Position Paper: Embracing Heterogeneity— Improving Energy Efficiency for Interactive Services on Heterogeneous Data Center Hardware

Yuxiong He and Sameh Elnikety
Microsoft Research

Abstract

Data centers today are heterogeneous: they have servers from multiple generations and multiple vendors; server machines have multiple cores that are capable of running at difference speeds, and some have general purpose graphics processing units (GPGPU). Hardware trends indicate that future processors will have heterogeneous cores with different speeds and capabilities. This environment enables new advances in power saving and application optimization. It also poses new challenges, as current systems software is ill-suited for heterogeneity.

In this position paper, we focus on interactive applications and outline some of the techniques to embrace heterogeneity. We show that heterogeneity can be exploited to deliver interactive services in an energy-efficient manner. For example, our initial study suggests that neither high-end nor low-end servers alone are very effective in servicing a realistic workload, which typically has requests with varying service demands. High-end servers achieve good throughput but the energy costs are high. Low-end servers are energy-efficient for short requests, but they may not be able to serve long requests at the desired quality of service. In this work, we show that a heterogeneous system can be a better choice than an equivalent homogeneous system to deliver interactive services in a cost-effective manner, transforming heterogeneity from a resource management nightmare to an asset. We highlight some of the challenges and opportunities and the role of AI and machine learning techniques for hosting large interactive services in data centers.

Introduction

Data centers have heterogeneous hardware from multiple vendors. At a large scale, it is hard to keep servers homogenous, with the same hardware and performance characteristics. One main source of heterogeneity is the fact that data centers operate several generations of servers from multiple vendors. Application needs, hardware innovations and pricing determine which batch of servers to buy. We also expect that in the future the degree of heterogeneity will increase, even going from server-level heterogeneity to heterogeneity within a single server system. More specifically, Hardware trends [1] suggest that future servers will have multiple cores running at different speeds. Moreover, reconfigurable processor hardware [2,3,4] will allow composing several thin cores into a fat core.

Although there are many prior studies [4, 5, 6] on heterogeneous systems, they mostly focus on batch workloads without time constraints. The impact of heterogeneity on interactive data center services with time-bounded computations and stringent SLA requirement remains largely unstudied. This class of computation constitutes a large portion of data center workloads and includes important applications, such as web search, web content serving and video-on-demand. Interactive workloads require a substantial amount of computational resources and energy each year. In large data centers, even a few percent of improvement in server throughput can save millions of dollars in operating expenses.

There is a debate on whether to use high-end fast servers or low-end energy-efficient servers to host large-scale interactive services. Our initial study suggests that neither high-end nor low-end servers alone are very effective in servicing realistic workloads, which typically include requests with varying service demands. On one hand, high-end servers achieve good throughput but the energy costs are high. On the other hand, low-end servers are cost effective for short requests, but they may not be able to serve long requests at the desired quality of service. In this research, we show that heterogeneity can be exploited to deliver interactive services in a cost-effective manner. A heterogeneous system can be a better choice than an equivalent homogeneous system, transforming heterogeneity from a resource management nightmare to an asset.

Scheduling Model

We present a scheduling model for interactive requests to study the impact of using heterogeneous hardware. We assume a set $JobSet = \{J_1, J_2, \ldots, J_n\}$ of n requests (or jobs). Each request $J_i \in JobSet$ is characterized by an *arrival time* r_i, *deadline* d_i, and *service demand* (aka *total work*) w_i. For each request J_i, a scheduling algorithm starts to execute the request at time s_i and completes it at c_i, where $s_i \geq r_i$ and $c_i \leq d_i$. Let p_i denote the *processed work* request J_i receives during $[s_i, c_i]$; p_i represents the total amount of work done for request J_i before its deadline. Here a scheduler can either be preemptive or non-preemptive depending on the implementation environment. A preemptive scheduler can suspend a request from one processor and resume it on another processor with the same or different speed. The value of p_i is accumulated by adding the contribution of the processors running request J_i, and this relates to both the processing time and the relative speed of processors. A *quality function* f_i: $R \rightarrow R$ maps the processed work p_i of request J_i to a quality value gained by executing the request.

The objective of the scheduler is to minimize the energy consumption of servicing these requests while satisfying the SLA requirements of the applications. For example, an SLA of a web search engine can specify that requests should be satisfied within 150 ms, and obtain an average response quality greater than 99.8%.

This scheduling model is general and it can be adapted to match application characteristics, execution environment and hardware.

Quality profile. Different applications can have quality functions with various properties. For example, traditional server components often require a request to be fully processed by its deadline to return any result; the corresponding quality profile is a step function. However, many interactive applications such as web search and on-demand video server, find the best available result within a predefined response time. In these applications, the response quality improves with the increased processing time [7].

Scheduler types. A scheduler can be clairvoyant or non-clairvoyant; a clairvoyant scheduler knows the service time of the request when it arrives. The execution environment may allow the scheduler to be preemptive.

In the next section we show a simple example showing the benefits of a heterogeneous system, consisting of fast and slow servers with homogeneous cores. We use a scheduler which is clairvoyant and non-preemptive, and we assume that the request quality is proportional to its processing time.

Motivating Example

We present a simple example to show the benefits of using a heterogeneous system. We assume two types of servers and a workload from an interactive service and study the number of servers required under an SLA (service level agreement) similar to those used in Web search.

Servers. There are two kinds of servers: fast and slow. A fast server has one fast core, and a slow server has 15 slow cores. A fast core is three times as fast as a slow core, and consumes 15 times more power than a slow core (power is between a quadratic or cubic function of speed; 15 is between 3^2=9=and 3^3=27). The CPU power consumption of a fast server and a slow server are the same. Table 1 and 2 describe their relative speed and power consumptions.

Core	Speed	Power Consumption
Fast core	3X	15X
Slow core	1X	1X

Table 1. Speed and power consumption of fast and slow cores.

Machine	Cores	CPU Power Consumption
Fast Server	1 fast core	15X
Slow Server	15 slow cores	15X

Table 2. Speed and CPU power consumption of fast and slow servers.

	Small request	Long request
Fast core	10 ms	40 ms
Slow core	30 ms	120 ms

Table 3. Service demands of small and long requests on fast and slow cores

Configuration.	Sustainable QPS on a slow-core	#slow servers	Sustainable QPS on fast core	#fast servers	#total servers	Description
A	-	-	54	18.5	18.5	fast servers only
B	3	22.2	-	-	22.2	slow servers only
C	17	3.5	10	10	13.5	900 short requests to slow servers; 100 long ones to fast servers
D	15	3.6	21	9.0	12.6	800 short + 10 long requests to slow servers; 100 short + 90 long requests to fast servers

Table 4. Hardware requirement and scheduling choices

Workload. We assume a load of 1000 requests per second. Among them, there are 900 short requests and 100 long requests, whose service demands are described in Table 3. The requests arrive following a Poisson distribution.

Request Quality Profile. The deadline of each request is 150 ms. We allow partial results: within the deadline, if the request is fully satisfied, the quality of the result is 1; if the assigned processing time of a request is p fraction of the total service time, the quality of the result is p.

Target SLA. The quality requirement is 99.8%, i.e. the average quality of the returned results must be at least 99.8%.

Hardware and scheduling choices. Table 4 presents four configurations, A, B, C, and D, with various server choices. In configurations A and B, we use homogeneous systems: Configuration A uses fast servers only and B uses slow servers only. In configurations C and D, we use heterogeneous systems by combining fast and slow servers. In each configuration, we show the number of servers needed to satisfy the SLA. Since our fast server and slow server have the same CPU power consumption, the number of total servers needed represents the energy requirement for different configurations. We developed a discrete-event simulator to model processing of requests on different configurations. In the table, the sustainable arrival rate is obtained by simulation using a FIFO scheduler while meeting the 99.8% SLA requirement at 150 ms deadline.

Table 4 shows that homogeneous configurations are not efficient. We can use either 18.5 fast servers (configuration A) or 22.2 slow servers (configuration B). However, if we use a heterogonous system by dispatching long requests to fast servers and short requests to slow servers, we reduce the number of servers to 13.5. Moreover, a more advanced dispatcher can send 100 short requests + 90 long requests to fast servers and the remainder to slow servers, reducing the total number of servers to 12.6. Compared to the better choice of a homogeneous configuration (fast servers only in configuration A), a heterogeneous system (configuration D) saves 30% of the total energy consumed by CPU. We conclude that heterogeneous systems open optimization opportunities that are not possible with equivalent homogenous systems.

Opportunities and Challenges

The motivating example demonstrates the benefits of using heterogeneous servers in a simple case where requests have only two different service times, however in practice, the variance of requests can be much larger. For example, in the production Bing search workload, we observe that the request service time distribution has a tail, where the average service time is around 15 ms but more than 10% of requests have service time larger than 100 ms. Similar observations have been made on many other workloads [8]. The large variance of request service time further increases the benefits of using heterogeneous systems: we can use-fast servers to handle long requests to meet their SLA and use slow servers to handle a large percentage of small and median requests to achieve good energy efficiency.

Although heterogeneity brings potential opportunities to offer cost-effective interactive services, some challenges come along the way: how to use heterogeneous systems effectively? There are many questions to answer and issues to resolve:

Service Time Prediction. Request service time is an important factor to decide where to schedule a request, how-

ever, it is not known for some applications. Is there an effective way to predict the request service time? For example, in web search, the service time of a query is related to the number of keywords, the number of matching documents, whether the index is available in memory or stored on hard disk. Such information is available when a query arrives without demanding extensive processing. It would be interesting to explore whether and how we can build a prediction model using such information by offline learning from historical data with assistance of light-weight online learning.

Resource Management. To schedule requests on heterogeneous hardware, intuitively, it is desirable to dispatch long requests to fast servers, and short requests to slow servers, saving hardware and energy costs while meeting target quality of service. However, a naïve way to divide requests based on a threshold of service demand does not offer the best solution. A better allocation need to consider the performance of processors, request service time distribution, request quality profile, load on the servers, and the desired SLA. This is a complex system with many parameters. Machine learning techniques can be very helpful here to monitor the status of the complex system, extract performance features, and perform adaptive resource management. In particular, developing such a resource management system is an open research question that requires advances in performance modeling and analysis.

Hardware Selection. We want to investigate how to schedule requests both among a heterogeneous cluster with different homogeneous servers, and on servers with heterogeneous cores.

- If we can select only two types of servers, how will we choose the speed of the servers? How does this problem evolve with more server types? When do we get most of the gain with a least types of servers?
- New technology trends allow us to dynamically adjust frequency and voltage of cores on a processor chip. How can we configure the existing hardware to the best operational point?
- Reconfigurable hardware [2, 3, 4] allows composing several thin cores into a fat core. How can we exploit this flexibility to improve energy efficiency?

Given such a large space of hardware and configurations to explore, we hope that search and optimization techniques in AI can help us to effectively trim the search space and lead us to a pool of good selections.

In summary, interactive data center services like web search, map search, online gaming use a substantial amount of computational resources and energy each year. These services can benefit from heterogeneity in data centers, but there is little prior work addressing this issue. Given the complexity of the system, machine learning and AI are promising techniques to optimize such systems and to develop an effective resource management framework. There are many interesting research problems to explore.

References

[1] M. D. Hill and M. R. Marty. Amdahl's law in the multicore era. In IEEE Computer, pages 33–38, 2008.

[2] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, N. Ranganathan, D. Burger, S.W. Keckler, R. G. McDonald, and C. R. Moore. Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture. In International Symposium on Computer Architecture, pages 422–433, 2003.

[3] S. Swanson, K. Michaelson, A. Schwerin, and M. Oskin. WaveScalar. In Symposium on Microarchitecture, pages 291–302, 2003.

[4] Divya Gulati, Changkyu Kim, Simha Sethumadhavan, Stephen W. Keckler, Doug Burger: Multitasking workload scheduling on flexible-core chip multiprocessors. In PACT 2008: 187-196.

[5] Rakesh Kumar, Dean M. Tullsen, Parthasarathy Ranganathan, Norman P. Jouppi, Keith I. Farkas. Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload PerformanceInternational. In Symposium on Computer Architecture, pages 64-75, 2004.

[6] Kevin T. Lim, Parthasarathy Ranganathan, Jichuan Chang, Chandrakant D. Patel, Trevor N. Mudge, Steven K. Reinhardt. Understanding and Designing New Server Architectures for Emerging Warehouse-Computing Environments. In International Symposium on Computer Architecture, pages.315-326, 2008.

[7] Yuxiong He, Sameh Elnikety, Hongyang Sun. Tians Scheduling: Using Partial Processing in Best-Effort Applications. ICDCS 2011, Minneapolis, MN, June 20-24, 2011.

[8] Mor Harchol-Balter, "The Effect of Heavy-Tailed Job Size. Distributions on Computer System Design," In Proceedings of ASA-IMS Conference on Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics, Washington, DC, June 1999.