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

Tilmann Rabl
Mohammad Sadoghi
Kaiwen Zhang
Hans-Arno Jacobsen

DEBS Conference 2013
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. To address these shortcomings, we propose Multilayered, Adaptive, Distributed Event Store (MADES): a massively distributed store for collecting, querying, and storing event data at a rate of millions of events per second.
Application Performance Management

- Enterprise system architectures
  - Very complex distributed systems
  - Need of detailed monitoring
  - Service level agreements
- Application performance management
  - How many transactions fail?
  - Where is the root cause of failure?
  - What is the end to end response time?
  - Which component is the bottleneck?
  - Which and how many transactions are there?
Enterprise System Architecture
Java Byte Code Instrumentation

- **JSR – 163**
- **JVM** is augmented with agent
- **Agent** can run additional code
  - No change of code base
  - Trace transactions
  - Measure response times
  - Other types of measurements
- **Huge number of events**
  - Potentially for every method invocation
APM Performance Requirements

• High insert rates
  ▫ Millions inserts / sec

• High query rates
  ▫ Thousands queries / sec

• Write ratio: >99 %

• Agents send data in bulks
  ▫ Different periods (seconds to minutes)
Data Sizes in APM Systems

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Value</th>
<th>Min. Value</th>
<th>Max. Value</th>
<th>Data Points</th>
<th>Start Time (millis)</th>
<th>Stop Time (millis)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontends</td>
<td>3</td>
<td>2.0</td>
<td>4.0</td>
<td>2</td>
<td>131412145 5000</td>
<td>131412145 5000</td>
</tr>
<tr>
<td>ApplicationX:Averageresponse Time (ms)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Nodes in an enterprise architecture
  - 100 – 10K
- Metrics per node
  - Up to 50K, avg 10K
- Reporting period
  - 10 sec avg
- Event rate
  - 1M / sec
- Data size
  - 100B / event
- Raw data
  - 100MB/sec, 355GB/h, 2.8 PB/y
K/V-Store Performance

- Performance evaluation of K/V stores in APM setup
- 99% writes, in-memory
- Published in VLDB’12
  - Rabl et al.: *Solving Big Data Challenges for Enterprise Application Performance Management*. PVLDB 5(12)
MADES Project

• Current system’s performance
  ▫ YCSB results < 15K ops / sec
  ▫ TPC-C results ~ 500K transactions / sec
  ▫ VLDB’12 results ~ 200K ops / sec

• Need for a new architecture
  ▫ Multi-layered Adaptive Distributed Event Store
  ▫ Highly scalable
  ▫ High write throughput
  ▫ Apart from measurements data mostly static
  ▫ Static queries
  ➢ Hybrid key-value store
MADES Architecture

- Lightweight on-line store nodes (short term data)
- Dedicated nodes for historic store (long term data)
- Push and pull based communication
On-line Store Architecture

- Local storage for the agent
- Distributed storage for other on-line stores
- Column-based storage
- Run-length encoding et al.
Logical Node Organization

- MapReduce style aggregation
- In-memory replication
- Pub/Sub realization
Contact

• Tilmann Rabl
• Mohammad Sadoghi
• Kaiwen Zhang
• Hans-Arno Jacobsen