

Model-Driven Placement of Compute Tasks and Data in a Networked Utility

Piyush Shivam, Adriana Iamnitchi, Aydan R. Yumerefendi, Jeffrey S. Chase

Department of Computer Science
Durham NC 27708

E-mail: {shivam, anda, aydan, chase}@cs.duke.edu

Abstract

An important problem in resource management for networked resource-sharing systems is the simultaneous allocation of multiple resources to an application. Self-optimizing systems must co-allocate resources in a way that reconciles competing demands and maximizes global system objectives under dynamic conditions. We propose a simple model-driven approach to estimate the performance of a candidate assignment of resources, and select the best candidate to meet local or global goals. In this work, we address the placement of batch compute tasks and data in a network of compute and storage sites. We use the model to select placements for a set of synthetic benchmarks and a functional MRI processing application. Our experiments show that the model predicts the performance of candidate assignments within 10% of the empirical values.

1 Introduction and Overview

Self-optimizing systems must assign resources in a way that reconciles competing demands and maximizes global system objectives. We explore the use of application models to evaluate candidate assignments, as a key element of a comprehensive approach to autonomic resource allocation. We focus on an important instance of the general problem: placement of tasks and data in a network of compute and storage sites—an on-demand networked utility.

Placement is crucial in large-scale utilities (e.g., grids, networked data centers etc.) because the system must balance the effect of placement choices on migration costs, access costs, and the amount of resource (e.g., CPU cycles and disk arms) available at each candidate location. Utilities can and will provide mechanisms to realize a range of placement choices for data staging and task placement as shown in Figure 1. BAD-FS [1] is one recent example that explores mechanisms to coordinate task placement and data storage. Previous studies show that placement choices in such environments can have a substantial impact on performance (e.g., [2]) for data-intensive computational applications.

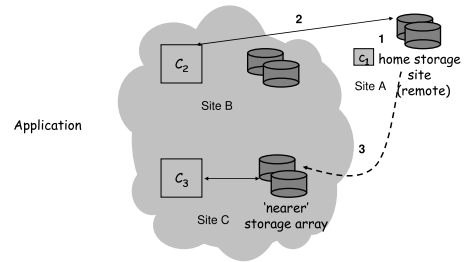


Figure 1. Three choices for placing a data-intensive task in a wide-area utility: (1) place the compute tasks next to data (site A); (2) run the task at a remote site and access data remotely (site B); (3) run the task remotely and stage the data to the remote site (site C).

The goal of our work is to provide a foundation for policy agents to select among competing alternatives. We focus on batch applications whose execution time is limited only by the properties of the resources assigned to them. Thus the work is complementary to other works that use queuing models to drive dynamic provisioning of Internet services under changing load.

In our approach, a policy agent estimates the completion time that would result from each candidate choice, based on two inputs: a *hardware element profile* relative to a reference hardware defining the candidate bundle, and an *application profile* that approximates how the application uses its resources. Each profile is represented as a vector of parameters derived from measurements. The hardware profile captures known properties of the hardware such as CPU clock speed, network latency and bandwidth, and properties of storage arrays. The application profile is independently obtained from observations of the application executing on a set of hardware bundles.

We propose and evaluate simple approximate models based on asymptotic bounds analysis. These models are compact and efficient, and our experimental results show that even primitive formulations can be sufficiently accu-

rate to guide placement. The models predict the value of a resource bundle in terms of the performance it offers to the hosted application or *guest*. The agent can combine this prediction with measures of the value of that performance given the relative importance of the guest and/or the economic value associated with meeting (or failing to meet) contractual obligations to the guest [4]. In essence, this combination results in a utility function for each customer across the space of candidate bundles. A number of systems exist that approximate the NP-hard optimal assignments if this information is given as input.

For example, a policy agent for a customer of an out-sourced storage provider [6] might use our model-driven techniques to determine the cost-effectiveness of different service levels offered by the provider, given knowledge of how various customer applications use the data. We are conducting our research in the context of SHARP [3], which provides safe mechanisms to delegate control of resources to such policy agents in federated utilities. We assume that the utility’s resources are virtualized, so that a policy agent can reserve and assign specific resource shares to each hosted application (e.g., [5]).

2 Model

We model applications as data flow pipelines with multiple stages. A *stage* is a function, e.g., computation or I/O, that is mapped to some hardware element. We characterize the service demand of a stage i by its *work occupancy* per unit of data on a reference hardware element of the same type—the average rate at which the work of the stage ‘occupies’ the hardware, a_i . A stage could be busy doing useful work on the data or stalling, i.e., waiting for another stage to produce or consume data. The stall creates bubbles in the pipeline that ‘occupy’ the stage hardware in a manner similar to the actual work. We find it useful to characterize this as an occupancy measure. The rate at which the stall occupies stage i is denoted by its *stall occupancy* s_i .

A legal candidate bundle for an application pipeline consists of a vector of the hardware element profiles, one for each stage in the pipeline. The work occupancy at each stage depends on the underlying hardware element for that stage. The general pipeline model gives the stall occupancy of a stage from the work occupancy measures of all the stages. The application profile consists of: (a) a_i for each stage i ; (b) a function f_i that estimates how the a_i scales with change in the underlying hardware element for that stage.

In this work, we focus on a two-stage pipeline: the compute stage, and the I/O stage. The candidate bundle for this two-stage pipeline consists of the profiles of compute hardware and the storage hardware elements. The question is *how much* of each resource to provision and *where* (Figure 1). The scaling function f_i captures the how much fac-

tor for the compute and storage hardware. The latency of the storage is in the element profile of the storage hardware.

We *evaluate* a candidate bundle i by predicting the throughput attained by the application mapped to this bundle as $T_i = \max_i(1/(a_i f_i + s_i))$. For batch applications the averaged data throughput predicts completion time when the volume of data to be processed is also known. Similarly, for a desired throughput T , we can *specify* the relation between the compute and storage hardware profiles of a candidate bundle i that meet that target by solving $a_i f_i + s_i \leq 1/T$.

3 Model Use and Validation

We use the model to choose the ‘best’ assignment for a set of sample applications. The candidate bundles consist of compute nodes that differ in compute speed, and storage devices that differ in the network latency from the compute nodes. First we parameterize the model by running it on a set of *training set* bundles, and then use it to predict the performance of *test* bundles to decide the ‘best’ choice.

The applications include five synthetic benchmarks generated as a sequence of compute stage and a storage stage with varying access patterns, and a functional MRI statistical parametric mapping application. We also use our model to explain and predict the results obtained in the storage outsourcing study done by Wee Teck Ng *et al.* [6].

We find that model-driven assignment has a significantly better performance than random placement of tasks and data. Moreover, the model is reasonably accurate (within 10%) in predicting the completion time on a candidate assignment. On the applications we study the throughput is bursty, but completion time is given by the average throughput and is independent of the bursts.

References

- [1] J. Bent, D. Thain, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny. Explicit Control in a Batch-Aware Distributed File System. In *Symposium on Networked Systems Design and Implementation (NSDI)*, March 2004.
- [2] W. Elwasif, J. Plank, and R. Wolski. Data Staging Effects in Wide Area Task Farming Applications. In *IEEE International Symposium on Cluster Computing and the grid, Brisbane, Australia*, pp. 122-129, May 2001.
- [3] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An Architecture for Secure Resource Peering. In *The 19th ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.
- [4] D. Irwin, L. Grit, and J. Chase. Balancing Risk and Reward in a Market-based Task Service. In *13th IEEE Symposium on High Performance Distributed Computing*, June 2004.
- [5] W. Jin, J. S. Chase, and J. Kaur. Interposed Proportional Sharing for a Storage Service Utility. In *Conference on Measurement and Modeling of Computer Systems*, June 2004.
- [6] W. T. Ng, B. Hillyer, E. Shriver, E. Gabber, and B. Ozden. Obtaining High Performance for Storage Outsourcing. In *Conference on File and Storage Technologies (FAST)*, January 2002.