Model-driven Autoscaling for Hadoop clusters
Anshul Gandhi (Stony Brook University); Parijat Dube, Andrzej Kochut, Li Zhang (IBM)

Problem

• Hadoop performance vulnerable to variations in cloud
  – Worker nodes can fail during job execution
  – Resource contention in the cloud can dynamically impact progress
  – Such variations lead to SLO violations if left unattended

• Prior work:
  – Mostly (ARIA, CRESP, Starfish) focuses on optimal static allocation
  – Others (KOALA, Jockey) rely on heuristics or complex simulations

• How to accurately and dynamically resize Hadoop?

Problem Statement: How to successfully autoscale Hadoop while job is in progress

Challenges

• How to estimate Hadoop resource requirements?
  – Complex system, several metrics (200+ via Ganglia)
  – Workload- and data-dependent behavior
  – Need a practical model relating resource allocation and performance (execution time)

• Cloud environment is very dynamic
  – Workload volume and mix are subject to change
  – Node failures, resource contention are common
  – Need a dynamic solution

Solution

• Model-driven approach to autoscaling
  1. Develop workload-dependent performance models
     – Closed-form expressions relating performance to various parameters (resources, workload, Hadoop)
     – Focus on few important parameters
  2. Leverage performance models for autoscaling
     – Keep track of %age input data processed
     – Scale-out: Launch new VMs and start Hadoop services
     – Scale-in: Stop Hadoop services and remove VMs

Modeling Results

• WordCount: \( T_{\text{map/red}} \), map/red stage time

\[
T_{\text{map}} = \left( 430 \frac{D}{M} + 6 \right) \left( \frac{M}{N_{mc} \cdot n_{mc}} \right) \cdot n_{ms}
\]

\[
T_{\text{red}} = \left( 5 + 0.5 \frac{D}{R} + 6 + 0.7 \frac{D}{R} \right) \left( \frac{R}{N_{rc} \cdot n_{rc}} \right) \cdot n_{rs} + 0.1 \frac{M}{R}
\]

\( M \) (\( R \)) Number of Map (Reduce) tasks
\( N_{mc} \) (\( N_{rc} \)) Number of Map (Reduce) configured cores
\( n_{mc} \) (\( n_{rc} \)) Number of Map (Reduce) slots per core
\( D \) Size of input data, in GB

– \((M/R)\) term for data movement in Shuffle

– Obtained via regression on training data
– Similar results for TeraSort and Kmeans
– Modeling error is about 4% (max 10%)

Autoscaling Evaluation

– WordCount results on various Hadoop clusters
– Autoscaling managed by simple reactive controller

Lessons:

• Simple analytical models can suffice for resource estimation

• Hadoop jobs can be dynamically autoscaled to meet SLOs

Limitations:

• Preliminary results based on simple use-cases
• Need to address HDFS data movement