Cloud Monitoring
A challenging Application for Complex Event Processing

Bastian Hoßbach, Bernhard Seeger

Philipps Universität Marburg

ETH Zürich – October 7, 2011
Agenda

- Introduction and motivation
- Design and implementation
- Examples
- Conclusion and research issues
The use of cloud computing leads to
- more flexibility and mobility
- lower costs of IT

BITKOM is expecting
- sales of 8.2 billion euro in 2015 in Germany
- 50 % average growth of sales per year up to 2015

Over 75 % of all enterprises are interested in cloud computing\(^1\)

\(^1\)Symantec Corporation: „Virtualization and Evolution to the Cloud Survey“.
Over 80% of all enterprises aren’t in the cloud\(^2\), because of:
- reliability risks (78%)
- security risks (76%)
- performance risks (76%)
- lack of monitoring and management tools (63%)

ENISA has worked out 35 critical risks and recommends:
- the use of public clouds only for non-critical data
- the permanent holding ready of an exit strategy

\(^2\)Stratecast Research Group: „Overcoming Obstacles to Cloud Computing“. 
Challenges in Cloud Monitoring

- Monitoring in real-time
- Dynamic scalability
- Holisitic monitoring
- Flexibility and extensibility
- Powerful and custom analysis (inter-layer monitoring)
- Control of single cloud components
- Prediction of trends (proactive monitoring)
Mass of sensor data has to be processed in real-time
- 1 mio. physical and 60 mio. virtual machines (OpenCloud)
- 120 PB storage cluster with 200,000 disks (IBM)

Mass of monitoring rules has to be executed continuously
- Customers have individual SLAs
- Each SLA results in custom monitoring rules
Logical view on sensor streams:
Optimization of real sensor streams

1. Bundles of single events and complete streams (less messages)
2. Compression on each bundle (lower size of messages)
Benefits of a SQL-based CEP-Engine

- Inter-layer analysis (depends on the data model)
- Full declarative power of SQL + pattern matching
  - Easy creation/modification of continuous queries at runtime
  - High performance through continuous query optimization
- Solid base for advanced data stream mining techniques
  - Pattern detection
  - Period detection
  - Anomaly detection
  - Frequent itemsets
  - Classification
  - Trends
  - Evolutions
  - ...

This makes a big difference to current monitoring tools!
Example 1: Monitoring the IaaS-Layer (1/3)

CPU Usage in %
- User: 50.00%
- System: 50.00%
- Total: 50.00%

Memory Usage in %
- 86.20%

Network Traffic in KB/s
Example 1: Monitoring the IaaS-Layer (2/3)

**CPU Usage in %**
- User: 90.60%
- System: 9.40%
- Total: 100.00%

**Network Traffic in KB/s**

**Memory Usage in %**
- 86.20%
Example 1: Monitoring the IaaS-Layer (3/3)

```sql
SELECT ip_address,
       avg(bytes_in), avg(bytes_out),
       stddev(bytes_in), stddev(bytes_out)
FROM NetworkSensor WINDOW(RANGE 1 MINUTE)
GROUP BY ip_address
HAVING stddev(bytes_in) > 0.2 * avg(bytes_in)
OR stddev(bytes_out) > 0.2 * avg(bytes_out);
```

Bastian Hoßbach, Bernhard Seeger
Cloud Monitoring
Example 2: Monitoring the PaaS-/SaaS-Layer (1/4)

<table>
<thead>
<tr>
<th>Thread</th>
<th>State</th>
<th>CPU in %</th>
<th>BlockedTime</th>
<th>#Blocked</th>
<th>WaitedTime</th>
<th>#Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompilerThread0</td>
<td>WAITING</td>
<td>0,3</td>
<td>0,000 s</td>
<td>0</td>
<td>30 m 29 s</td>
<td>1760</td>
</tr>
<tr>
<td>Finalizer</td>
<td>WAITING</td>
<td>0,0</td>
<td>0,002 s</td>
<td>10</td>
<td>20 m 49 s</td>
<td>11</td>
</tr>
<tr>
<td>Low Memory Detector</td>
<td>BLOCKED</td>
<td>0,0</td>
<td>0,525 s</td>
<td>1755</td>
<td>29 m 30 s</td>
<td>1168</td>
</tr>
<tr>
<td>main</td>
<td>WAITING</td>
<td>0,0</td>
<td>0,005 s</td>
<td>147</td>
<td>28 m 46 s</td>
<td>85</td>
</tr>
<tr>
<td>Reference Handler</td>
<td>BLOCKED</td>
<td>0,1</td>
<td>0,009 s</td>
<td>220</td>
<td>28 m 49 s</td>
<td>194</td>
</tr>
<tr>
<td>Signal Dispatcher</td>
<td>WAITING</td>
<td>0,0</td>
<td>0,000 s</td>
<td>0</td>
<td>0 m 0 s</td>
<td>9</td>
</tr>
<tr>
<td>Thread-13</td>
<td>RUNNABLE</td>
<td>18,3</td>
<td>0,015 s</td>
<td>21</td>
<td>0 m 6 s</td>
<td>12</td>
</tr>
<tr>
<td>Thread-8</td>
<td>WAITING</td>
<td>0,0</td>
<td>0,000 s</td>
<td>0</td>
<td>28 m 46 s</td>
<td>534</td>
</tr>
<tr>
<td>Timer</td>
<td>BLOCKED</td>
<td>0,0</td>
<td>0,012 s</td>
<td>541</td>
<td>29 m 49 s</td>
<td>5300</td>
</tr>
</tbody>
</table>

Heap Memory Usage in MB

Non-Heap Memory Usage in MB
Example 2: Monitoring the PaaS-/SaaS-Layer (2/4)

Bastian Hoßbach, Bernhard Seeger

Cloud Monitoring
SELECT j.jvm_key, (j.used_heap_memory - past.avg_heap_memory) / past.avg_heap_memory * 100
FROM JVMMemorySensor j,
(SELECT jvm_key,
  avg(used_heap_memory) AS avg_heap_memory
FROM JVMMemorySensor WINDOW(RANGE 2 SECONDS)
GROUP BY jvm_key) past
WHERE j.jvm_key = past.jvm_key;
Example 2: Monitoring the PaaS-/SaaS-Layer (4/4)

```
SELECT j.jvm_key, (j.used_heap_memory - past.avg_heap_memory) / past.avg_heap_memory * 100
FROM JVMMemorySensor j,
(SELECT jvm_key,
  avg(used_heap_memory) AS avg_heap_memory
FROM JVMMemorySensor WINDOW(RANGE 2 SECONDS)
GROUP BY jvm_key) past
WHERE j.jvm_key = past.jvm_key;
```
Additional Functionality on Top of CEP4Cloud

- Security monitoring
- Real-time or proactive control of cloud infrastructure
- Guarantee of SLAs
- Elasticity of *CEP4Cloud*
- Automatic installation/configuration of *CEP4Cloud* agents
- Query advisor for parameters, thresholds and new queries
- Hannover (Germany): CeBit 2011
  - Visual demonstration of first ideas
- Berlin (Germany) & Orlando (USA): Process World 2011
  - Joint venture between Software AG + University of Marburg
  - Live presentations of CEP4Cloud
1 Data model
2 Pattern matching
3 Inter-layer analysis
Relationships among different layers must be modeled:

There are more entities and other relationships in a cloud
- ’Clouds’ (one monitoring instance for parallel running clouds)
- ’Users’ (profiling, fraud detection)
- ...
Example 3: Pattern Matching

SELECT threadKey, threadState
FROM JVMThreadSensor
MATCHING ( PARTITION BY thread_key
    MEASURES threadKey String, threadState String
    PATTERN 'ab{4}'
    WITHIN 1 MINUTE
DEFINE a DO threadKey = thread_key,
    threadState = thread_state
b AS thread_key = threadKey
    AND thread_state = threadState );
Example 4: Inter-Layer Analysis

```
SELECT process_key, process_memory
FROM ProcessSensor p,
     (SELECT ip_address, max(process_memory) AS maxi
      FROM ProcessSensor WINDOW(RANGE 10 SECONDS)
      GROUP BY ip_address) sub,
     MemorySensor WINDOW(RANGE 10 SECONDS) m
WHERE p.ip_address = sub.ip_address
AND p.ip_address = m.ip_address
AND p.process_memory > 0.8 * sub.maxi
AND m.mem_free / m.mem_total < 0.2;
```