High efficient data movement for AI training by implementing Mnemonic durable data model

Helin Cao (Intel/Client Computing Group); Gang Wang (Intel/Software and Services Group); Jianying Lang (Intel/Al Product Group)

Significance of data has been emphasized since the past century for its crucial role in scientific and industrial innovation, business competitiveness, and government strategic decision making. As a concrete example in the context of artificial intelligence (AI), model performances generally benefit from large training samples. While processing data-intensive modeling and simulation on HPC, among many ways to improve computation efficiency, one focus has been on data movement costs across storage and memory. Here we show data movement overhead can be largely reduced through implementation of “durable data model” from Mnemonic library (an Apache open source project).

New Storage Media and Opportunity

- In traditional way, Big Data developers have to build their own customized mechanisms for memory subsystem.
- Project Mnemonic optimizes memory usage for Big Data to take advantage of HW features.

Durable data model creates non-volatile objects directly saved on storage (e.g. SSD and non-volatile memory), therefore it reduces overhead thanks to the following 3 advantages:

- the objects can be read into memory without serialization/de-serialization;
- applications can handle non-volatile objects without generating unnecessary temporary objects which cause unpredictable garbage collection pauses;
- the loading, caching and checkpoint could be simplified for processing on a large number of linked objects.

Mnemonic Software Stack

Spark | Hadoop | NoSQL DB
---|---|---
Generic Non-Volatile Java Object Model
Java Std. Library
JVM
Mnemonic

Contains:
- Generic Non-Volatile Java Object Model Specification and sample usage code
- Mnemonic library implementation
  - ~6000 lines of code
  - 88% Java, 11% C code, 1% shell scripts
- Integration for Spark Machine Learning: Non-Volatile RDD (Resilient Distributed Dataset)
  - ~600 lines of Scala code

Mnemonic Software Stack

Apache Spark

![Mnemonic Software Stack Diagram]

Spark Integration on Intel® hardware

In Memory Computing Applications

- Partitioning
- Sort-Merge-Shuffling
- Cache/Non-volatile RODs
- Directed Acyclic Graph (DAG)
- Caching
- Persistent Objects

Mnemonic

- Support heterogeneous memory, NVMe, SSD, Network and Cloud Storage
- JVM/Runtime Heap management
- M HARDWARE
- Network Cloud Storage
- M HARDWARE
- System Memory
- DRAM
- Persistent Memory
- NVMe

Spark + Mnemonic: MLlib Kmeans

Major Kmeans Steps Performance Impact
(Time to complete the work, the lower the better)

Load data | TakeSample | Collect | Aggregate
---|---|---|---
72 | 32 | 36 | 66
32 | 66 | 76 | 40

H. Cao thanks Michael Rosenzweig, Guru Raj for supporting this pathfinding efforts.
G. Wang acknowledges Uma Maheswaran, Rao G for mentoring Mnemonic project and Milind Damle for directing and managing project execution.
J. Lang acknowledges Jennifer Myers and Arjun Bansal for the travel support.

Spark + Mnemonic: MLlib Kmeans

Average 1.95X faster due to GC reduction

<table>
<thead>
<tr>
<th>Exp:1</th>
<th>GC Sum of 20 executors</th>
<th>Default</th>
<th>Mnemonic</th>
<th>Mnemonic/Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of FullGC</td>
<td>23</td>
<td>12</td>
<td>0.522</td>
<td></td>
</tr>
<tr>
<td>Total number of YoungGC</td>
<td>1310</td>
<td>864</td>
<td>0.660</td>
<td></td>
</tr>
<tr>
<td>Total FullGC Pause Real (s)</td>
<td>129.36</td>
<td>59.52</td>
<td>0.460</td>
<td></td>
</tr>
<tr>
<td>Total YoungGC Pause Real (s)</td>
<td>225.98</td>
<td>158.79</td>
<td>0.703</td>
<td></td>
</tr>
<tr>
<td>Total GC STW pauses (s)</td>
<td>355.34</td>
<td>218.31</td>
<td>0.614</td>
<td></td>
</tr>
</tbody>
</table>

Specifications for Performance Measurement:
- Platform: Software Stack
- JRE: Oracle 1.8.0.3
- Optimized: Intel® HPC Developer Conference 2017
- Configuration:
  - JVM: JDK8 update 60;
  - Hadoop: 2.7.0;
  - Hadoop + Mnemonic: 2.7.0 (Hybrid Mnemonic)
  - Hadoop: 2.7.0 (Hybrid Mnemonic)
  - Mnemonic: 1.0.0 (Beta)

Guru Rao G for mentoring Mnemonic project.

Intel® HPC Developer Conference 2017