The Netflix Challenge: Datacenter Edition

Christina Delimitrou and Christos Kozyrakis
Stanford University
{cdel, kozyraki}@stanford.edu

Abstract—The hundreds of thousands of servers in modern warehouse-scale systems make performance and efficiency optimizations pressing design challenges. These systems are traditionally considered homogeneous. However, that is not typically the case. Multiple server generations compose a heterogeneous environment, whose performance opportunities have not been fully explored since techniques that account for platform heterogeneity typically do not scale to the tens of thousands of applications hosted in large-scale cloud providers. We present *ADSM*, a scalable and efficient recommendation system for *application-to-server mapping* in large-scale datacenters (DCs) that is QoS-aware. ADSM overcomes the drawbacks of previous techniques, by leveraging robust and computationally efficient analytical methods to scale to tens of thousands of applications with minimal overheads. It is also *QoS-aware*, mapping applications to platforms while enforcing strict QoS guarantees. ADSM is derived from validated analytical models, has low and bounded prediction errors, is simple to implement and scales to thousands of applications without significant changes to the system. Over 390 real DC workloads, ADSM improves performance by 16% on average and up to 2.5x and efficiency by 22% in a DC with 10 different server configurations.

Index Terms—Super (very large) computers, Heterogeneous (hybrid) systems, Scheduling and task partitioning, Application studies resulting in better multiple-processor systems.

1 Introduction

Warehouse-scale systems now provide the compute and storage infrastructure for most popular online services [2], making their performance and power optimization critical design challenges. In this work, we examine one aspect of DC architectures which until recently has gone mostly unnoticed and has significant performance and energy potential. DCs have traditionally been embraced as homogeneous platforms [1]. However, as machines get replaced over the 15-year provisioned deployment lifetime [2], [9], [12], they introduce some inherent heterogeneity in the system. This heterogeneity is at the platform level, differs from the one found in CMPs, and ignoring it can lead to significant inefficiencies.

Previous work has quantified the performance potential from sophisticated placement of workloads to heterogeneous machines [9], [10], [12]. However, the proposed schemes rely mainly on empirical observations, require detailed profiling and do not scale beyond a small number of applications. This makes them unsuitable for *virtualized environments* that host thousands of new applications every day (e.g., EC2 [5], Azure [15], Google AppEngine [6], or vMotion). Additionally, these schemes focus on performance, not energy savings. As a result, applications are mapped to servers with limited heterogeneity considerations. Going forward, heterogeneity-aware workload mapping is critical, and doing so in a computationally efficient manner is a strict constraint for scalability.

We present *Application to Datacenter Server Mapping* (ADSM), a scalable and efficient scheme for application-to-server mapping that addresses the limitations of previous proposals. ADSM is derived from *validated, computationally efficient analytical methods*, not merely empirical observations, which allow it to scale to tens of thousands of servers and applications, improving system performance and efficiency, while enforcing strict per-application QoS guarantees, marginal overheads and tight error bounds. It is designed as a lightweight controller that resides with the cluster scheduler and requires minimal changes to the system. ADSM identifies similarities between incoming and known applications and makes efficient recommendations of application placement to heterogeneous servers. Additionally, because the recommendation system is driven by analytical methods we can explicitly compute its complexity and guarantee marginal training and decision overheads. ADSM is based on a recommendation system similar to the one deployed as part of the Netflix Challenge [3], with users replaced by applications and movies by server configurations.

We evaluate ADSM for a wide range of applications: single-threaded, multi-threaded and multiprogrammed standard benchmark suites and 390 *real DC workloads* from Microsoft, including latency-critical (e.g., Search) and computationally-intensive applications. We use two 40-machine clusters; a "homogeneous" production cluster, with ten server configurations and a heterogeneous cluster with ten high-end and low-power machine types.

First, we quantify the mapping algorithm's benefits and overheads. ADSM improves performance by 22% and energy efficiency by 18% on average in the "homogeneous" cluster, over a heterogeneity-oblivious mapping scheme. The results are more striking in the heterogeneous cluster. ADSM not only does not degrade performance when mapping applications to the low-power systems, but improves it by 17% while reducing energy by 24% on average. ADSM, thus, motivates more radical heterogeneity in DCs, showing its efficiency potential, while enforcing strict QoS guarantees. Second, we validate the accuracy of the analytical methods that drive ADSM. Finally, we perform a sensitivity study on configuration parameters, such as training set size and time and strictness of QoS guarantees.

. Manuscript submitted: 26-Apr-2012. Manuscript accepted: 20-May-2012. Final manuscript received: 23-May-2012.

Posted to the IEEE & CSDL on 5/31/2012
DOI 10.1109/L-CA.2012.10 1556-6056/12/$31.00 © 2012 Published by the IEEE Computer Society

2 ADSM Techniques and Design

2.1 Overview

ADSM is an application-to-server mapping scheme that is scalable and efficient. It works for single-threaded, multi-threaded and multiprogrammed applications. It is aimed for large-scale virtualized environments, which receive thousands of new applications each day, and run them over tens of thousands of servers.

ADSM does not require extensive profiling across different server platforms. Instead, it leverages the similarities between virtualized applications to rank server configurations and predict the one that will benefit an application the most. This allows ADSM to provide accurate recommendations for large-scale application mappings.

ADSM's mapping scheme is derived from analytical methods rather than empirical observations. This means that it achieves *provable, strong guarantees* on the accuracy of the predictions, low and tight error bounds and marginal training and decision overheads, while being accurate and scalable. The following sections provide the mathematical background on the concepts used in ADSM and a description of the scoring functions and the mapping scheme.

2.2 Mathematical Background

Singular Value Decomposition: Singular value decomposition or SVD is a matrix factorization method used among others, for dimensionality reduction and similarity identification. For example, SVD is applied in recommendation systems to identify similarities between users and items [3]. Let A be the matrix containing the scores users (rows) assign to items (columns); input to SVD. Factoring A produces the decomposition to matrices U, V and Σ.

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} = U \cdot \Sigma \cdot V^T$$

where the matrices of left and right singular vectors are:

$$U_{m\times r} = \begin{pmatrix} u_{11} & \cdots & u_{1r} \\ u_{21} & \cdots & u_{2r} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mr} \end{pmatrix}, V_{n\times r} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{r1} & v_{r2} & \cdots & v_{rn} \end{pmatrix}$$

and $\Sigma_{r\times r} = \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{pmatrix}$, the diagonal matrix of singular values. r is the rank of matrix A.

Similarity concepts are represented by singular values σ_i and the confidence in a similarity concept by the magnitude of the corresponding singular value. The U matrix represents the strength of the row-to-concept similarity, while the V matrix the concept-to-column similarity. The complexity of SVD on a $m \times n$ matrix is $min(n^2m, m^2n)$.

PQ Reconstruction with Stochastic Gradient Descent: To map applications to servers, we need the scores in matrix A. We use PQ-reconstruction and building from SVD we have: $Q_{m\times r} = U$ and $P^T_{r\times n} = \Sigma \cdot V^T$. The product of Q and P^T gives the approximation matrix R with the score predictions. If R had no missing entries, we would be done. However, ADSM performs minimal application profiling to reduce overheads, leaving missing entries in A which propagate through Q and P^T to R. To reconstruct the full utility matrix, we use Stochastic Gradient Descent (SGD), a scalable and lightweight latent factor model that iteratively recreates A using the following process until convergence: $\forall r_{ui}$, where r_{ui} an element of the reconstructed matrix R

$\epsilon_{ui} = r_{ui} - q_i \cdot {p_u}^T$
$q_i \leftarrow q_i + \eta(\epsilon_{ui} p_u - \lambda q_i)$
$p_u \leftarrow p_u + \eta(\epsilon_{ui} q_i - \lambda p_u)$

where η the learning rate and λ the regularization factor. Convergence is achieved when $|\epsilon|_{L_2} = \sqrt{\sum_{u,i} |\epsilon_{ui}|^2}$ becomes marginal.

2.3 Scoring Functions

Matrix A is populated with application performance scores. Different applications, however, should be characterized by different scores, e.g., IPC cannot be used blindly to capture perfomance of multithreaded workloads. Here we describe scoring functions for each application type.

- *Single-threaded workloads:* Applications such as SPEC CPU2006 have no synchronization primitives, therefore we can use IPC as the scoring function. Execution time is also possible, however it requires running to completion which would increase ADSM's training overheads. Comparison between the two exhibited minimal deviations.

- *Multiprogrammed workloads:* Similarly, we use aggregate IPC across the multiprogrammed mix. Although this will select the platform that maximizes aggregate IPC, on its own it does not guarantee per-application QoS. To provide such fairness guarantees, we additionally enforce no performance degradation for each application in the mix and verify this through per-mix *hmean* of speedup.

- *Multi-threaded workloads:* Here IPC is not the straightforward metric [14]. Locking schemes, especially spin-locks, can taint the instruction count, masking idle waits as high performance. We address this by periodically polling low-overhead performance counters to detect changes in the register file and weight-out of the IPC computation such execution segments. We have verified that mappings using this "useful" IPC are almost identical to mappings produced using execution time as the scoring function.

- *DC workloads:* Large-scale cloud providers typically host multi-threaded applications, therefore the default scoring function for this workload type is "useful" IPC. Even for other workloads using "useful" IPC will yield the same benefits as using raw IPC. The differentiation exists to eliminate the overhead of spin-lock detection when the workload type is known. In general, when workloads are not known, or multiple workload types are present we use "useful" IPC to drive the analytical methods of ADSM.

This categorization assumes that we focus on performance optimization. The same principles apply for energy (IPC/W) or cost efficiency optimization ($IPC/(W \cdot \$)$).

2.4 Mapping Scheme

ADSM works by assigning per-application scores to different server platforms based on the metric we optimize for.

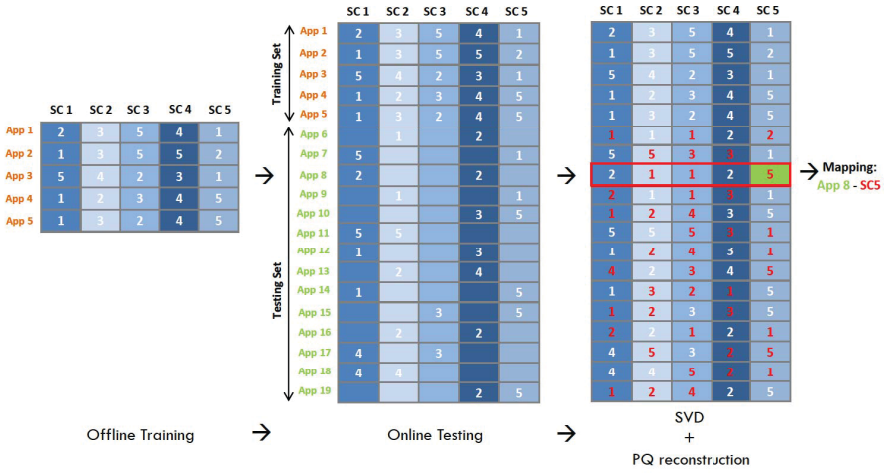

Fig. 1. Offline, online phases of ADSM and reconstruction of utility matrix with PQ reconstruction using SGD.

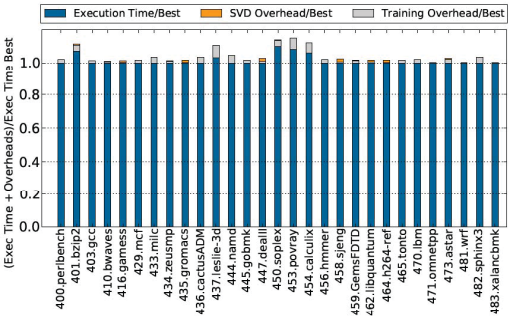

Fig. 2. Execution time breakdown for ADSM over the 29 SPEC CPU2006 workloads.

It is divided in an *offline* and an *online* mode. In the offline mode, it selects a small number of applications ($<$ 5-10%) and profiles them on all server configurations. In the online mode, ADSM provides application-to-server mapping recommendations. When a new application arrives, ADSM samples its behavior on two server platforms (to satisfy SVD's sparsity constraints), assigns scores and uses SVD to identify similarities between known and new applications and predict missing scores in the sparse matrix A. The rows in A represent applications and the columns server configurations (SC). U captures the application to similarity concept strength and V the SC to similarity concept strength, while Σ the strength of each similarity concept. As more applications are added in A the training set increases and the recommendation accuracy improves. Fig. 1 shows a walk through the steps of mapping applications to servers for a simplified scenario. Scores have a 1-5 granularity and the training set consists of the first 5 applications. For simplicity, we assume integer scoring values; in a real experiment, the entries of matrix A can have real values.

The core idea behind ADSM is that applications with similar characteristics will have similar behaviors on the same platforms. For example, applications that are bound by memory capacity are likely to benefit from a SC with large memory. This means that the singular value for the corresponding similarity concept of memory-bound applications will be large. Obviously, two applications can be similar in one characteristic but different in others, especially when scaling to large application spaces. This is where SVD helps by uncovering hidden similarities and filtering out the ones less likely to have an impact on the application's behavior. Identifying such similarity concepts and evaluating their strength enables ADSM to perform efficient application-to-server mappings.

3 Evaluation

We use a collection of single-threaded (SPEC CPU2006), multithreaded (PARSEC [4], SPLASH-2 [16], BioParallel [8], Minebench [11]), multiprogrammed (350 mixes of 4 SPEC workloads, based on the methodology described in [13]) and finally 390 DC workloads from Microsoft.

A. Evaluation of ADSM

Performance/Energy impact: We briefly present the results for the "homogeneous" DC and focus on the findings for the heterogeneous system.

- *Homogeneous DC:* ADSM improves performance by 19% for single-threaded, 17% for multi-threaded, 24% for multi-programmed and 22% for DC workloads over random and is within 3.8% from an oracle policy that always chooses the best platform for an application. Similarly, given the scoring function for efficiency optimization it achieves 18% energy savings, while providing QoS guarantees.
- *Heterogeneous DC:* Fig.3 shows the performance benefits on the multiprogrammed (18%), multi-threaded (17%) and DC workload suites (16%). The data in Fig. 3a and 3c are shown from worst to best workload, and the speedup is measured over the performance of a random assignment. ADSM finds the optimal platform for 85-90% of workloads on average and for the remaining workloads it finds a platform within 5-10% of the best one, and is within 4.6% from the oracle mapping scheme.

Overheads: Fig. 2 is the breakdown of execution time for the 29 SPEC applications and highlights two results; first, the ratio of the time that corresponds to useful computation over execution time in the optimal platform is very close to 1 and second, training and decision overheads are on average 0.1% and 2.0% respectively.

Validation of analytical methods: Fig. 3 also shows the comparison between predictions from the analytical models (red dots) and performance measured in the corresponding platform. On average the deviation is less than 3.8%, guaranteeing the accuracy of the methods driving ADSM. We have additionally verified the accuracy of the utility matrix reconstruction through Pearson correlation coefficients for problems of smaller scale and have observed less than 5% deviation between the two methods.

B. Sensitivity Study

Sensitivity to training set: Fig. 4a, 5a show the sensitivity of ADSM to training set sizes for SPEC and DC workloads. The error is defined as the difference in performance between optimal and recommended platform. For SPEC

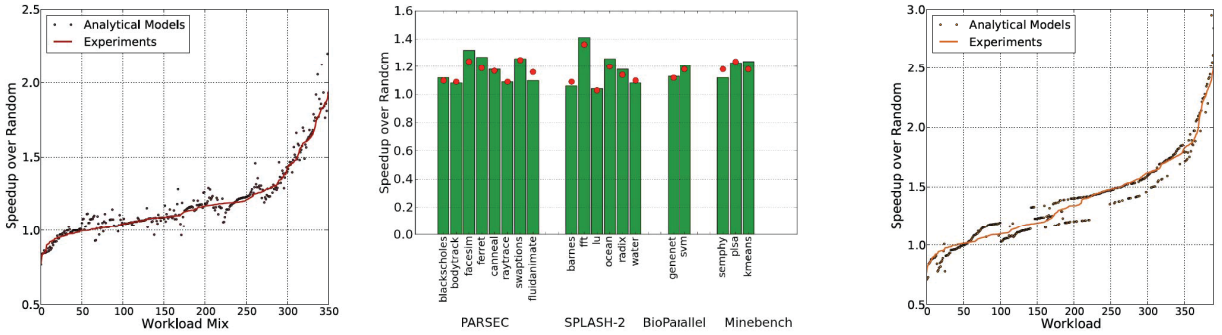

Fig. 3. Performance benefits from using ADSM on multiprogrammed (Fig. 3a), multi-threaded (Fig. 3b) and DC workloads (Fig. 3c).

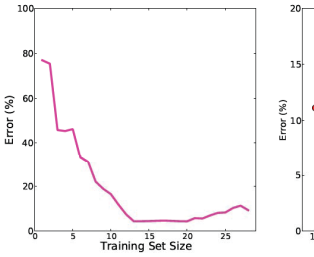

Fig. 4. Sensitivity to training set size and training time for the SPEC workloads.

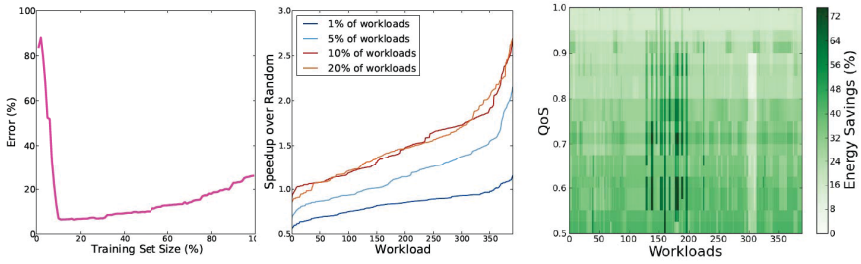

Fig. 5. Sensitivity to training set size for DC applications: (a) accuracy and (b) performance benefits. Fig. 5c: Sensitivity of energy savings to QoS guarantees.

the error decreases for larger training sets and is minimized for 13 applications. Applications in the training set are chosen randomly, but consistently across experiments. The error increases after 13 applications, due to overfitting. DC applications exhibit a more radical decrease in error. For sizes 1% and 5% errors are high, but converge to constant and low values as the training set reaches 10%, and do not improve significantly after that, while for large sizes (over 50%) there is overfitting. This change is reflected in performance benefits as well, with speedups converging to maximum at 10% (Fig. 5b). Comparing SPEC and DC workloads shows that ADSM requires certain workloads to extract similarity features, however their fraction becomes negligible as the application space grows.

Sensitivity to training time: Fig. 4b shows the accuracy of ADSM when the training time changes from 1M to 40B instructions, after a 40B instrs warmup period, for the SPEC benchmarks. The error is high for short training times (1-400M instrs) and converges to a constant value less than 5% at 1B instrs. Longer training times capture transient workload phases and improve mapping decisions. The results are consistent for the DC applications as well.

Sensitivity to QoS strictness: Previously we described that given appropriate scoring functions ADSM can optimize for efficiency while maintaining performance requirements. Here, we additionally examine how these energy benefits increase as QoS restrictions are relaxed. Fig. 5c shows the savings for DC applications, when performance degradation is tolerable. This affects significantly applications which can now be mapped to low-power platforms (x-axis:135-200), while others need high-performance systems irrespective of QoS (x-axis: 300-315). Finally, most applications improve due to ADSM allocating fewer threads or mapping them to a slightly more efficient platform. For 80% QoS there is a 38% energy improvement.

4 Conclusions

We presented ADSM a *scalable* and *efficient* scheme for application-to-server mapping that is QoS and heterogeneity-aware. ADSM scales to thousands of applications with minimal training and decision overheads, and because it leverages robust analytical methods it can guarantee upper-bounded and small errors. We used ADSM on a wide collection of workloads and showed significant performance gains. Additionally, when ADSM is used for energy efficiency optimizations it yields important energy savings while enforcing strict QoS guarantees.

References

[1] L. A. Barroso. "Warehouse-Scale Computing: Entering the Teenage Decade". ISCA Keynote, SJ, June 2011.
[2] L. A. Barroso, U. Holzle. "The Datacenter as a Computer". Synthesis Series on Computer Architecture, May 2009.
[3] R. M. Bell. Y. Koren, C. Volinsky. "The BellKor 2008 Solution to the Netflix Prize". Technical report, AT&T Labs - Research, October 2007.
[4] C. Bienia, S. Kumar, et al. "The PARSEC benchmark suite: Characterization and architectural implications". *In Proc. of PACT, 2008.*
[5] Amazon Elastic Compute Cloud-EC2. http://aws.amazon.com/ec2/
[6] Google AppEngine. http://code.google.com/appengine/
[7] J.R. Hamilton. "Cost of Power in Large-Scale Data Centers". TR, 2008.
[8] A. Jaleel, M. Mattina, B. Jacob. "Last Level Cache (LLC) Performance of Data Mining Workloads On a CMP - A Case Study of Parallel Bioinformatics Workloads". *In Proc. of 12th HPCA, TX, 2006.*
[9] J.Mars, L.Tang et al. "Heterogeneity in "Homogeneous" Warehouse-Scale Computers: A Performance Opportunity". *In IEEE CAL, 2011.*
[10] J. Mars, L. Tang, et al. "Bubble-Up: Increasing Utilization in Modern Warehouse Scale Computers via Sensible Co-locations". *In Proc. of MICRO-44, Sao Paolo, 2011*
[11] R. Narayanan, B. Ozisikyilmaz, et al. "MineBench: A Benchmark Suite for DataMining Workloads ". *In Proc. of IISWC, CA, 2006.*
[12] R. Nathuji, C. Isci, and E. Gorbatov. "Exploiting platform heterogeneity for power efficient data centers". *In Proc. of ICAC'07, FL, 2007.*
[13] Vantage: Scalable and Efficient Fine-Grain Cache Partitioning. Daniel Sanchez, Christos Kozyrakis. *In Proc. of the 38th ISCA, CA, 2011.*
[14] T. Wenisch, R. Wunderlich et al. "SimFlex: Statistical Sampling of Computer System Simulation." In IEEE Micro, July 2006.
[15] Windows Azure. http://www.windowsazure.com/
[16] S. Woo, M. Ohara, et al. The SPLASH-2 Programs: Characterization and Methodological Considerations. *In Proc. of the 22nd ISCA, 1995.*