FMEM: A Fine-grained Memory Estimator for MapReduce Jobs

Lijie Xu\textsuperscript{1,2}, Jie Liu\textsuperscript{1}, and Jun Wei\textsuperscript{1}

\textsuperscript{1}Institute of Software, Chinese Academy of Sciences
\textsuperscript{2}University of Chinese Academy of Sciences

6/26/2013
Outline

- What is FMEM?
- Why do we develop it?
- Is it a hard problem?
- How to develop it?
- How to estimate the fine-grained memory usage?
- Evaluation
- Related work
- Conclusion
What is FMEM?

- It is designed to **analyze**, **predict** and **optimize** the fine-grained memory usage of map and reduce tasks.
Why do we want to develop FMEM?

- **For users**
  - feel hard to specify the memory-related configurations.
    - e.g., buffer size, reducer number, Xmx, Xms, and etc.
  - OutOfMemory error or performance degradation

- **For task schedulers**
  - e.g., YARN and Mesos
  - they allocate resource and schedule tasks according to the CPU and memory requirement of mappers/reducers.

- **For Hadoop committers**
  - design a more memory-efficient framework.
So, what is the fundamental problem?

- Tell job’s memory usage before running it on big dataset.
- \( f(\text{Data } d, \text{ Conf } f, \text{ Program } p, \text{ Resource } r) \)
  → memory usage \( mu \) of mappers and reducers
- Add \( <\text{SampData } sd, \text{ SampConf } sf, \text{ SampProfiles } sp> \)
  → memory usage \( mu \) of real mappers and reducers
Is it a hard problem?

User layer

Framework layer

Execution layer

Data

map() {
} reduce() {
}
Pig /HiveQL

Memory

JVM

disk

 Eden Space and New Gen Heap Memory Usage

Old Generation Heap Memory Usage

Old Gen

S0

S1

Eden

Perm Gen
How to do it? dataflow→objects→usage

User layer

Data

Configuration

Pig / HiveQL

Framework layer

Dataflow volume

In-memory objects’ size

Execution layer

memory usage

rules of memory usage
How to do it? dataflow → objects → usage

- **Built-in Monitor**
  - get Records/Bytes Statistics & real-time memory usage

- **Profiler**
  - count intermediate data and TempObjs in each phase
Estimate the dataflow

- We actually build a MapReduce simulator to estimate the intermediate data under specific \(<\text{Data } d, \text{ Conf } f, \text{ sample profiles } sp>\).
Which are memory-consuming objects?

- **Memory buffer** (configuration)
  - e.g., spill buffer (io.sort.mb) and user-defined arrays

- **<K, V> records** (dataflow)
  - map() outputs records into spill buffer, in-memory segments in shuffle buffer

- **Temporary objects** (dataflow and programs)
  - byproducts of records such as char[], byte[], String and ArrayList
  - A WordCount mapper produces 480MB java.nio.HeapCharBuffer objects with 64MB input split.

- **Others**
  - code segment, native libraries, call stack and etc.
How to do it? dataflow $\rightarrow$ objects $\rightarrow$ usage

- Memory usage
  - In-memory objects
  - Management mechanism
  - Summarized rules
  - Dataflow volume
  - Configuration
  - Generation GC effects
  - Job history
Rules-statistics based approach

RULE 1. $MapOU \approx f(conf)$
RULE 2. $MapNGU \approx f(Xmx, Records, TempObjs)$
RULE 3. $RedNGU \approx f(Xms, Xmx, Records, TempObjs)$
RULE 4. $RedOU \approx f(Xms, Xmx, Records, TempObjs)$
How to do it? dataflow → objects → usage

User layer

Framework layer

Execution layer

Dataflow volume

Data

Configuration

Memory

In-memory objects’ size

Data

Configuration

Map() { }
Reduce() { }
Pig / HiveQL

Map

Reduce

JVM

Memory

Eden Space and New Gen Heap Memory Usage

Old Generation Heap Memory Usage

memory usage

rules of memory usage
Evaluation

Table 1: Representative MapReduce Applications

<table>
<thead>
<tr>
<th>Applications</th>
<th>Dataset</th>
<th>Combine</th>
<th>Compress</th>
</tr>
</thead>
<tbody>
<tr>
<td>WikiWordCount</td>
<td>9.4 GB</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>BuildInvertedIndex</td>
<td>9.4 GB</td>
<td>N</td>
<td>SeqBlock</td>
</tr>
<tr>
<td>UserVisits_Aggre-pig</td>
<td>75 GB</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>TwitterBiEdgeCount</td>
<td>24.4 GB</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>TeraSort</td>
<td>36 GB</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

\[
\text{relative error} = \left| \frac{\text{emu} - \text{rmu}}{\text{rmu}} \right| \times 100\%
\]

Figure 2: Relative error with standard deviation of \( <\text{emu},\text{rmu}> \)
How about the related work?

- Execution time estimation
  - ParaTimer, KAMD, ARIA and etc.

- Find optimum configurations
  - Starfish can predict job’s performance (mainly runtime) with different configurations and has a cost-based optimizer to find the optimum configuration.

- Find optimum GC policy for multicore MapReduce
  - Garbage collection auto-tuning for java MapReduce on multi-cores, ISMM 2011
Provide a detailed analysis of job’s memory usage
  • considering dataflow and memory management from user-level to JVM internals.

Develop a fine-grained memory estimator
  • predict job’s memory usage in a large space of configurations.

FMEM will be improved to do auto-configuration, auto-optimization and etc.
  • JIRA MAPREDUCE-4882 & MAPREDUCE-4883
Thanks, Q&A