Poster: MADES - A Multi-Layered, Adaptive, Distributed Event Store

Tilmann Rabl[1], Mohammad Sadoghi[2], Kaiwen Zhang[1], and Hans-Arno Jacobsen[1]
[1]Middleware Systems Research Group, University of Toronto
[2]IBM T.J. Watson Research Center

ABSTRACT

Application performance monitoring (APM) is shifting towards capturing and analyzing every event that arises in an enterprise infrastructure. Current APM systems, for example, make it possible to monitor enterprise applications at the granularity of tracing each method invocation (i.e., an event). Naturally, there is great interest in monitoring these events in real-time to react to system and application failures and in storing the captured information for an extended period of time to enable detailed system analysis, data analytics, and future auditing of trends in the historic data. However, the high insertion-rates (up to millions of events per second) and the purposely limited resource, a small fraction of all enterprise resources (i.e., 1-2% of the overall system resources), dedicated to APM are the key challenges for applying current data management solutions in this context. Emerging distributed key-value stores, often positioned to operate at this scale, induce additional storage overhead when dealing with relatively small data points (e.g., method invocation events) inserted at a rate of millions per second. Thus, they are not a promising solution for such an important class of workloads given APM's highly constrained resource budget. In this paper, to address these shortcomings, we present **M**ulti-layered, **A**daptive, **D**istributed **E**vent **S**tore (MADES): a massively distributed store for collecting, querying, and storing event data at a rate of millions of events per second.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Distributed databases*

Keywords

APM; MADES; event processing; event storage

1. INTRODUCTION

In this paper, we present the **M**ulti-layered, **A**daptive, **D**istributed **E**vent **S**tore (MADES): a massively distributed store for collecting, querying, and storing event data. MADES is specialized in storing monitoring data as produced by APM systems. Our target workload consists of small periodically reported data points. In order to support the necessary data rates, we propose a hierarchical architecture of short-term in-memory stores (*on-line stores*) that cluster and aggregate the incoming data and use compression and bulk insertion to significantly reduce the load on the long-term storage (*historic store*). MADES employs push-based communication (*publish/subscribe paradigm*) (PS), in which the on-line stores' data is pushed periodically to the historic store as well as to other on-line stores. Data movement among the on-line stores achieves not only implicit in-memory replication of data, but also improves efficiency for evaluating continuous queries in which data from multiple sources must be aggregated and joined together over a given sliding window. Moreover, our multi-layered on-line store organization scheme enables each on-line store to adapt to its load by pushing down the excess load to other on-line stores in the lower-layer and ultimately to the historic store. Finally, our proposed architecture naturally provides the elasticity aspect demanded in cloud infrastructures because as new (virtual) machines are added (or removed) from the enterprise infrastructure, a new lightweight instance of the on-line store is also loaded on the newly added machine to collect execution data from the local monitoring agents.

In other words, MADES multi-layered architecture comprises (1) a lightweight on-line store that is capable of answering the desired continuous queries to monitor the system health and publish the measurement results and (2) a durable historic store for enabling extensive data analytics over historic data. Essentially, the on-line store acts as a distributed (adaptive, replicated) cache, with specialized query and compression capabilities, for the historic store and in this way, MADES's novel on-line store design paves the way to provide real-time health and reduce the load on the historic store through bulk insertions of reordered and compressed data. The on-line store is in-memory only and is deployed close to the data monitoring agent (optimally on the same host), while the historic store is durable and deployed on dedicated machines. Moreover, MADES on-line store must be able to deal with (sudden) node addition and removal which not only happens in case of system failures, but also in scaling out the application, e.g., when additional nodes are employed in times of high loads or during system failover and reboots.

DEBS'13, June 29–July 3, 2013, Arlington, Texas, USA.
ACM 978-1-4503-1758-0/13/06.

2. MADES QUERY LANGUAGE

In the APM use case two classes of (continuous) queries are prevalent: (1) time series queries that retrieve continuous measurements of a specified metric and (2) aggregation queries that computes an aggregation function over multiple time series queries.

Our time series query language can be formalized similar to traditional database SPJG (continuous) queries: selection (σ_c), projection (π), join ($\times$, $\bowtie_c$), and group by (Γ) operators. In fact, we adapt PADRES SQL (PSQL) [2], an expressive SQL-based declarative language for registering continuous queries on event streams over either a time-based or a count-based sliding window model. Essentially the sliding window is a snapshot of an observed finite portion of the event stream.

3. MADES ARCHITECTURE

MADES comprises two main components: the lightweight on-line store and the durable historical store. An architectural overview of MADES is demonstrated in Figure 1. The on-line store nodes are deployed preferably on the production machines directly, where the data is been generated by the monitoring agents. This way the data insertion by the agents is a local function call instead of a remote procedure call or communication over the network. In contrast, the

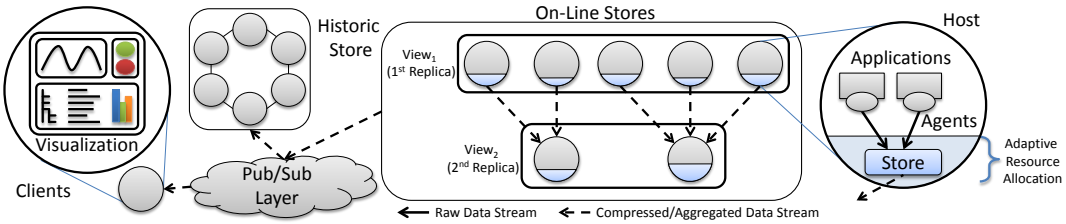


Figure 1: High-Level Overview of MADES Architecture

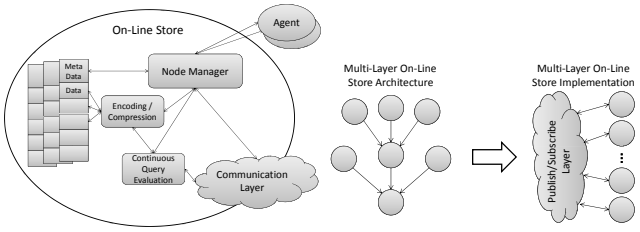
historical store is deployed on dedicated machines and can leverage any modern key-value store. MADES utilizes Cassandra [3], because of its high performance in the APM scenario [5].

An overview of the on-line store can be seen in Figure 2(a). The on-line store is purely based on main-memory in order to keep the overall overhead on the production hosts low. The size of the store is dependent on the capacity of the host and may be changed based on the requirements of the production system. This makes the store highly adaptive within a single machine and elastic across all machines. Either upon reaching the allocated memory limit or periodically, the data is published and subsequently pushed into the historic store. The data that is produced by an agent or pushed from another on-line store (on a higher layer) is re-ordered and stored in a columnar fashion. The columnar storage is specially attractive in the APM context because the data has a highly regular pattern of metric name followed by (aggregated) measurement values. This column-based storage model enables an effective compression scheme, such as run-length encoding, which is essential in order to keep a low memory footprint while maintaining a low processing overhead.

In particular, most metrics tend to have long consecutive sequences of the same value (low entropy) especially during normal operation of the enterprise system, in which resource failures and usage fluctuations account for a small fraction of the entire enterprise infrastructure. Therefore, when operating on data with low entropy, then any simple and fast compression scheme is far more effective. This is evident for measurements that are generated with a higher frequency than the monitored events and for error reporting metrics that often have long sequences of null (or 0) values. Furthermore, both the on-line store as well as the historic store can operate directly on the compressed data. The overall approach has several advantages, not only the memory footprint is highly decreased by the compression, but also due the columnar alignment, huge space savings is made possible because metric names (which account for a large fraction of each tuple's size) do not have to be stored multiple times. The processing of the compressed data results in a faster operation including scanning and filtering while reducing network traffic as well. Furthermore, bulk insertion through compression and data aggregation (i.e., summarization) substantially reduces the insertion rate of the historic store in MADES.

Not all queries are concerned only with the data that is produced on a single machine. Consider the average CPU utilization over a subset of all monitored machines; in order to compute efficiently these *derived metrics*, we direct all relevant data (followed by aggregation of the data) to a single on-line store in the middle-layer of our MADES multi-layered architecture (as shown in Figure 2(b).) These middle-layer on-line stores are selected based on the excess of resources. If no such node is available, then the final aggregation must be processed on the client at the cost of increased processing (and network traffic) on the client-side.

The on-line stores also act as clients to the historic store. All on-line stores periodically push data to the historic store and the clients. Therefore, all communications, e.g., among on-line stores, between on-line and historic stores, and between on-line stores and clients (e.g., system administrator), are abstracted in form of the decoupled and highly distributed PS paradigm. As a result, both clients and historic stores are modeled as subscribers of monitoring data, while the on-line stores are producers of monitored data. Due



(a) On-store Store (b) Communication

Figure 2: Overview of MADES Architecture

to MADES multi-layered on-line store design, each layer of the on-line stores also subscribes to a subset of the on-line stores in the preceding layer (depicted in Figure 2(b).) MADES PS communication layer is built over PADRES [2], an open-source PS system.

Apart from adopting the push-based communication of the PS abstraction, for the purpose of data insertion and data propagation, the on-line stores also comprise notifications and query capabilities. In order to retrieve the most recent history of each metric, the client can directly query the on-line nodes for the latest measurements. Furthermore, clients can subscribe to certain metric or aggregation of metrics. This is beneficial for keeping a client user interface up-to-date as well as enable (partial) data replication for higher availability. The data replication naturally arises because the data is replicated (with a different degree of aggregation) in each layer of MADES. Essentially, each layer in our event store provides a different view of the same raw data. Most important, the problem of view maintenance is much simpler in our context because APM's generated data follows a read- and append-only characteristics. Moreover, since all on-line stores are regularly sending their data to the historical store, the historic store has also a relatively up-to-date view of the of the on-line stores (providing yet another redundant form of data replication but durable.)

4. RELATED WORK

MADES builds upon the well-established query language and semantics in streaming database [2], yet MADES distinguishes itself in fundamental respects from state-of-the-art systems, including Ganglia [4] and Nagios [1]: (1) MADES is optimized (with a highly constrained budget) for the write-dominated, append-only APM workloads with millions of insertions per seconds (2) MADES introduces novel multi-layered, distributed on-line stores which are lightweight in-memory stores to process not only continuous queries, but also to enable many layers of aggregation and compression by orchestration within its multi-layered architecture.

5. REFERENCES

[1] C. Gaspar. Deploying nagios in a large enterprise environment. In *LISA'07*.

[2] H.-A. Jacobsen, V. Muthusamy, and G. Li. The PADRES event processing network: Uniform querying of past and future events. *it - Information Technology'09*.

[3] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *SIGOPS Review'10*.

[4] M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing'04*.

[5] T. Rabl, M. Sadoghi, H.-A. Jacobsen, S. Gómez-Villamor, V. Muntés-Mulero, and S. Mankowskii. Solving Big Data Challenges for Enterprise Application Performance Management. *PVLDB*, 5(12):1724–1735, 2012.