Heterogeneity in "Homogeneous" Warehouse-Scale Computers: A Performance Opportunity

Jason Mars[1], Lingjia Tang[1], Robert Hundt[2]
[1]University of Virginia, [2]Google
{jom5x,lt8f}@cs.virginia.edu, rhundt@google.com

Abstract—The class of modern datacenters recently coined as "warehouse scale computers" (WSCs) has traditionally been embraced as homogeneous computing platforms. However, due to frequent machine replacements and upgrades, modern WSCs are in fact composed of diverse commodity microarchitectures and machine configurations. Yet, current WSCs are designed with an assumption of homogeneity, leaving a potentially significant performance opportunity unexplored. In this paper, we investigate the key factors impacting the available heterogeneity in modern WSCs, and the benefit of exploiting this heterogeneity to maximize overall performance. We also introduce a new metric, *opportunity factor*, which can be used to quantify an application's sensitivity to the heterogeneity in a given WSC. For applications that are sensitive to heterogeneity, we observe a performance improvement of up to 70% when employing our approach. In a WSC composed of state-of-the-art machines, we can improve the overall performance of the entire datacenter by 16% over the status quo.

Index Terms—Super (very large) computers, Heterogeneous (hybrid) systems, Scheduling and task partitioning, Design studies.

1 Introduction

As more of today's computing moves into the cloud, the computing platform recently coined as *warehouse-scale computers* (WSCs) [2, 6, 7] is emerging as an important area of study for computer architecture research [3]. WSCs have been embraced as homogeneous computing environments [1, 2, 4]. However, as acknowledged in prior work [9], this is not the case in practice. As machines are replaced in these datacenters, new generations of hardware are deployed while older generations continue to operate. This leads to a WSC that is composed of a mix of machine platforms, e.g. a *heterogeneous* WSC. As we show in this work, ignoring this heterogeneity can lead to inefficient execution of applications in these datacenters.

The heterogeneity in WSCs differs than that found in a heterogeneous multicore chip. In a WSC, it is the diversity in *execution environments* that must be considered. We define an application's *execution environment* as the set of all factors that can impact the execution of the application. These include an application's machine configuration, underlying microarchitecture, simultaneously co-located jobs, the state of the machine's system software and its configuration, among others. In this paper we focus our study on the heterogeneity in the underlying microarchitecture as well as the set of jobs simultaneously running on the machine.

Another difference between the heterogeneous WSC and other heterogeneous computing platforms is that there is a set of key applications (search, maps, etc) that are run in these datacenters continually. These applications and the machine platforms are known a priori by the WSC operators and job management runtimes. This observation leads to an important insight. The performance opportunity present from the heterogeneity in machines are defined by the mix of applications that will run on these machines. In turn, the performance opportunity present from the diversity in applications running in the heterogeneous datacenter is defined by the particular mix of underlying machines. As we vary either, the amount of performance opportunity changes significantly.

This paper makes a number of important contributions:

- **Quantifying Datacenter Heterogeneity:** We demonstrate the performance variability endured by applications as they are placed in various execution environments in WSCs.
- **Opportunity Factor:** We introduce a metric, the *opportunity factor*, that quantifies how sensitive an application is to the heterogeneity in a datacenter. This metric approximates the performance opportunity for a given application when mapped intelligently.
- **Opportunistic Mapping and Map Scoring:** We present *opportunistic mapping* of jobs to machines to take advantage of the heterogeneity in "homogeneous" WSCs. A required component of such an approach is the ability to score and rank job placement mappings. We provide four such map scoring policies and discuss the key trade-offs between them.
- **Heterogeneity in Production:** We demonstrate the effectiveness of our techniques in production datacenters running Google internet services including web search.

In this work, we use both an experimental testbed and production datacenters running live internet services to evaluate the potential of exploiting the heterogeneity in modern WSCs. We show that individual applications can improve by up to 70% when employing our *opportunistic mapping*. Overall, we can improve the performance of an entire cluster composed of three types of state-of-the-art machines by 16%. We confirm this result in the wild on Google's production datacenters and workloads with a postmortem analysis, demonstrating a 15% performance

• *Manuscript submitted: 30-Apr-2011. Manuscript accepted: 23-May-2011. Final manuscript received: 30-May-2011*

TABLE 1
Production Microarchitecture Mix

Type	CPU	GHz	Cores	L2/L3	Mem.
Production	Xeon E5345	2.33ghz	2x4	4x4mb	conf.
Production	Opteron 8431	2.4ghz	6	6mb	conf.
Testbed	Core i7 920	2.67ghz	4	8mb	4gb.
Testbed	Core 2 Q8300	2.5ghz	4	4mb	3gb.
Testbed	Phenom X4 910	2.6ghz	4	6mb	4gb

TABLE 2
Production Application Mix

Applications	Description
docs	content analysis
bigtable	storage software for massive amount of data
websearch	Google search engine
S (co-runner)	image processing and computer vision
P (co-runner)	protocol buffer

improvement. Improving the overall performance of jobs running in WSCs improves the cost efficiency of building and operating the datacenter [5, 6, 12]. At the scale of Google, a 1% improvement results in millions of dollars saved.

2 Heterogeneity in WSCs

In this section, we demonstrate and compare the performance variability introduced by microarchitectural and co-runner heterogeneity in real production datacenters with the variability present in our experimental testbed. The machine types that compose these platforms are presented in Table 1. The amount of memory used on our production machines is confidential. The Google applications we use in the study are described in Table 2. These applications cover a number of Google's large industry-strength workloads [12]. Figure 1 illustrates the microarchitectural and co-runner heterogeneity for these three key applications. The number of threads and input workloads are the same for each application on both architectures and are composed of real queries and requests from production. The y-axis shows the performance slowdown of each application when it is running alone on the Opteron as well as co-running with P and S on both architectures, compared to its performance when running alone on the Xeon. As the figure shows, there is a significant performance variability across various co-running pairs and microarchitectures. In addition, applications have different microarchitectural preferences when its co-runner changes. For example `Doc` prefers to run on Xeon and has a 40% performance swing across various scenarios. `Bigtable` prefers to run on Opteron and has a 50% performance swing. `Websearch` prefers to run on Opteron when running alone and when co-ruinnng with `P`. However, it prefers Xeon when running with `S`.

We performed a similar study in our experimental testbed and observed the same phenomena. In our experimental infrastructure we use 22 SPEC CPU2006 benchmarks on their `ref` input as our application types and three types of state-of-the-art microarchitectures presented in Table 1. Figure 2 illustrates the performance variability when co-locating each application with `lbm` on the three microarchitectures. We use `lbm` as it is a well known memory intensive benchmark and will exhibit various levels of contentiousness across various co-running applications and memory subsystem designs. In this experiment, we observe that the benchmark workloads on our testbed have a similar sensitivity to

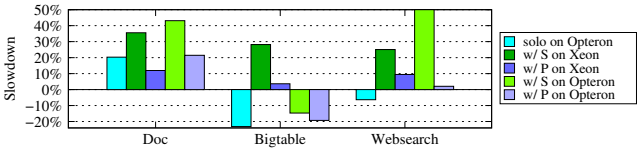

Fig. 1. Google workload sensitivity to machine and co-runner diversity. The baseline is each application's solo run on Xeon.

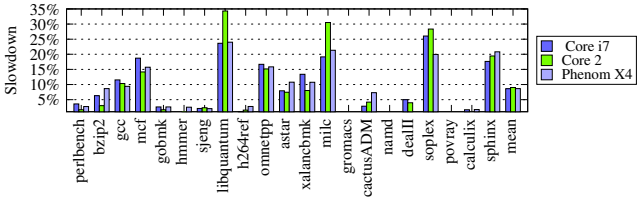

Fig. 2. Application sensitivity to microarchitectural diversity when co-located with `lbm` compared to running alone.

microarchitectural and co-runner diversity. Each microarchitecture serves as the poorest and best performer in various cases across the applications. In summary, we observe a significant performance variability from the heterogeneity in both production datacenters and our testbed.

3 Opportunity Factor

An important insight arises from the previous section. Depending on how "immune" an application is to microarchitectural and co-runner variation, each application would benefit differently from a job mapping policy that takes advantage of heterogeneity. We introduce a metric, *opportunity factor*, that approximates a given application's potential performance improvement opportunity relative to all other applications, given a particular mix of applications and machine types. The higher the opportunity factor, the more an application can potentially benefit from exploiting the heterogeneity in the WSC. Note that this opportunity factor can be calculated only if the application mix and the machine mix are known. For this we use services such as Google Wide Profiling.

[Google Wide Profiling] In modern production WSCs, continuous monitoring and logging systems are in place that collect and archive performance information in production. Google uses one such system, the Google Wide Profiler [10]. GWP is a standard component in the software stack of each production deployment. This mechanism is used to provide historical and continuous performance information as a service to operators and other software subsystems in the datacenter. For the remainder of this work, we describe the opportunity factor and opportunist mapper, both of which leverage this service. It is important to make the distinction between the costs associated with populating the GWP database (refered to later as *profiling complexity*), versus the cost of using the information in GWP's database, which is in the order of minutes.

[Opportunity Factor] For a given WSC, we can denote the application of type i as A_i, and the microarchitecture of type j as M_j. We define the speedup factor for A_i as:

$$SF_{A_i} = \frac{max_{j,k}\{IPS_{A_i,M_j,C_k}\} - min_{j,k}\{IPS_{A_i,M_j,C_k}\}}{min_{j,k}\{IPS_{A_i,M_j,C_k}\}}, \quad (1)$$

where IPS_{A_i,M_j,C_k} is application A_i's IPS (instruction per second) when it is running on machine M_j with a set of co-

TABLE 3
Mapping Scoring Policies

Policy	Description	Complexity
OM-C	**Colocation Score:** This score is based only on co-location penalty and only requires profiling the co-location penalty on any type of machine. Once a co-location profile is collected it is then used to score that co-location regardless of the underlying microarchitecture.	$\|A\|^n$
OM-Cs	**Colocation Score (Smart):** This score is based on co-location penalty with microarchitecture specific information. Information about co-location penalty must be collected for all platforms of interest.	$\|A\|^n \times \|M\|$
OM-M	**Microarchitectural Affinity Score:** This score is based on microarchitectural affinity and captures only the speedup of running each application on one microarchitecture over another.	$\|A\| \times \|M\|$
OM-MCs	**Microarchitectural Affinity and Colocation Score:** This scoring method includes both microarchitectural affinity and microarchitecture specific co-location penalty. Profiling includes both solo and corun performance (e.g. n+1). This scoring technique has the heaviest profiling requirements.	$\|A\|^{n+1} \times \|M\|$

runners C_k. The SF_{A_i} is essentially the amount of performance variability of A_i in all possible configurations of the execution environment, composed of the cross product of all machine options and all co-runner options. Using SF_{A_i}, we can define the *Opportunity Factor* (OF) for A_i as:

$$OF_{A_i} = \frac{SF_{A_i}}{\sum_j SF_{A_j}} \quad (2)$$

This OF_{A_i} represents the sensitivity of each application type to the overall heterogeneity of a given application mix, relative to all other applications. This metric allows datacenter designers and operators to identify applications that are most likely to benefit from heterogeneity-aware job mapping. This capability is of particular importance when managing the QoS requirements of various latency sensitive applications as applications that are more sensitive to heterogeneity must be placed carefully in the datacenter to ensure their QoS targets. We evaluate the accuracy of OF in Section 4.

4 Opportunistic Mapping

We formulate the problem of mapping jobs of different types and different characteristics to a set of heterogeneous machine resources as a combinatorial optimization problem. Note that special rules can be applied for latency sensitive applications that have a QoS requirement that may be affected by certain co-runners [8]. Our *opportunistic mapper* is essentially a solver for the optimization problem of finding the optimal mapping. The core algorithm we use to address the main optimization problem is based on well established iterative optimization techniques [11, 13]. However, to compare maps, we need an approach to score maps.

[Map Scoring] An essential part in the *opportunistic mapper* is the scoring policy used in each optimization iteration to compare the mappings' performance. To produce a *score* of a job's performance in a particular placement, we mine GWP's profiling information. To score an entire map of jobs to machines we use the sum of all of the placement scores. The higher the score, the better the map. In this work, we present and evaluate a number of scoring policies. As GWP is run continually in production, more information becomes available as time passes. We use the term *profiling complexity* to refer to the amount of information needed from GWP for the ideal functioning of the scoring policy. Table 3 shows the description and the profiling complexities for our map scoring policies, where $|A|$ corresponds to the number of application types, $|M|$ corresponds to the number of machine types, and n corresponds to the number of co-runners allowed. The key trade-off when selecting a scoring policy is between the amount of profiling information needed for ideal functioning and maximizing the performance gain. In addition, the diversity present in the WSC has a significant impact on the usefulness of some information. For example, if there is little microarchitectural diversity OM-C may be sufficient.

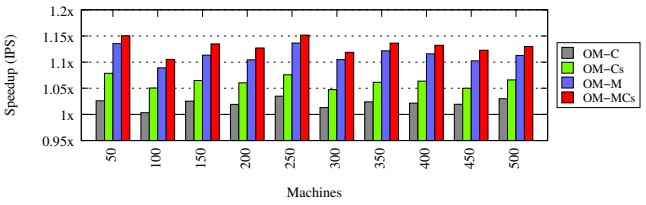

Fig. 3. Performance speedup (IPS) from Opportunistic Mapping

To investigate the potential performance improvement gained by exploiting the heterogeneity in a given WSC, we perform opportunistic mapping in both our experimental testbed and production datacenters. To measure performance, we use the aggregate *instructions per second* (IPS) of the entire cluster. IPS is used as it is *micro*architecture independent.

[Experimental Testbed] We conducted experiments on a range of WSC sizes, from 50 machines to 500 machines. The composition of machines in each WSC includes an evenly matched quantity of each of the three machines shown in Table 1. The workload run on each cluster is composed of twice as many jobs as there are machines. These jobs are randomly selected (uniform distribution) from the SPEC 2006 benchmarks. This scenario is representative of how jobs are run in production datacenters as typically only one or two major jobs are allocated to a given machine. It is important to note that while each SPEC job is a single threaded workload, as we have shown in Section 2, we observe a similar performance variability from the microarchitectural and co-location heterogeneity as Google's commercial workloads. The current job placement approach in production datacenters treats all machines as homogeneous. Except for some special rules applied to certain job types, a job can be placed on any machine in a single cluster. This approach is used as our baseline.

Figure 3 shows the performance improvement achieved when using each of the map scoring policies shown in Table 3. Each bar shows the average IPS of the cluster using various scoring policies normalized to the IPS of random job placement with the same workload. The amount of performance benefit remains fairly stable as we increase the size of the datacenter. The small variation in performance improvement across different datacenter sizes is due to the fact that the baseline for each cluster of bars is a random mapping. In the figure, we observe that scoring policies that only consider co-location heterogeneity (OM-C, OM-Cs) are quite effective, generating up to an 8% improvement over

random mapping. This is impressive considering that less than half of the 22 SPEC benchmark applications used are sensitive to performance interference when co-located. On the other hand, only considering microarchitectural heterogeneity without considering co-location (OM-M) can produce higher performance benefit over the random mapping, 12% on average. When the opportunistic mapper combines both machine heterogeneity and co-location penalty heterogeneity (OM-MCs), the average performance improvement is increased to almost 16%.

TABLE 4
Number of Machine Types in Production Datacenters

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
4	3	2	3	2	3	2	5	2	2

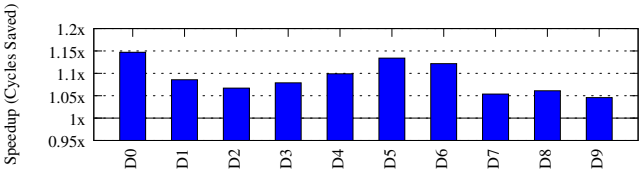

Fig. 4. Calculated performance improvement from opportunistic mapping over a 1 month period of time

[Production Datacenter] Table 4 shows the amount of platform diversity found in 10 randomly selected anonymized Google datacenters in operation. Figure 4 shows the calculated performance improvement from a heterogeneity-aware mapping of applications in 10 of Googles active WSCs. We performed this experiment by using our opportunistic mapping algorithm on real GWP information to calculate the changes in IPS when mapped to various machines in the datacenter. Although major applications are already mapped to their best platform through manual assignment, we have measured significant potential improvement of up to 15% when using opportunistic mapping to place the remaining jobs.

[Opportunity Factor] Figure 5 presents the performance improvement at the application level from the scenario with 500 machines (1000 jobs) shown in Figure 3. The y-axis shows each application type's average speedup using the opportunistic mapper, normalized to each type's average performance in a random mapping. This figure demonstrates that application types have varying amounts of performance benefit from the opportunistic mapping. For example, while there is a 16% performance improvement overall, `lbm`, a benchmark that is sensitive to both microarchitectural and co-location heterogeneity, achieves a 70% performance improvement over random mapping. There are also applications that suffer performance degradation. However, as shown in the figure, this effect is minimal. Figure 6 presents the opportunity factor (OF) of each application, calculated using Equation 1 and Equation 2 in Section 4. As Figures 5 and 6 show, OF correctly predicts the top applications that benefit from the opportunistic mapper including `lbm`, `soplex`, `sphinx` and `mcf`, etc. Remember that mapping to exploit datacenter heterogeneity is a constraint optimization problem. As a result not all applications can be mapped to their individual optimal situations and thus the performance potential of some applications, such as `perlbench` and `xalanbmk`, are not fully realized. However, as we are using a global optimization approach, the biggest winners are prioritized.

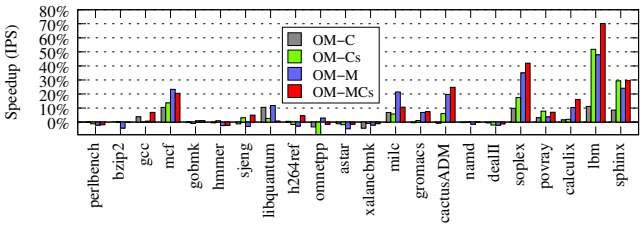

Fig. 5. Speedup at the application level.

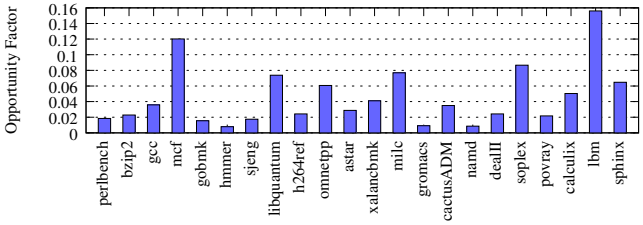

Fig. 6. Opportunity factor of each application.

5 Conclusion

In this work, we investigate microarchitectural heterogeneity in the datacenter and find that even when considering platforms from competing generations, there is a significant performance opportunity when acknowledging the heterogeneity in "homogeneous" WSCs. We also present *opportunistic mapping* and a new metric, an application's *opportunity factor*. Using opportunistic mapping, applications that are sensitive to heterogeneity have a performance improvement of up to 70%. Overall, opportunistic mapping improves the performance of an entire cluster by 16% as shown with our experimental testbed and 15% in real production datacenters as shown by our postmortem analysis.

References

[1] L. A. Barroso, J. Dean, and U. Hölzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2):22 – 28, 2003.

[2] L. A. Barroso and U. Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, pages 1–120, Sep 2009.

[3] L. A. Barroso and P. Ranganathan. Guest editors' introduction: Datacenter-scale computing. *IEEE Micro*, 30:6–7, 2010.

[4] S. Bykov, A. Geller, G. Kliot, J. Larus, R. Pandya, and J. Thelin. Orleans: A framework for cloud computing. Technical Report MSR-TR-2010-159, Microsoft Research, November 2010.

[5] C. Kozyrakis, A. Kansal, S. Sankar, and K. Vaid. Server engineering insights for large-scale online services. In *Micro, IEEE*, volume 30, pages 8–19. IEEE, 2010.

[6] K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, and S. Reinhardt. Understanding and designing new server architectures for emerging warehouse-computing environments. ISCA '08, pages 315–326, Washington, DC, USA, 2008. IEEE Computer Society.

[7] K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, and S. Reinhardt. Server designs for warehouse-computing environments. In *Micro, IEEE*, volume 29, pages 41–49. IEEE, 2009.

[8] J. Mars, N. Vachharajani, R. Hundt, and M. L. Soffa. Contention aware execution: online contention detection and response. In *CGO '10*, pages 257–265, New York, NY, USA, 2010. ACM.

[9] R. Nathuji, C. Isci, and E. Gorbatov. Exploiting platform heterogeneity for power efficient data centers. *Auto. Comp., Inter. Conf. on*, 0:5, 2007.

[10] G. Ren, T. Moseley, E. Tune, S. Rus, and R. Hundt. Google-wide profiling: A continuous profiling infrastructure for datacenters. *IEEE Micro*, 2010.

[11] S. M. Sait and H. Youssef. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1999.

[12] L. Tang, J. Mars, N. Vachharajani, R. Hundt, and M. L. Soffa. The impact of memory subsystem resource sharing on datacenter applications. In *ISCA '11*, New York, NY, USA, 2011. ACM.

[13] J. A. Winter and D. H. Albonesi. Scheduling algorithms for unpredictably heterogeneous cmp architectures. *DSN*, 2008.