AROMA: Automated Resource Allocation and Configuration of MapReduce Environment in the Cloud

Palden Lama and Xiaobo Zhou, ICAC '12 Proceedings of the 9th international conference on Autonomic computing Pages 63-72.

Summarized by: Drew Wicke
October 13 2015
Introduction

• Distributed data processing in the cloud (MapReduce Hadoop)

• Problems:
  • How to allocate VMs cost effectively.
  • How to set the parameters in Hadoop.

• Goal:
  • Meet job completion deadlines and minimize cost.

• Solution:
  • AROMA: Automatically allocate VMs and set Hadoop params. to meet deadlines and minimize costs.
Challenges

• Heterogeneous dynamic cloud environment (not just Hadoop VMs) therefore same parameters might not produce same results.

• No performance models for such a complex distributed processing framework.

Note that a technical report in 2011 and a workshop paper in 2012 give performance models for Hadoop.

MapReduce

from
MapReduce Parameters

- Number of Reducers.
- Size of memory buffer for sorting map output.
- Number of concurrent connections to fetch input from Mapper.
Experiment Setup

• Seven HP Pro-Liant BL460CG6 blade server modules and a HP EVA storage area network. (Custom built cloud environment).

• Hadoop jobs: Sort, Wordcount and Grep (standard examples from Hadoop distribution).

• RandomWriter and RandomTextWriter tools in the Hadoop package to generate data of various sizes for the Sort, WordCount and Grep programs.

• Compare AROMA to default.
AROMA

User

Submit Job

AROMA

Allocate #/Type VMs

Configure Parameters

Job Manager

Resource Allocator

Turn on VMs

Submit To Master
AROMA - workflow
Offline Phase

- Process log files.
- Collect resource utilization data.
- Produce performance models.
Clustering

• Group jobs with similar utilization patterns of CPU, network and disk resources.

• K-Mediod based clustering technique, minimizes a sum of pairwise dissimilarities (rather than distances as is with k-means).

• Longest Common Subsequence (LCSS) distance measure to deal with noise.
Clustering

Sort and Wordcount are in the same cluster.

Grep is in the second cluster.
Performance Modeling

• For each cluster AROMA produces a performance model.

• Features are selected using a step-wise regression technique.

• Support Vector Machine (SVM) estimates the completion time of jobs belonging to a cluster for different input data sizes, resource allocations and configuration parameters.
SVM vs Linear Regression

Why compare to linear regression?
Note

• Stepwise Regression is very controversial [1-3].

• Possible overfitting [2].

• “It yields p-values that do not have the proper meaning, and the proper correction for them is a difficult problem” [3].

• Details in the paper on the stepwise regression are sparse.

Online Phase

• Run a scaled down version of the job (reduced VMs and data size).

• Collect performance data (the resource utilization signature).

• Determine which cluster the job belongs to by comparing to cluster centroid using LCSS of the signatures.
Optimizer

• Solve the non-linear constrained optimization problem for minimizing the number of VMs allocated while constraining the time to less than the deadline and appropriate settings for Hadoop.

• Solved using a pattern search algorithm, the generating set search. No specific toolbox/implementation is mentioned. No runtime or cost for calculating is mentioned.
Experiment Setup

- Seven HP Pro-Liant BL460CG6 blade server modules and a HP EVA storage area network. (Custom built cloud environment).

- Hadoop jobs: Sort, Wordcount and Grep (standard examples from Hadoop distribution).

- Compare AROMA to default.
Experiments

- Performance Model Accuracy
- Auto-configuration
- Resource Allocation and Configuration
- Adaptiveness to Ad-hoc Jobs
Note that the relative error between the actual and predicted job completion time is less than 12% on all the cases (sort, word count and grep).
Auto-Configuration

• Only tested for the Hadoop Sort benchmark

• Parameters for Hadoop are auto-configured for both small and medium size VMs using 10 GB input data

• Graphs showing the performance and cost for both small and medium sized VMs with 4, 6, 8 and 10 VMs.

• “The improvement in job performance and cost due to AROMA increases from 17% to 30% when using more number of [small] VMs.”

• However, the paper shows the Default vs AROMA parameters found for 6 small and medium VMs on 20 GB input data, but no performance and cost graphs.
Resource Allocation and Configuration

• They next compared AROMA with and without auto-configuration

• They showed for the sorting problem that on average AROMA can produce a cost efficiency of 25%

• Demonstrated that AROMA’s optimizer is important to weed out costly configurations
Adaptiveness to Ad-hoc Jobs

- To test AROMA's adhoc ability they studied the wordcount problem with 10 GB input.
- Found that AROMA is able to identify it as similar to the Sort job.
- AROMA uses the info gathered in the staging area and the prediction data to scale the resource allocation and configurations.
- They claim less than 12.5% error in prediction of execution time.
- They show prediction versus actual execution times for 20 different configurations.
Conclusion

• AROMA automatically sets
  • VM configuration (number of homogenous VMs)
  • Hadoop configuration

• “Optimal job provisioning decisions with respect to resource allocation efficiency in the face of unpredictable input data sizes and performance expectation”
Questions

- Why did they not find or create performance models?
- Why did they examine so few test cases?
- Why did they use step-wise regression when other methods exist?
- What does AROMA stand for?