Thoth: Towards Managing a Multi-System Cluster

Mayuresh Kunjir
Duke University
mayuresh@cs.duke.edu

Prajakta Kalmegh
Duke University
pkalmegh@cs.duke.edu

Shivnath Babu
Duke University
shivnath@cs.duke.edu

ABSTRACT

Following the 'no one size fits all' philosophy, active research in big data platforms is focusing on creating an environment for multiple 'one-size' systems to co-exist and cooperate in the same cluster. Consequently, it has now become imperative to provide an *integrated management* solution that provides a database-centric view of the underlying multi-system environment. We outline the proposal of DBMS$^+$, a database management platform over multiple 'one-size' systems. Our prototype implementation of DBMS$^+$, called Thoth, adaptively chooses a *best-fit* system based on application requirements. In this demonstration, we propose to showcase Thoth DM, a data management framework for Thoth which consists of a data collection pipeline utility, data consolidation and dispatcher module, and a warehouse for storing this data. We further introduce the notion of ***apps***; an ***app*** is a utility that registers with Thoth DM and interfaces with its warehouse to provide core database management functionalities like dynamic provisioning of resources, designing a multi-system-aware optimizer, tuning of configuration parameters on each system, data storage, and layout schemes.

We will demonstrate Thoth DM in action over Hive, Hadoop, Shark, Spark, and the Hadoop Distributed File System. This demonstration will focus on the following ***apps***: (i) Dashboard for administration and control that will let the audience monitor and visualize a database-centric view of the multi-system cluster, and (ii) Data Layout Recommender ***app*** will allow searching for the optimal data layout in the multi-system setting.

1. INTRODUCTION

A data-driven enterprise today needs systems for advanced computations like clickstream log analysis, getting real-time insights into streaming data, business forecasting, and near-interactive experience for large volumes of data, etc. Organizations like Facebook, Netflix, Zynga, LinkedIn, and Yahoo are adopting many 'one-size' systems that cater to these requirements. Different 'one-size' systems serve different processing needs such as batch processing, bulk-synchronous MPI processing, key-value stores, in-memory analytics, graph computations, and columnar storages.

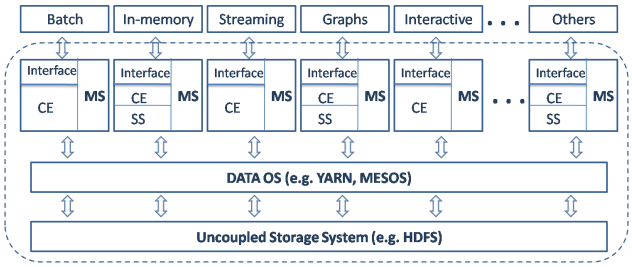

Figure 1: Data OS Multi-System Management

Following the 'no one size fits all' philosophy, active research in big data platforms is focussing on creating an environment for multiple 'one-size' systems to co-exist and cooperate in the same cluster. Sharing a cluster across multiple systems is desired in order to save on huge data migration costs involved in dataflow pipelines. Some of the solutions existing today include Apache Yarn [11], Mesos [7], Facebook's Corona, and Google's Omega [10]. The approach taken by all these systems is to provide a management layer capable of managing resource allocation across systems. We call this layer *Data OS* since it is functionally closer to an operating system for data. Figure 1 shows a broad architectural view of these systems.

However, OS functionalities are incapable of managing database-specific needs of systems. The 'no one size fits all' philosophy poses many challenges for system administrators and application developers. They are faced with the challenge of finding a system most suitable for their applications since each system may serve many overlapping classes of problems. For instance, log analysis can be performed on parallel databases like Teradata/Greenplum, batch systems like Hadoop, column-oriented systems for efficient OLAP analysis like Vertica, or in-memory analytics with systems like SAP HANA. An application developer is burdened with the task of choosing an *appropriate* system based on resource availability and changing application requirements. This entails considerable manual effort in tuning each system individually based on available resources.

A management layer on top with a database-centric view of systems can use a learning-based approach to recommend the right system to use for an application. This is a functionality of database management systems that will be hard to

Proceedings of the VLDB Endowment, Vol. 7, No. 13

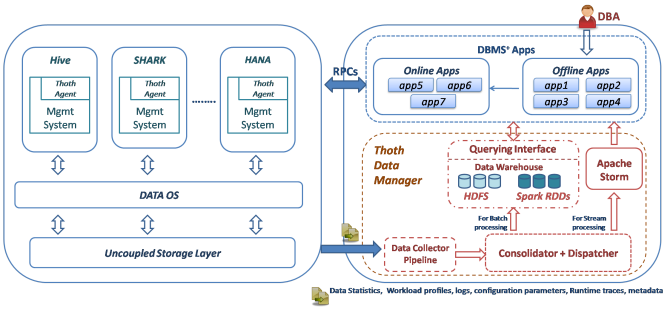

Figure 2: Thoth Multi-System Management

provide with a purely OS-centric approach. In [8], we made a case for building a unified DBMS$^+$ system in a shared-cluster environment that understands an application's requirements and finds the *best fit* system for executing an application. To think of *database-as-a-service* is more convenient for system administrators and application developers for better resource provisioning and data layout. This also relieves them of significant effort in tuning each system separately, allocating resources, and most importantly choosing a system most suitable for an application.

In this work, we introduce a prototype for DBMS$^+$: Thoth. Thoth provides a platform to develop various multi-system manageability applications (called *apps* in short). The applications could range from a dashboard to monitor running applications to a multi-system optimizer. We first list down some of the major research challenges that drive our vision:

- **Database-centric view:** Create a view over multi-system cluster by integrating monitoring data from all the systems and all the layers of software stack for users to be able to easily track and understand performance of their applications without a need to understand internals of each system.
- **Auto-tuning:** Build a multi-system aware optimizer capable of picking right system for execution along with settings for its tuning knobs.
- **Dynamic Resource Provisioning:** Find a best resource allocation strategy across systems by studying historical workflow traces. Use it to allocate resources on-the-fly to meet applications' demands.
- **Data Layout Management:** Provide recommendations on data format, storage engine, partitioning and materialization based on usage patterns of workloads.

Contributions

- Proposal of a multi-system management architecture: DBMS$^+$ and design of a prototype: Thoth. Thoth provides a data driven platform for various cluster management *apps*.
- Data management utility for Thoth: Thoth DM. It provides a unified view over data collected from multiple systems running on cluster for *apps* to make system management decisions.
- A dataflow visualization *app* and a data layout management *app* developed in Thoth. While dataflow visualizer allows real-time monitoring of applications, data layout manager processes data profiled by Thoth DM to recommend changes to data layout.

Outline. The rest of the paper is organized as follows: Section 2 describes how *apps* in Thoth contribute towards building a multi-system management layer. It also illustrates the methodology for collection and management of data. Section 3 provides a proposal to demonstrate a dataflow visualization app and a data-layout recommender app to the VLDB audience.

2. MULTI-SYSTEM MANAGEMENT PLATFORM

We propose a prototype of DBMS$^+$ architecture, a multi-system management platform, called Thoth. Let's first look at some of the important design goals of Thoth.

1. **Extensible:** Thoth should provide an extensible platform for development of manageability features that cannot be provided by *Data OS*.
2. **Data-driven:** Thoth should enable data-driven decisions which means an ability to collect and process data generated by systems is required.
3. **Real-time:** Thoth should support real-time and frequent decision making.
4. **Online and offline modes:** Some of the manageability decisions are more of recommendations to administrators while some other may change state of systems. Thoth should support both modes of functioning.
5. **Co-operative development:** Thoth should interact with the systems and *Data OS* layers through well-defined APIs so that Thoth *apps* can be developed side-by-side as the systems and *Data OS* evolve.

2.1 Thoth Apps

Thoth provides a platform for development of various multi-system management applications, *apps* in short. These *apps* help Thoth achieve easy extensibility. Figure 1 shows the Thoth platform; the backbone of Thoth is a data manager which enables data-driven decision making. Thoth DM collects data in the form of system logs, runtime statistics, resource profiles, metadata and configuration options from multi-system cluster. It consolidates and stores the data in a warehouse through which it is made available to *apps*. We discuss the Thoth DM design in Section 2.2.

DBMS$^+$ *apps* register themselves with Thoth DM to receive a subset of instrumented data. The *apps* can be of two types: (a) `offline` *apps*; and (b) `online` *apps*. The `offline` *apps* analyze data to supply recommendations on system design or system monitoring data to database administrators. DBAs can use this information to better design applications or to better tune systems. Since these *apps* do not take any action on systems by themselves, they are termed `offline` *apps*. The `online` *apps*, on the other hand, are capable of executing an action that may change system design. To invoke these actions, we add *agents* inside systems. The `online` *apps* communicate with the agents using a RPC mechanism. This allows for development *apps* without intruding system space much, one of the foremost goals of DBMS$^+$ design.

We list down a few small *apps* to showcase potential of DBMS$^+$ platform. A discussion on building a fully functional multi-system optimizer prototype is left out of the scope of this demonstration proposal. In section 3, we give a demo plan for *app1*, *app2* and *app5* from the below list.

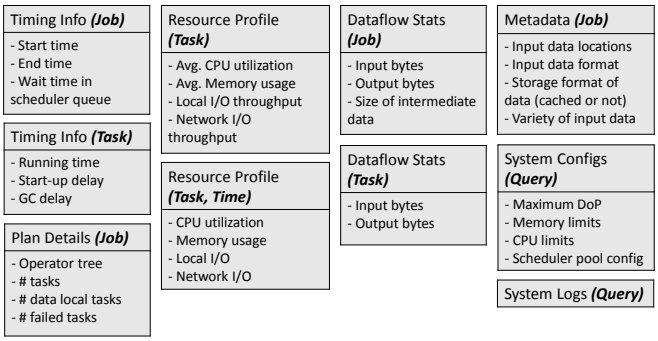

Figure 3: Categorization of Profiled Data

- `offline` *apps*

  app1 Dashboard for administration and monitoring a.k.a. Database-Centric View

  app2 Data layout recommender

  app3 Multi-tenant resource allocation recommender

  app4 Prediction for system tuning parameters

- `online` *apps*

  app5 Data layout auto-tuner

  app6 Multi-tenant resource allocation auto-tuner

  app7 Cluster Configuration Parameters auto-tuner

2.2 Thoth Data Manager

Thoth DM is a single entry point for all application logs and profile data. Its functionalities include: (a) consolidating data to a system-agnostic form; (b) maintaining a warehouse for data; and (c) supplying relevant data to various *apps* of DBMS$^+$. Key challenges in Thoth DM are the choice of data collection granularity and a need of system-agnostic traces. We describe how Thoth DM addresses these challenges next.

Query execution on any distributed system can be imagined as being carried out in three levels – query, job, and task. A query is executed as a multi-stage process. Each stage, alternatively called a job, is split into multiple tasks distributed across cluster. We collect data at all three granularities. The data includes configuration settings, metadata, runtime statistics and profiles. We use lightweight agents for system instrumentation keeping in tune with our goal of building Thoth manageability functionalities without intruding systems space much. Thoth DM consolidates the collected data into multiple categories shown in Figure 3. This allows for optimizing storage and supplying only the relevant data to *apps*.

Figure 2 shows Thoth DM flow. The data is generated by various events in life-cycle of queries. It is collected by data collector module of Thoth DM and passed on to a data consolidator, which is implemented as an 'interceptor' in Apache Flume. Once consolidated, data can be dispatched to consumers. A consumer can either be a streaming *app* or Thoth data warehouse. Real-time processing *apps* require data to be sent over a stream in Apache Storm. The *apps* doing a batch processing, on the other hand, require data to be stored in data warehouse. Thoth DM's supports data to be stored either on HDFS or in Spark RDDs as per *app* requirements. To extract required information from data warehouse, *apps* use a querying interface provided by Thoth DM.

Data stored in warehouse exhibits a large variety; for example query logs are highly unstructured, resource profiles and other statistics adhere to a fixed structure, and query plans have hierarchical/nested structure. Thoth *apps* may favor one storage system over other based on the type of data they need. This points to an interesting research challenge in finding right data layout for Thoth data warehouse which we plan to address in future.

2.3 Other Approaches

Apache Chukwa [9] is a data collection system that enables monitoring a distributed system. Its scope is limited to large scale log collection on Hadoop HDFS. It also provides a reporting and monitoring framework for near real-time analysis of the cluster. Our solution of Thoth DM enables but is not limited to log collection on HDFS. Another key difference is that Thoth DM allows streaming of data to Apache Storm for efficient real-time processing of events. Netflix's adoptation of Chukwa in Suro [1] provides a support for not only arbitrary data formats but also real-time processing of data. Suro provides central data management for Netflix applications running in different clusters. Our vision, however, is to contribute a database management layer for a multi-system cluster.

Splunk [4] provides a real-time platform for proactive monitoring of large volumes of machine data. It collects data from various sources like logfiles, messages, config files, tickets, messages, etc and makes it available for further analysis. Splunk also promotes the notion of apps and categorizes them into application management, business analytics, etc. thus enabling more advanced analytics. Our solution is motivated from Splunk but highly differs in the vision; we aim at providing an integrated management solution for a multi-system environment. Our prototype of Thoth DM collects and consolidates data relevant for provisioning of core DBMS functionalities in the form of *apps*.

3. DEMONSTRATION PLAN

The purpose of this demonstration is to get vision of DBMS$^+$ across to community. We want to achieve this by showcasing our prototype Thoth. Thoth provides a rich platform to develop various management *apps* with a database-centric view of underlying systems. It is made possible by Thoth DM utility that collects data (statistics, profiles, configurations, etc.), consolidates, and supplies it in real-time to Thoth *apps*.

A demo cluster is set up on Amazon EC2 cloud with two systems: (a) Hive over Hadoop, and (b) Shark over Spark. Two Thoth *apps* are developed. First *app* processes data in real-time and streams it to a dashboard which supports monitoring execution flow and understanding performance. The second *app* analyzes historical workload traces to recommend dynamic changes in data layouts.

3.1 Database-Centric View

Many cluster monitoring tools are available to assist DBAs, e.g. Ganglia [3]. These tools have a *system-centric* view of cluster which means they treat database applications like any other processes using hardware resources. Such a narrow view is good enough for many *Data OS* management

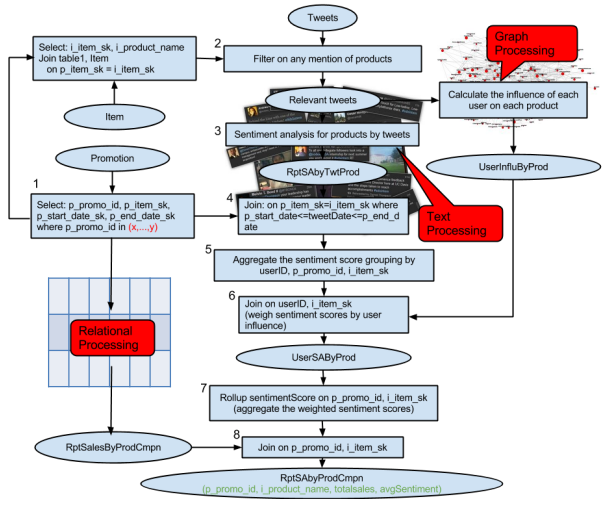

Figure 4: A workflow in BigFrame

decisions such as provisioning resources in case of failures. But for a DBMS$^+$ management layer, what is needed is a *database-centric* view of systems. An understanding of intricacies of database systems is essential for admins to better tune systems or better design applications. e.g. To diagnose a query failure, an understanding of behavior of jobs during execution of the query would help pinpoint whether the failure was due to: a sub-optimal query plan; an inferior data layout; or insufficient resource allocation. This is where Thoth helps with its management of data collected from system instrumentation and logs. Another important contribution of Thoth is to provide a consistent view of multiple systems sharing the cluster. An admin managing a multi-system cluster may not be equipped with expertise of each system but rather interested in high level easy-to-understand information that can assist her find best settings for applications.

We use BigFrame benchmarking service [2] to exhibit Thoth's role in managing a multi-system cluster. BigFrame is an ongoing effort targeted towards creating a *benchmarking-as-a-service* solution for big data analytics. An example application workflow provided by the benchmarking service is shown in Figure 4. The workflow exhibits a variety of use-cases: (a) SQL queries with relational operators; (b) Statistical analysis and machine learning algorithms from NLP applied to tweets to extract user sentiments; and (c) Iterative graph computations to compute user influence that weights their sentiment scores. When presented with such a workflow and with multiple possible paths of execution, not just making a *right* choice of systems for execution but also understanding behavior of systems under such a workload in itself is a big challenge for system admins. We demonstrate how Thoth dashboard ***app*** (***app1***) assists admins on BigFrame workloads.

The dashboard ***app*** receives data from Thoth DM in form of a stream sent over Apache Storm. It builds visualizations that show various features like query plans, resource utilization profiles, dataflow statistics, etc. Each of the visualization includes tuning options to pick a set of statistics to be included, the granularity for summarization, and so on. The demonstration will give a walk-through over these features targeted at managing a BigFrame application workflow.

3.2 Data Layout Recommender

We demonstrate another ***app*** in Thoth that provides workload-aware recommendations for data layout (***app2***). The purpose is to showcase how workload traces can be used in building powerful models for tuning data layouts across multiple systems. According to a study carried out in [6], analytical workloads exhibit interesting temporal computational patterns that call for dynamic changes in system design. We plan to leverage a wide array of literature(e.g. [5]) on building models for data layout recommendations using historical traces. The flow of ***app2*** looks like: Trace is periodically queried from Thoth warehouse; Models are invoked over the trace to get recommendations on data format, materialization, partitioning, etc.; The recommendations are relayed to DBA who can use them in re-structuring of data.

4. REFERENCES

[1] Announcing suro: Backbone of netflix's data pipeline, `http://techblog.netflix.com/2013/12/announcing-suro-backbone-of-netflixs.html`.

[2] Bigframe: Benchmarking service for big data analytics, `https://github.com/bigframeteam/BigFrame`.

[3] Ganglia monitoring system, `http://ganglia.sourceforge.net/`.

[4] Splunk, `http://www.splunk.com/`.

[5] S. Agrawal. Automatic physical design tuning: workload as a sequence. In *In Proceedings of the ACM International Conference on Management of Data (SIGMOD*, pages 683–694. ACM Press, 2006.

[6] Y. Chen, S. Alspaugh, and R. Katz. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *Proc. VLDB Endow.*, 5(12):1802–1813, Aug. 2012.

[7] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 22–22, Berkeley, CA, USA, 2011. USENIX Association.

[8] H. Lim, Y. Han, and S. Babu. How to fit when no one size fits. In *CIDR*, 2013.

[9] A. Rabkin and R. Katz. Chukwa: A system for reliable large-scale log collection. In *Proceedings of the 24th International Conference on Large Installation System Administration*, LISA'10, pages 1–15, Berkeley, CA, USA, 2010. USENIX Association.

[10] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes. Omega: Flexible, scalable schedulers for large compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 351–364, New York, NY, USA, 2013. ACM.

[11] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 5:1–5:16, New York, NY, USA, 2013. ACM.