Chapter 16

Data Partitioning for Highly Scalable Cloud Applications

Robert Neumann  
Otto-von-Guericke-Universität Magdeburg, Germany

Matthias Baumann  
Otto-von-Guericke-Universität Magdeburg, Germany

Reiner Dumke  
Otto-von-Guericke-Universität Magdeburg, Germany

Andreas Schmietendorf  
Berlin School of Economics and Law, Germany

ABSTRACT

Cloud computing has brought new challenges, but also exciting chances to developers. With the illusion of an infinite expanse of computing resources, even individual developers have been put into a position from which they can create applications that scale out all over the world, thus affecting millions of people. One difficulty with developing such mega-scale Cloud applications is to keep the storage backend scalable. In this chapter, we detail ways of partitioning non-relational data among thousands of physical storage nodes, thereby emphasizing the peculiarities of tabular Cloud storage. The authors give recommendations of how to establish a sustainable and scaling data management architecture that – while growing in terms of data volume – still provides the same high throughput. Finally, they underline their theoretical elaborations by featuring insights won from a real-world cloud project with which the authors have been involved.

INTRODUCTION

The idea of the Cloud to have access to unlimited resources and only pay for what is really used (Armbrust et al., 2009), is inspiring visionaries worldwide to come up with applications that are bringing together millions of people. While in pre-Cloud times, applications, such as Facebook, Twitter or YouTube, required enormous upfront investments in capital in order to establish the necessary infrastructure, the Cloud has changed things such that nowadays everybody can build the next “big thing”. Mega-scale applications typically
have an impact on the lives of millions of people on a global scale and—due to the enormous value they provide to their users—affect people from various countries and cultures. In 2009, our team began exploring what is known as the Public Cloud, a development platform that provisions resources on demand. This consisted then and still consists of the three big players Amazon (EC2), Google (App Engine) and Microsoft (Azure). We found that all three of these are quite different in structure and in terms of the services they offer. Due to the Public Cloud’s advanced developer experience and its smooth integration with concepts known from outside the Cloud, we have emphasized our research efforts on Microsoft’s Windows Azure Cloud platform. Nevertheless, many concepts presented in this chapter are generally applicable or can easily be transported to other platforms too.

After discussing the various types of Cloud storage available on the Windows Azure platform, we will engineer a highly scalable partitioning architecture for a Cloud storage backend. We will emphasize the Do’s and Don’ts of querying Windows Azure Table Storage and will provide guidelines for how to ensure a consistently high throughput, especially in times when your application grows in terms of users. The second part of this chapter details intermediate techniques of ensuring that your storage backend constantly delivers high performance. In particular, we cover non-relational secondary indexes, location-aware data partitioning, Cloud caching as well as content delivery networks (CDNs). Throughout the entire chapter, our elaborations will be based on the practical experiences we have gained with a real-world Cloud application that we have been developing.

All in all, this chapter aims to contribute to the efforts of those architects and engineers who want to create high-performance, mega-scale Cloud applications with the potential of having an impact on the lives of millions of people all around the world.

**CLOUD COMPUTING**

The economics of Cloud Computing have been discussed by several researchers and practitioners. Essentially, the three core benefits are (Armbrust et al, 2009):

1. Infinite computing resources available on demand
2. No up-front commitment by Cloud users
3. Pay for computing resources as needed

For application developers, the elimination of an up-front commitment and the ability to dynamically adapt the used resources to the system load have offered them the possibility to give formerly risky ideas a chance. If one such idea turned out to be successful, it would be inevitable to have designed the application in such a way, that it would be able to accomodate a quickly increasing load. Keeping an application scalable not only means keeping its business logic, but also its data logic scalable.

Mega-scale applications, such as Facebook, Flickr, YouTube or Yelp present new challenges for storing and querying large amounts of data. SQL databases have dominated application development for decades as their ACID characteristics have especially benefited data integrity and safety. Performance and throughput, on the other hand, are aspects that have been suffering from ACID, a fact that was no longer acceptable for mega-scale application engineers. The advent of light-weight and non-relational key-value storages, such as Apache Cassandra (originally developed by Facebook) or membase, has facilitated the shift from ACID to BASE, which stands for “Basically Available, Soft-state, Eventually consistent”. Though BASE’s eventual consistency characteristic complicates data integrity, at the same time it allows for enhanced concurrency and scale-out – traits that are crucial when serving millions of queries in less than a second (e.g., Facebook: 13,000,000 queries per second).