Aeromancer: A Workflow Manager for Large-Scale MapReduce-Based Scientific Workflows

Presented by Sarunya Pumma
Supervisors: Dr. Wu-chun Feng, Dr. Mark Gardner, and Dr. Hao Wang
Outline

• Introduction & Motivation
  • Cloud computing and its benefits
  • SeqInCloud and scalability analysis
• Aeromancer
  • Usage scenario
  • Workflow and architecture
  • Discussion
• Experiments & Initial Results
Introduction & Motivation
What is Cloud?
What is Cloud?

I would like 6 virtual machines for a month

I would like 3 virtual machines for 2 months

virtual resources : resource pool

physical resources

user a

user b
How is Cloud beneficial for Us?
How is Cloud beneficial for Us?

Next-generation sequencing (NGS) produces a huge amount of DNA sequence data

- 2k sequencers produce 15 PB genetic data/year

Cost of sequencing a human-size genome decreases over time

- 2011 → $95M
- 2012 → $6.5k
- 2014 → $1k
How is Cloud beneficial for Us?

Big Data
How is Cloud beneficial for Us?

Cloud can accelerate our ability to identify the cure for cancer.
Provisioning compute and storage resources to analyze the exponentially increasing genomic data
SeqInCloud

- It is implemented based on GATK (Genome Analysis Toolkit)
- All six stages have to be executed in order
- Each stage is implemented as a small separated Hadoop program
  - It can be broken down into small jobs and run on multiple computers/processors
  - We expect a linear speed-up on every stage
SeqInCloud
Genome variant analysis pipeline

1. FASTQ file
   - Align (BWA)
   - Sort
   - Index
   - Local-Realignment & Sort
   - MarkDuplicate

2.
   - Align (BWA)
   - Sort
   - Index
   - Local-Realignment & Sort
   - MarkDuplicate

3. Count Covariate
   - Count Covariate
   - Merge Covariates
   - Table Recalibration
   - Genotyper & Variant Filter
   - Combine Variants

4.
   - VCF file
   - BAM file
SeqInCloud Scalability Analysis

• To run SeqInCloud efficiently on Hadoop
  • Number of compute nodes must be varied based on the scalability of each stage

• Challenges
  1. How many nodes do we need for each stage?
  2. How to rapidly adjust a number of nodes with negligible overhead?

An automatic Hadoop workflow execution is needed!
Hadoop Workflow (SeqInCloud) → Hadoop Workflow Manager → Cloud
Hadoop Workflow (SeqInCloud) → Aeromancer → Cloud
Aeromancer

• Platform for executing the scientific Hadoop workflows
  • Based on Hadoop
    • MapReduce
    • HDFS
  • Build on top of CloudGene
  • Workflow (DAG)
    • Node/Stage
    • Edge (represent dependency between two stages)
Aeromancer Usage Scenario

1. Add a new application to Aeromancer
Aeromancer Usage Scenario

Application details

Public cloud details

Execution details
- Execution command
- Stage dependencies
- Data dependencies

YAML file
Aeromancer Usage Scenario

1. Add a new application to Aeromancer
2. Submit a new job
Aeromancer Usage Scenario

User Portal
Aeromancer Usage Scenario

1. Add a new application to Aeromancer
2. Submit a new job
3. Wait for the execution to complete
Aeromancer Discussion

• Currently, Aeromancer worker nodes are virtual machines in the cloud

• Pro
  • Number of worker nodes are flexibly adjustable upon demand

• Con
  • The overhead of creating/deleting worker nodes is high
  • Execution speed drops significantly

Containerization may be able to improve Aeromancer’s performance
Virtualization

Containerization

Reference: https://www.docker.com/whatisdocker
Experiment

• Platforms
  • Bare metal machines
  • KVM
  • Docker

• Benchmarks (HiBench)
  • Sort
  • WordCount
  • TeraSort

• Performance
  • Speed (execution time)
  • HDFS throughput

• Nutch Indexing
• PageRank
• EnhancedDFSIO
Experiment: Hadoop on Bare Metal Environment

MiCloud2 (Master Machine)

- ApplicationHistoryServer
- SecondaryNameNode
- JobHistoryServer
- AmbariServer
- NameNode
- ResourceManager

MiCloud3 (Slave Machine)

- DataNode
- NodeManager

MiCloud5 (Slave Machine)

- DataNode
- NodeManager
Experiment: Hadoop on Virtual Machine Environment

AmbariServer
ApplicationHistoryServer
SecondaryNameNode
JobHistoryServer
NameNode
ResourceManager

Micloud2
(Master VM)

br0
eth0

br0

NodeManager
DataNode

MiCloud3
(Slave VM)

br0

NodeManager
DataNode

MiCloud5
(Slave VM)

eth0: 192.168.4.x/24
br0: 192.168.254.x/24

12 vCPUs
16 GB-vMemory
Experiment: Hadoop on Docker

Micloud2 (Master Container)
- AmbariServer
- ApplicationHistoryServer
- SecondaryNameNode
- JobHistoryServer
- NameNode
- ResourceManager

MiCloud5 (Slave Container)
- NodeManager
- DataNode

MiCloud3 (Slave Container)
- NodeManager
- DataNode

Network:
- eth0: 192.168.4.x/24
- br0: 192.168.6.x/24

Resources:
- 12 vCPUs
- 16 GB-vMemory
Initial Results: Slow Down

- Sort: KVM 62%, Docker 95.4%
- Wordcount: KVM 92%, Docker 98.6%
- DFSIOE: KVM 14%, Docker 59.5%
- NutchIndexing: KVM 88%, Docker 97.4%
- TeraSort: KVM 9%, Docker 56.8%
- PageRank: KVM 83%, Docker 94.8%
Future Work

- Automatic resources provisioning
  - Determining the number of computing nodes needed for each stage
    - Based on the scalability of each stage
    - Based on user’s constraints (budget/expected response time)
- Automatic task scheduling
  - Determining the optimal execution location for each stage (either client or cloud)
Thank you