009.

[37] Oracle's Netra SPARC T3-1 Server Architecture. http://www.oracle.com/technetwork/articles/systems-hardware-architecture/ovm-ref-tech-302440.pdf, Feb. 2011.

[38] V. J. Reddi, B. C. Lee, T. Chilimbi, and K. Vaid. Web search using mobile cores: quantifying and mitigating the price of efficiency. In *Proceedings of the 37th Annual International Symposium on*

Computer Architecture, June 2010.

[39] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: a many-core x86 architecture for visual computing. In *Proceedings of ACM SIGGRAPH*, Aug. 2008.

[40] SPECweb2009. http://www.spec.org/web2009/.

[41] The Apache Cassandra Project. http://cassandra.apache.org.

[42] The Apache Nutch Project. http://nutch.apache.org.

[43] J. Turley. Cortex-A15 "Eagle" flies the coop. *Microprocessor Report*, 24(11):1 –11, Nov. 2010.

[44] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor. Conservation cores: reducing the energy of mature computations. In *Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2010.

[45] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe. SimFlex: statistical sampling of computer system simulation. *IEEE Micro*, 26(4):18–31, Jul-Aug 2006.

[46] B. Wheeler. Tilera sees opening in clouds. *Microprocessor Report*, 25(7):13 –16, July 2011.

[47] L. Zhao, R. Iyer, S. Makineni, J. Moses, R. Illikkal, and D. Newell. Performance, area and bandwidth implications on large-scale CMP cache design. In *Proceedings of the 1st Workshop on Chip Multiprocessor Memory Systems and Interconnects*, Feb. 2007.