Big Data Challenges in Application Performance Management

Tilmann Rabl and Hans-Arno Jacobsen
Middleware Systems Research Group
University of Toronto
Toronto, Canada
{tilmann,arno}@msrg.utoronto.ca

Serge Mankovskii
CA Labs
Toronto, Canada
serge.mankovskii@ca.com

ABSTRACT

Many web companies deal with enormous data sizes and request rates beyond the capabilities of traditional database systems. This has led to the development of modern Web Data Platforms (WDPs). WDPs handle large amounts of data and activity through massively distributed infrastructures. To achieve performance and availability at Internet scale, WDPs restrict querying capability, and provide weaker consistency guarantees than traditional ACID transactions. The reduced functionality is sufficient for many web applications. High data and query rates also appear in application performance management (APM). APM has similar requirements like current web based information systems such as weaker consistency needs, geographical distribution and asynchronous processing. At the same time, APM has some unique features and requirements that make previously published research and existing architectures inapplicable.

1. INTRODUCTION

Achieving the right trade-off between different properties of big data stores largely depends on the use case and the requirements of the industrial application. Existing big data solutions were created within large web companies for handling web applications. The needs of big data for enterprise information technology management (EITM) are different. An example of an enterprise system architecture can be seen in Figure 1. In such a system each component will be augmented with an agent that can monitor its performance and trace processed transactions. EITM monitoring infrastructures can generate tens of millions of measurements every minute at constant rate using Java byte code instrumentation (added in JSR-163 [8]). During critical situations these rates can escalate even higher due to cascading of problems triggered by earlier problems. Every measurement can contain a vital piece of information needed to triage and identify the source of the problem. Hence, every piece of information has to be reliably retained and analyzed. We are not aware of any big data solutions suitable for managing, governing and securing mission critical IT infrastructures.

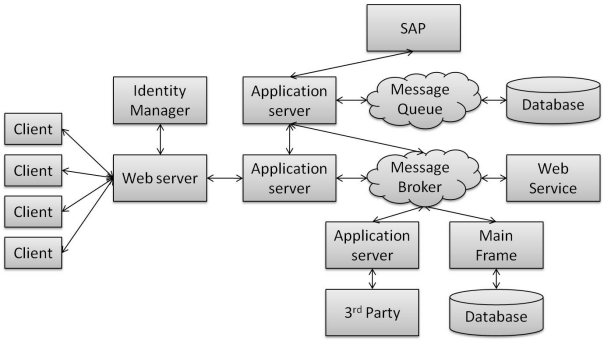

Figure 1: Example Enterprise System Architecture

Management of IT infrastructures has a unique blend of requirements distinct which, however, not dissimilar to the requirements of large web application companies that created dominant big data systems of today. The CA Tech line of application performance management products is a typical example of EITM specific data storage requirements. The CA APM products require high availability for retaining every single measurement of managed IT infrastructure at sustained arrival rate of tens of millions of measurements per minute. The underlying data store needs to work across multiple geographically distributed data centers as well as provide high read access speeds needed for analytic processing of collected measurements. This data store has availability requirements similar to Amazon Domino / Cassandra [7, 9], it has to work across multiple data centers and feature real-time querying like Yahoo PNUTS [4], furthermore it should provide rich support for analytic processing found in Hadoop / MapReduce [6].

In this abstract, we present a brief overview of application performance management. We will give details on APM's specific requirements for data storage and retrieval and propose a specialized architecture for web data platforms for APM.

2. APPLICATION PERFORMANCE MANAGEMENT

Application performance management refers to the monitoring and management of enterprise software systems. An example of an enterprise system is shown in Figure 1. There are different ways to manage this kind of architecture. A common approach is the ARM standard [1]. In this approach

every component has to implement the ARM API that is available for C and Java. Prominent ARM instrumented applications are the Apache HTTP server and IBM DB2. Although several common enterprise software systems are already ARM enabled it is often not feasible to implement the ARM API in existing systems. Another approach that is applicable for Java based systems is byte code instrumentation. This is enabled by the Java Virtual Machine Tool Interface that was specified in JSR-163 and introduced in J2SE 5.0. Its intention is to present an interface for profiling and debugging. Byte code instrumentation allows to augment Java based software components with agents that have access to the state and the method invocations. This approach that is used in the CA APM products enables monitoring components, tracing transactions, and root cause analysis without changing the code base of the monitored system. The monitoring can be done on a very high level of detail. Current systems can generate millions of measurements per second where each data point may contain important information and therefore has to be stored safely. To fully understand the state of the monitored systems thousands of queries have to be processed on the measurements per second. Due to these enormous insert and query rates the underlying data store has to be highly scalable. To support the necessary insert rates current systems often store the measurement information in flat files on disk and process them asynchronously and in batch. The introduction of a integrated data store would allow more sophisticated data analysis like trend analysis that current systems cannot offer.

3. PROPOSED ARCHITECTURE

The biggest challenge in the APM use case is to support the enormous write rates. These require the system to be especially designed for scalable write performance at a level unusual in current web data platforms. The YCSB write-heavy test case has a write ratio of 50% [5], while APM may have a write ratio of 99%. In Figure 2 an overview of the proposed architecture can be seen. Core of the architecture is a distributed, asynchronous key-value store. Since queries are usually known in advance we propose an approach that uses materialized views. This enables a timely answer of queries. The views are maintained asynchronously by sophisticated view managers.

Efficient view maintenance is vital for the integration and selective distribution of large volumes of data across wide-area networks and administrative boundaries such data warehouses [2], parallel and distributed databases [10] and WDPs [3, 7, 4, 9]. In these environments, view maintenance can minimize the amount of data transferred across the network, reduce query latency for queries over data exposed as views, implement secondary indexes, and provide notification and alerting services. Moreover, especially in wide-area networked environments, views can serve to conform to privacy regulations that disallow certain data from leaving specific geographic regions, while requiring other data to be exposed for availability, performance, and operational reasons.

This massive scale, the inherent parallelism, the asynchronous update processing, and the lack of many traditional database mechanisms available in WDP impose several challenges not addressed by existing approaches.

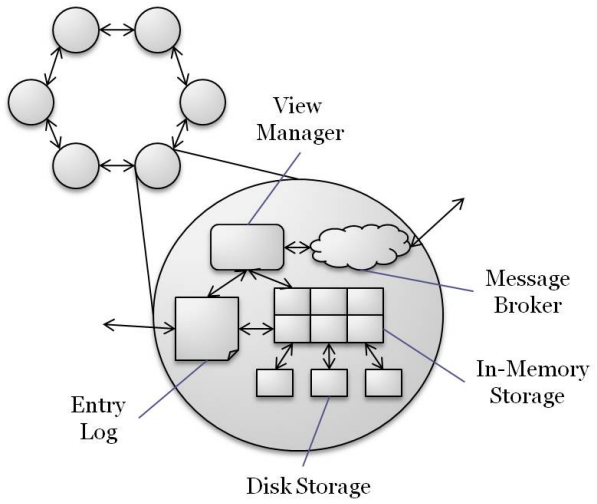

Figure 2: Asynchronous Data Store Architecture

4. REFERENCES

[1] Application response measurement (arm) issue 4.1 v1. http://www.opengroup.org/management/arm/, 2007.

[2] D. Agrawal, A. E. Abbadi, A. K. Singh, and T. Yurek. Efficient view maintenance at data warehouses. In *SIGMOD '97: Proceedings ACM SIGMOD International Conference on Management of Data*, pages 417–427, 1997.

[3] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *OSDI '06: 7th Symposium on Operating Systems Design and Implementation*, pages 205–218, 2006.

[4] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. Pnuts: Yahoo!'s hosted data serving platform. *Proceedings of the VLDB Endowment*, 1(2):1277–1288, 2008.

[5] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *SoCC '10: Proceedings of the 1st ACM Symposium on Cloud Computing*, pages 143–154, 2010.

[6] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[7] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *SOSP '07: Proceedings of the 21st ACM Symposium on Operating Systems Principles*, pages 205–220, 2007.

[8] R. Field. Java virtual machine profiling interface specification (jsr-163). http://jcp.org/en/jsr/summary?id=163, 2004.

[9] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[10] G. Luo, J. F. Naughton, C. J. Ellmann, and M. Watzke. A comparison of three methods for join view maintenance in parallel rdbms. In *ICDE '03: Proceedings of the 19th International Conference on Data Engineering*, 2003.