Hadoop Performance Self-Tuning Using a Fuzzy-Prediction Approach

Authors: Gil Jae Lee and José A. B. Fortes
Summarized by: Ibifubara Iganibo
OUTLINE

• Introduction
• Background & Motivation
  • Map Reduce
  • YARN
  • Fuzzy logic
• Design and implementation
• Experimental Evaluation
• Conclusion
ACRONYMS

- **AM**: ApplicationMaster;
- **AMRP**: yarn.scheduler.capacity.maximum-am-resourcepercent;
- **Atask(t)**: task activity ratio;
- **GR**: Grep;
- **nAM(t)**: number of concurrently running AMs;
- **MR**: MapReduce;
- **NM**: NodeManager;
- **ntotal**: total number of containers that can run simultaneously in the cluster;
- **RM**: ResourceManager;
- **TS**: Terasort;
- **WC**: Wordcount
INTRODUCTION
WITHOUT YARN
PROBLEM DESCRIPTION

- Hadoop has over 200 tunable configuration parameters
- Manual tuning of Hadoop performance is hard and time-consuming
- Other approaches have :-
  - A time-consuming training phase
  - Unsuitable for concurrently running varied MR jobs
The proposed approach optimizes Hadoop performance by using a fuzzy-prediction controller for dynamic adjustment of the number of concurrent MR jobs.

- The controller learns from past and current values.
- Self-optimization method that updates the AMRP value 
  \( \text{yarn.scheduler.capacity.maximum-am-resourcepercent} \) dynamically rather than using a static value.
BACKGROUND & MOTIVATION

• What is MapReduce (MR)?
• MapReduce is a programming model for processing large amounts of raw data.
BACKGROUND & MOTIVATION

• What is YARN?
  • It is a resource manager for Hadoop clusters
  • YARN is a framework to provide computational resources for execution engines.

• Components of YARN
  • Resource Manager (One per cluster)
  • Node Manager (one per data Node)
COMPONENTS OF YARN
NODE MANAGER
STEPS TO LAUNCH MR JOB TASKS
BACKGROUND & MOTIVATION

• What is Fuzzy logic?
  • Within 1 and 0
  • Partially true or partially false

• Fuzzy-prediction controller constructs fuzzy rules based on resource usage of MR jobs
EXPERIMENTAL SETUP
EXPERIMENTAL SETUP

MR jobs used for the experiments:
- Terasort (TS) for IO-intensive (memory)
- Grep (GR) for CPU-intensive
- Wordcount (WC) for balanced jobs
CORRELATION BETWEEN RESOURCE USAGE AND TASKS?

• Fuzzy-prediction controller constructs fuzzy rules based on resource usage of MR jobs in real-time

• What is Correlation between resource usage and number of concurrent tasks in each MR jobs?

• The proposed controller uses 2-dimensional inputs consisting of CPU usage and the number of concurrent tasks.

<p>| Table II: Correlation between the number of concurrent tasks and CPU usage or memory usage when (1) each MR job is scheduled individually after the prior job finishes (second, third and fourth columns) and (2) 3 jobs are scheduled at 5-seconds intervals (last column). |
|---------------------------------|------|------|------|------|</p>
<table>
<thead>
<tr>
<th>Tasks vs. CPU usage</th>
<th>TS</th>
<th>GR</th>
<th>WC</th>
<th>3-TSWCGR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks vs. memory usage</td>
<td>0.97</td>
<td>0.98</td>
<td>0.97</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>0.84</td>
<td>0.86</td>
<td>0.86</td>
<td>0.91</td>
</tr>
</tbody>
</table>
STATIC CONTROL OF MR JOBS

• What number of concurrent MR jobs affects overall performance of a Hadoop cluster?

• Changing the AMRP value improved performance significantly - up to 36% (in case of 18-TS).

![Graph showing execution time for different MR job sequences]

Fig. 2. Execution time of 6-TSWCGR, 18-TS, 18-GR, and 18-WC with the default AMRP value (0.1) and the optimal AMRP value (0.3, 0.3, 0.5, and 0.2, respectively) for each MR job sequence.
DESIGN AND IMPLEMENTATION
### Table III: Parameters used in the proposed approach, which are derived from some of Hadoop default values.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM_POSSIBLE_TASKS</td>
<td>Total number of containers that can be used simultaneously for tasks and AMs in the cluster, initially set by Hadoop. E.g., NUM_POSSIBLE_TASKS is 40 when there is 40GB of memory.</td>
</tr>
<tr>
<td>AM_UNIT</td>
<td>Number of containers that an AM takes. Hadoop default is 2.</td>
</tr>
<tr>
<td>AM_PERCENT_UNIT</td>
<td>Equals $AM_UNIT/NUM_POSSIBLE_TASKS$. AMRP is increased/decreased by this value when increasing/decreasing the number of AMs by one.</td>
</tr>
</tbody>
</table>
AUTONOMIC CONTROL OF MR JOBS

• Proposed a method to dynamically control the number of concurrent MR jobs by adjusting AMRP
• Controlling the number of concurrent MR jobs by properly setting the value of AMRP can improve performance significantly
AUTONOMIC CONTROL OF MR JOBS

• **Local Resource Monitor**
  ▪ Collects overall resource usage of all MR jobs running in the node

• **Global MR Job Monitor**
  ▪ Collects the resource usage information from local monitors
  ▪ Checks the status of each MR job from the Resource Manager
  ▪ Feeds input into the fuzzy-prediction controller

• **Fuzzy-Prediction Controller**
  ▪ Seeks to avoid time-consuming training and over/under-allocation of MR jobs
  ▪ The fuzzy-prediction controller only needs to know resource usage of MR jobs.
  ▪ Based on the past resource usage in the system, future resource usage is predicted.

• **Capacity Scheduler Controller**
Fuzzy-Prediction Controller

• Seeks to avoid *time-consuming training* and *over or under* allocation of MR jobs based on resource usage in the cluster

• The Fuzzy-Prediction Controller consists of the following modules:-
  • Fuzzy Rule Construction
  • Fuzzy Rule Update
  • Fuzzy Inference Module
FUZZY RULE CONSTRUCTION

• To create fuzzy rules, input and output spaces are divided into $2N + 1$ domains:
• A fuzzy membership function is assigned to each domain
• The input values are normalized to $[0, 1]$
• Since the number of concurrent tasks $n_{\text{task}}(t)$ (not including AMs) is an integer, it is converted to task activity ratio ($A_{\text{task}}(t)$) at time $t$

$$A_{\text{task}}(t) = \frac{n_{\text{task}}(t) + 2 \times n_{\text{AM}}(t)}{n_{\text{total}}}$$
FUZZY RULE UPDATE

• Whenever a new fuzzy rule is generated, the rule base is searched for whether or not rule-new exists

• If the search finds a matching rule i.e (rule-old)

• The membership values of the output part in ruleold are replaced by the centroids of the outputs of ruleold and ruleneu.

• Otherwise, ruleneu is added to the rule base. There would be a conflict in the rule base if an existing rule has the same IF part as ruleneu
The fuzzy inference module has four components:

- Fuzzy inference engine
- Fuzzification interface
- Defuzzification interface
- Rule base
CAPACITY SCHEDULER CONTROLLER

• Capacity scheduler controller uses the **CPU usage and task activity** information to **predict values**
• It determines whether the current AMRP value is appropriate in the given environment
• Updates the AMRP value, accordingly.
RAMPING

• An MR job is in “RAMPING” status if it is in “RUNNING” status and its AM task is launched, but no containers have been allocated to its other tasks.

• When there are many RAMPING AMs, there is significant performance degradation in the Hadoop system.

AMRP value fixed at 0.8

AMRP value fixed at 0.2
Under-allocation. The controller considers if the Hadoop cluster still has room for new AMs.

The value of AMRP will be increased only if the concurrent MR jobs is greater than or equal to the maximum number of MR jobs runnable under the current value of AMRP (line 5).

Over-allocation. Happens when increasing the number of concurrent AMs would not contribute to improve resource usage.

The scheduler ensures that there is no ramping job.

The value of AMRP is updated only if the new AMRP is different from the current one.

```
Algorithm 1 Updating the AMRP value
Input: U_{cpu}(t): current CPU usage at time t;
      A_{-task}(t): current task activity;
      U_{cpu}(t+1): predicted CPU usage;
      A_{task}(t+1): predicted task activity;
      n_{queue}(t): number of MR jobs in the queue;
      n_{total}(t): number of running jobs;
      n_{ramp}(t): number of ramping jobs;
      AMRP(t): current AMRP value at time t.
// maximum number of MR jobs runnable
1: n_{total}(t) = int(AMRP(t)/AM_PERCENT_UNIT).
2: ΔU_{cpu} ← U_{cpu}(t+1) - U_{cpu}(t).
3: ΔA_{task} ← A_{task}(t+1) - A_{task}(t).
4: If n_{total}(t) > 1 Then
5:   If n_{total}(t) >= n_{total}(t) Then
6:     If ΔU_{cpu} > 0. And ΔA_{task} > 0. Then // under-allocation of AMs
7:       newAMRP ← AMRP(t) + AM_PERCENT_UNIT.
8:   Else If ΔU_{cpu} < 0. And ΔA_{task} < 0. Then // over-allocation of AMs
9:     If n_{ramp}(t) = 0 Then // Wait until change of AMRP is effective
10:    newAMRP ← AMRP(t) - AM_PERCENT_UNIT.
11: End If
12: End If
13: End If
14: Else // No MR jobs waiting in the queue
15:   If n_{total}(t) < n_{total}(t) Then
16:     newAMRP ← AMRP(t) - AM_PERCENT_UNIT.
17: End If
18: End If
19: If newAMRP != AMRP(t) Then
20:   AMRP(t) ← newAMRP. // update AMRP with new AMRP
21: End If
```
Experimental Evaluation
Experimental Evaluation

• Experimental results were used to compare the performance of the proposed approach against the Hadoop default configuration.
Experimental Evaluation

Static versus Dynamic AMRP Values (value = 0.1, MR jobs =30 , dataset = 1GB)
• The proposed dynamic approach shows performance close to the static optimal AMRP cases. However the difference is insignificant -- less than 4%.
• But it not practical to find the static optimal of all MR jobs

Fig. 12. Normalized execution times of 30-MR job sequences using static and dynamic AMRP values with 1GB datasets.

Fig. 13. Normalized execution times of 30-MR job sequences using static and dynamic AMRP values with with 5GB datasets.

Fig. 14. Normalized execution times of 30-MR job sequences using static and dynamic AMRP values with with 10GB datasets.
Experimental Evaluation

- Static versus Dynamic AMRP Values (value = 0.2)
- In fig 18. Previous approach (memory) shows spikes as high 55%
Conclusion

• This paper proposed the use of a fuzzy-prediction controller for self-optimization of the number of concurrent MR jobs to maximize the throughput in the Hadoop system.

• Introduced the notion of “RAMPING” status of “RUNNING” MR jobs. “RAMPING” which can have negative influence on performance.

• They achieved significant improvement in performance of MR jobs by adjusting a single parameter in YARN regardless of resource usage patterns of the MR jobs.

• Proposed solution that does not require changes to either the MR jobs or Hadoop framework.
Thank you