Modern data-intensive services are now composed of large-scale, complex components, applications, and systems with different types of behavior, responsibility, and purpose due to the heterogeneity of the workloads handled by these services. For example, it is common for a Web content provider to have data-cycle workloads, such as behavioral advertisement targeting, and content personalization [2]. Consider a data-cycle workload that displays and provide personalized content to users in real-time. Aside from ensuring content requests are served in real-time, continuous streams of user-generated data and event logs are also processed and analyzed to build and update per-user content personalization models.

Currently, there is not a single system that handles and performs well for any types of workloads. For example, Hadoop MapReduce works well for batch data analytics workloads but does poorly for real-time serving of data. Simply using Hadoop for everything can cause performance problems. Rather, a careful mix and match of appropriate systems is needed to support diverse workloads as required by the data-intensive service, which is evident by the systems and components used by modern Web services [9]. For instance, LinkedIn uses a wide variety of systems such as Hadoop, Kafka, RDBMS, and Voldemort to handle their workloads.

The complexity and the number of components and systems that comprise these services poses several challenges for the management of workloads of said services. First, this leads to a lot of manual and time-consuming work for system administrators in maintaining these services. Careful tuning is required for each component and system. One approach for managing workloads on a particular system may not be applicable for other systems in these services. Moreover, the interactions and dependencies between choices for tuning and management of these systems can quickly lead to a large plan space. Finally, due to the number of complex systems that comprise these services, it is possible that any of these systems can execute a given workload, but intelligently deciding how to execute the workload and picking the most suitable system can be a non-trivial decision. Thus, my research interests lie in the automated management of workloads and resources of modern data-intensive services, such that the burden to use and maintain these services are completely taken off users.

To this end, my research work for my PhD dissertation and some of my outside work (i.e., internship) has been related to the above challenges. Specifically, I have done work on (1) designing and implementing the policy and mechanism for dynamically provisioning stateless applications; (2) designing and implementing a controller for allocating storage nodes; (3) designing an application-aware power budgeting system for virtualized data centers; (4) designing an automated optimizer for MapReduce workflows; and (5) building a middleware layer that can pick the optimal execution strategy within and across systems for executing windowed aggregation queries.

1. Dynamic Elasticity for Stateless Applications

Modern services usually have front-end components that directly interact with users. For example, a Web service may have a cluster of Web servers that delivers web content to user requests (e.g., displaying web pages to users). In these applications, the performance of the workload are often quantified by the latency to process user requests. Thus, managing
workloads for these components entails ensuring that the latency are maintained at an acceptable level as specified by the service level objectives. One approach administrators take is to ensure that the applications have enough resources to handle the amount of user requests. In the case of clustered systems, this involves ensuring that the overall size of the cluster can handle the incoming workload.

In this work [8], I designed and built a feedback controller that can dynamically provision cluster nodes based on user demands. One challenge in provisioning cluster nodes is that the control granularity is discrete, especially if the nodes are hosted on the cloud. For example, cloud providers, such as Amazon EC2, discretize resources into a small range of predefined sizes (e.g., you can request $n$ more $x$ type of nodes), which can lead to instability or oscillations when directly applying controllers that assume fine-grained access to resources. In order to address this challenge, I introduced proportional thresholding, a policy enhancement for feedback controllers that enables stable control across a wide range of cluster sizes using the coarse-grained control offered by popular virtual compute cloud services. This policy modifies an integral control by using a dynamic target range, instead of a single target value. Moreover, the dynamic target range decreases as the accumulated actuator values increases, which leads to an effective and efficient controller.

2. Automated Control for Elastic Storage

A natural extension to dynamic provisioning of stateless applications is to support stateful applications. In particular, modern services usually have a back-end distributed storage system that is responsible for storing and serving data, such as user-generated contents. Similarly, managing storage workload to ensure acceptable latency can be handled through dynamic provisioning of storage nodes.

However, stateful applications present a number of distinct challenges. First, it requires data rebalancing because adding a new node does not give immediate performance improvements due to the lack of data to serve from the new node. Moreover, moving or copying data to new nodes consumes resources that can otherwise be used to serve user requests and can temporarily further degrade the performance. I have designed and implemented a controller that take these challenges into account [7]. The controller is composed of three components: a Horizontal Scale Controller (HSC), responsible for growing and shrinking the number of storage nodes, a Data Rebalance Controller (DRC), responsible for controlling the data transfers to rebalance the storage tier after it grows or shrinks, and a state machine, responsible for coordinating the actions of the HSC and the DRC.

3. VPS: Virtualized Power Shifting

The costs of power to run a data center can be large. Thus, over-subscription is employed to maximize the utilization of provisioned power capacity of a data center, which means that the sum of the possible peak power consumptions of all the servers combined is greater than the provisioned power capacity. In order to ensure that over-subscription is safe and efficient, power budgeting methods are employed to ensure that actual consumption never exceeds capacity. However, current power budgeting methods enforce capacity limits in hardware and are not well suited for virtualized servers because the hardware is shared among multiple applications.

During my internship at Microsoft Research, we developed a power budgeting solution called virtualized power shifting (VPS) [6]. VPS is a distributed hierarchical control system
that follows the layout of applications hosted in the data center. It respects application priorities and application VM boundaries for allocating power budgets. Given a power budget, it also optimizes application performance by dynamically adjusting power proportions and exploiting multiple power control knobs, such as DVFS and VM CPU time.

4. Stubby: A Transformation-based Optimizer for MapReduce Workflows

There is a growing trend of performing analytics on large datasets using workflows of jobs. MapReduce is a popular platform for running workflows. Studies have shown that the gap in performance can be quite large between optimized and unoptimized workflows. However, building an optimizer for MapReduce workflows presents a number of challenges. i) The software ecosystem around MapReduce is growing rapidly with different interfaces for generating workflows. ii) Due to the flexibility of MapReduce, the information needed for optimization may not always be available. iii) There is a large plan space for finding the optimal workflow.

I am part of the research group that worked on a self-tuning system for big data analytics, called Starfish [3]. As part of the Starfish project, I developed Stubby [5] to address the previously mentioned challenges. Stubby can automatically optimize MapReduce workflows, without any need for users or administrators to understand the internals of the workflows and various knobs available in the MapReduce system. Stubby is an external optimizer that sits above the MapReduce execution engine and below any software system (interfaces) for generating workflows. It uses a transformation-based approach to optimization, which allows for easy extension and customization of functionality. It also uses optional annotations as medium for interfaces to convey information to Stubby. Finally, it is able to identify non-interacting subspaces to deal with the challenge of large optimization plan space.

5. Cyclops: A Middleware for Management and Orchestration of Systems for Workload Execution

Currently, in modern data-intensive services that comprise of a number of systems and components, administrators and developers have to carefully orchestrate the applications (and queries) that run on each of these systems to handle the overall workload. Since a lot of these systems can run the same type of queries but with varying degree of tradeoffs (e.g., latency, throughput, fault-tolerance), a lot of manual work is required not only for deciding the best strategy and system to execute the query, but also for implementing the queries for each system because of the differences in the exposed abstractions.

Cyclops [4] is an on-going research work to address the above challenges. As a first cut, we focus on windowed aggregation queries. We provide users with a language abstraction for declaring queries encapsulated from the underlying systems. Cyclops is a middleware that can map a query to the most suitable system. Our approach is to use a black-box approach that estimates the performance of the query on each system. As a bootstrap mechanism, Cyclops initially runs a pre-defined set of queries that serves as training data to build models representing each system. Given an input query from users, it then uses these models to predict, recommend, and run the query on the most suitable system.

Other Research Work

Aside from my research work related to automated management and provisioning of resources, I am also interested in distributed systems, mobile systems, and cloud comput-
ing. Specifically, I also collaborated with another PhD student to work on the Vis-à-Vis project [10, 1], which is outside of the scope of my dissertation work. In this project, we designed and implemented a privacy-preserving architecture that leverages on virtual individual servers in the cloud, for location-based online social networks. Moreover, during my internship at Microsoft Research Silicon Valley, I was also involved in a research project that quantified and characterized the performance loss and variability of computational workloads in the cloud.

References


