An Adaptive Replication Scheme For Elastic Data Stream Processing

Thomas Heinze, Mariam Zia, Robert Krahn, Zbigniew Jerzak, Christof Fetzer
July 02, 2015
Elasticity

- Utilization below 30% in most cloud data centers
- Users need to reserve required resources
  - Limited understanding of system performance
  - Limited knowledge of workload characteristics
Enabling Fault Tolerance

- Elasticity requires horizontal scaling → need fault tolerance
- Two mechanisms: Active Replication vs. Upstream Backup

![Graph showing recovery time vs. monetary cost with two lines and user-defined threshold]

Financial

Recovery Time (in sec.)

Monetary Cost($)

Active

Upstream

User-defined Threshold

02.07.2015

An Adaptive Replication Scheme
Outline

1. Introduction
2. An Adaptive Replication Scheme
3. Evaluation
4. Conclusion and Future Work
Related Work

a) Improving Upstream Backup:
   - Sweeping checkpoints [1,2], …
   - Faster Recovery by using Micro Batch Processing (*D-Stream* [3], *TimeStream* [4])
   - **But:** no user-configurable recovery time threshold

b) Combination of both mechanisms:
   - Already proposed for Borealis by *Hwang et al.* [5]
   - Static Optimizer proposed by *Updahyaya et al.* [6]
   - Dynamic switching to handle overload/fault case by *Martin et al.* [7]/ *Zhang et al.*[8]
   - **But:** static or without user-configurable recovery time threshold
Adaptive Replication Scheme

- Dynamically switch between upstream backup and active replication during runtime
- Replication Scheme describes current replication mode for all operators

**Key Questions:**

1) **When to switch** replication mode?
   → *Estimation Model for Upstream Backup*

2) **How to integrate** with our elastic scaling system?
   → *Adaptation Algorithm*
Estimation Model for Upstream Backup

- Many factors influence the recovery time:
  - Operator type and checkpointing interval (**static**)
  - State Size and queue length (**dynamic**)

- **Our solution**: estimation based on historical samples

- Result: 0.3 sec. error for 10 sec. recovery time (sample size: 1000)
Single Operator Scenario

- Observe current state size and queue length of all operators
- Adapt replication scheme if user threshold is not met
Integration with Elastic Scaling System

- **Architecture of an elastic scaling system**
  - Process many queries in parallel
  - Place operators on a varying number of hosts based on CPU + network consumption

- **Integration**
  - Replication-aware operator placement
  - System recovery time = max(recovery time per query)
  - **Adaptation Algorithm**: monitor recovery time for host $h$
Example: Multi Query Scenario

q1:

\[ t_{\text{rec}}(q1) = \max(t_{\text{rec}}(F1), t_{\text{rec}}(A1), t_{\text{rec}}(D1)) \]

q2:

\[ t_{\text{rec}}(q2) = \max(t_{\text{rec}}(F2), t_{\text{rec}}(A2), t_{\text{rec}}(D2)) \]

System Recovery Time:

\[ \max(t_{\text{rec}}(q1), t_{\text{rec}}(q2)) \]
Example: Operator Placement

q1:

S → F1 → A1 → D1
F1' → A1' → D1'

q2:

S → F2 → A2 → D2
F2' → A2' → D2'

Placement:

Host 1:
- F2
- F1
- A1

Host 2:
- F1'
- D1'
- A2

Host 3:
- A1'
- D2'
- D1

Host 4:
- F1'
- D2
- A2'

Recovery Time (max):
- $t_{rec}(F1), t_{rec}(F2), t_{rec}(A1)$
- $t_{rec}(A2)$
- $t_{rec}(D1)$
- $t_{rec}(A2), t_{rec}(D2)$
Example: Too High Recovery Time

Placement:

Host 1
- F2
- F1
- A1

Host 2
- F1’
- D1’
- A2

Host 3
- A1’
- D2’
- D1

Host 4
- F2’
- D2
- A2’

Recovery Time (max):
- $t_{\text{rec}}(F1)$, $t_{\text{rec}}(F2)$, $t_{\text{rec}}(A1)$
- $t_{\text{rec}}(A2)$
- $t_{\text{rec}}(D1)$
- $t_{\text{rec}}(A2)$, $t_{\text{rec}}(D2)$
Example: Too High Recovery Time

Placement:

q1:

- Host 1: F2, F1, A1
- Host 2: F1', D1', A1
- Host 3: A1', D2', A2
- Host 4: F2'

Recovery Time (max):

- $t_{rec}(F1), t_{rec}(F2)$
- $t_{rec}(A2), t_{rec}(D1')$
- $t_{rec}(D1), t_{rec}(A1)$
- $t_{rec}(A2), t_{rec}(D2)$
Evaluation
Setup

- Private cloud environment with 12 hosts
- Three Workloads: Financial, Twitter, Energy Sensors
- 20 crashes of a random host (immediately trigger recovery process)
- Recovery time as maximum latency observed after a host crash

- Two baseline algorithms: Active Replication and Upstream Backup
Recovery Time For Different Thresholds

<table>
<thead>
<tr>
<th>Recovery Time Threshold (sec.)</th>
<th>Financial</th>
<th>Twitter</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Upstream</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Legend: Green = Active, Blue = Adapt, Purple = Upstream
Adaptive Replication Scheme

![Graph showing recovery time versus monetary cost for different categories and time intervals: 2 sec. and 4 sec. for Active, Upstream, 2 sec., and 4 sec. categories.](image)
Summary

- Active replication/upstream backup forces a hard trade-off between resource overhead and recovery time
- Our adaptive replication scheme allows to customize trade-off while preserving user-defined recovery time threshold

• Future work
  - Formalize approach for replication degree >2
  - Network-bound workloads