Towards Enabling Data Intensive Enterprise Applications in Cloud

Seetharami Seelam, Paolo Dettori (IBM Research)
Pierre Andre Paumelle, Pierre Feillet (IBM Software Group)
Outline

• Introduction
• Exemplary scale-up enterprise application architecture
• Requirements for scale-out execution
• Architecture of Elastic Decision Grid
• Performance and Resource utilization implications
• Summary
Introduction

• Businesses are innovating with data-driven decision process
• Big-data processing requires handling an ever increasing velocity, volume, variety, and veracity of data
• Traditional enterprise applications are not fundamentally architected to flexibly accommodate big-data processing
• Various solutions provide big-data processing capabilities
• Enterprise applications tend to be scale-up, big-data solutions tend to be scale-out

• What would it take to integrate a scale-up enterprise application with scale-out hadoop framework?
Exemplary Enterprise Application Runtime Architecture

- Load the business logic
- Compile the business logic
- Thread initiates access to data, executes business logic, persists output
- Multiple threads used to perform computation in parallel
- Optimizations for scale-up
- Limited to single node
- Problematic to deal with large volumes of data

Architecture of Business Rules Engine

Scale-Up Model
Broad Trends in Customer Decision Requirements

Input size in records

Present

Future

Requests per Second

1 Million

1 Million

1 Million

Rules per Ruleset

1 Million

1 Million

5000

100 K

200 K

1 Billion

100 Millions

Number of Rule Sets
Requirements for Scale-out

- Need to analyze large volumes of data
- Need to analyze at the finest granularity
- Need to operate on real-time data
- Need to apply increasingly complex business logic
- Need to combine historical and real-time data
- Need to understand the impact of change in business policies
Elastic Decision Grid

• Combines enterprise logic with MapReduce execution model
  – Enterprise developers can continue to use their domain tools
  – Enables operations on Big Data
  – MapReduce model provides distributed execution
EDG Architecture
EDG Cluster Configuration (Hadoop 1.0.1)

IBM J9 VM Settings for Hadoop:

Set `mapred.map.child.java.opts` & `mapred.reduce.child.java.opts` to:

```
-Xgcpolicy:optthruput -Xmx1024M -Xms1024M -Xcompressedrefs -Xgcthreads2
```

Experiments with 16 hardware threads per node on 10 nodes
EDG Performance Study – Input and Logic

- 5824 bytes logic
  - Sequential rules
  - Ruleflow of 5 Sequential tasks
  - Each rule is a row of decision table
  - Java XOM

- 1 KB input data size

<table>
<thead>
<tr>
<th>Age</th>
<th>Income Group Code</th>
<th>Product Categories</th>
<th>Product Style</th>
<th>State</th>
<th>Low Create Date</th>
<th>Monthly Msg Frequency</th>
<th>Segment</th>
<th>Add score to segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0-25</td>
<td>books, media, electronics, home improvement</td>
<td>classic</td>
<td>AA</td>
<td>1/1/01</td>
<td>0-30</td>
<td>0-5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0-25</td>
<td>computers, software, kitchen and dining</td>
<td>trendy</td>
<td>AA</td>
<td>1/1/01</td>
<td>0-30</td>
<td>0-8</td>
<td></td>
</tr>
</tbody>
</table>

Goal: Analyze a billion input records
Proper Serialization & Compression are key

Message Pack serializer is 3.7 times faster than Java serializer
MP+BZIP2 data is 160x smaller than uncompressed java objects
Scale-up vs. Scale-out Comparison: Scale-up is better for 1 node but scale out gives distributed execution

- Scale-up: I/O costs 30% throughput
- Scale-up is 2x better than scale-out
## EDG Hadoop large scale results

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Execution</th>
<th>Avg. Time</th>
<th>TPS</th>
<th>% Time Hadoop +I/O (min)</th>
<th>% Time Compute</th>
<th>Num Mapper/Node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Total (min)</td>
<td>Rules Exe (ms)</td>
<td>Unmarshalling (ms)</td>
<td>Exec +Unmarshalling (min)</td>
<td></td>
</tr>
<tr>
<td>Stateless, No JVM Tuning</td>
<td>100 Million</td>
<td>76</td>
<td>1.37</td>
<td>0.06</td>
<td>60</td>
<td>21929</td>
</tr>
<tr>
<td>Stateful, No JVM Tuning</td>
<td>100 Million</td>
<td>67</td>
<td>1.67</td>
<td>0.07</td>
<td>48</td>
<td>24875</td>
</tr>
<tr>
<td>Stateful, J9 VM Tuning</td>
<td>100 Million</td>
<td>59</td>
<td>1.4</td>
<td>0.06</td>
<td>41</td>
<td>28248</td>
</tr>
<tr>
<td>Stateful, J9 VM Tuning</td>
<td>1 Billion</td>
<td>446</td>
<td>1.49</td>
<td>0.06</td>
<td>431</td>
<td>37369</td>
</tr>
<tr>
<td>Stateful, J9VM Tuning/SMT/native compression</td>
<td>1 Billion</td>
<td>338</td>
<td>3.22</td>
<td>0.14</td>
<td>298</td>
<td>49309</td>
</tr>
</tbody>
</table>

### Result
- Completed 1 Billion executions in less than 5.7 hours; would take 50-60 hours without optimizations
- Executed business rules in a Big Data approach for large workload footprints with elasticity and scalability
EDG Performance Study – Node scalability (1-10 nodes)

- Linear scalability with number of nodes
- But scale increases performance variation; resulting in straggler tasks
- Hadoop’s speculative task spawning results in sub-linear scaling
EDG Performance Study – Data Scalability (100 M – 2.1 B input sets)

- Optimizing shuffle phase with combiners is important
- Achieved 2B evaluations in about 10 hours
Summary

- Elastic Decision Grid combines an enterprise scale-up application with Hadoop scale-out distributed execution model.
- Scale-up execution performs better in single node case but scale-out enables multi-node execution.
- Efficient data representation, compression, and parameter tuning is necessary to achieve optimal performance.
- Enterprise software as well as Hadoop requires re-architecture to efficiently combine scale-up and scale-out models.
Questions?

• Contact: sseelam@us.ibm.com