Position Paper: Cloud System Deployment and Performance Evaluation Tools for Distributed DBs

Markus Klems – HotTopiCS Workshop 2013, April 20–21, 2013
Agenda

- Introduction
- Background on Cassandra and Cloud Benchmarking
- Our Experiment Framework & Tools
- Lessons Learned
- Related Work
- Conclusion & Outlook
Introduction

- **NoSQL/Cloud Databases**
  - Examples: Dynamo, BigTable, Sherpa, Cassandra, HBase, …
  - Distributed database systems which are designed in favor of high performance, elastic scalability, and high availability
  - Application-specific database solutions in lieu of one-size-fits-all

- **Guiding Questions**
  - Which database should I use for my application?
  - How well does the database really perform?
  - How can I optimize the database?
  - What are the trade-offs?

- **Problems**
  - The systems are too complex to be captured in analytical models
  - Performance evaluation experiments with distributed systems require substantial effort and use of hardware resources
Introduction – our experimentation experience

- Semi-automated performance experiments deliver initial insights.
- Conducting these experiments requires a lot of manual effort.
- It is difficult to reproduce the experiments without mistakes.
- The system under test is a “moving target” – every new software release brings along a lot of changes.

Cassandra average throughput, depending on the number of small EC2 instances and replication factor N [Klems, et. al. 2012].
Introduction

- **Objectives of this talk**
  - Suggest concepts for experiment automation in the cloud
  - Discuss challenges and lessons learned
- **Example: Apache Cassandra on Amazon EC2**
Background: Apache Cassandra
Dynamo-style Replication Architecture

Cassandra Cluster with 4 nodes
and Replication Factor = 2
Background: Apache Cassandra

Dynamo-style Replication Architecture

Cassandra Cluster with 4 nodes and Replication Factor = 2

async write to other node

sync write data to the fastest responding node

write data to key=633 consistency level 1

Cassandra Client
Background: Apache Cassandra
BigTable-style Storage Engine

Native OS Memory

Java Heap Memory

Disk

Write

commit log

SSTable

SSTable

SSTable

Bloom filter

Bloom filter

flush

Key cache - row cache

Read

1.

2.

3.

4.
Background: Cloud Benchmarking with the Yahoo! Cloud Serving Benchmark Tool (YCSB)

Source: [Cooper et al. 2010]
Background: Workload Model

Probability distributions: Horizontal axes represents items in order of insertion, and vertical axes represent probability of being chosen.

Source: [Cooper et al. 2010]
Experiment Framework

- **Experiment Module**
  - Capture the experiment plan in a “software module”

- **Elastic Lab**
  - **launch** - Provision a cluster or individual machine in the compute cloud in a given configuration.
  - **destroy** - Destroy the cluster or individual machine by terminating the virtual machine(s).
  - **apply-treatment** - Apply the experiment treatment to one of {cloud, cluster, machine}.
  - **prepare-experiment** - Configure the observer servers and the experimental unit servers.
  - **run-experiment** - Run an experiment and collect observation data.

- **Collaboration Features**
  - Collaborate on experiment modules
  - Share observation measurements
An Experiment Module’s Experiment Profile

service1:
  name: cassandra
  attributes:
    backup: ec2
    replication_factor: 1
    partitioner: RandomPartitioner
    key_cache_size_in_mb: 93
    row_cache_size_in_mb: 93

service2:
  name: ycsb
  attributes:
    load: 'no'
    recordcount: 3000000
    operationcount: 3000000
    readproportion: 0.99
    updateproportion: 0.01
    requestdistribution: veryhotspot
    hotspots: 300
    variance: 0.0001
    alpha: 0.5

profile1:
  provider: aws
  snapshot: false
  regions:
    region1:
      name: us-east-1
      machine_type: xlarge
      template: 3 cassandra, 1 ycsb
    region2:
      name: us-west-1
      machine_type: xlarge
      template: 2 cassandra, 1 ycsb
Elastic Lab – Launch (1)
Elastic Lab – Run Experiment (2)
Elastic Lab – Collect Data (3)
Elastic Lab – Destroy (4)
Multi-Cloud & Multi-DB Experiment Tool Design

1. a launch-cluster
2. a run-experiment
4. a destroy-cluster

1. c deploy & configure services
1/4. b launch/destroy instances

2. b pull
2. c push

Experiment module version control

VM instance
- cassandra service
- ganglia-monitor service

VM instance
- ycsb service
- zookeeper service

VM instance
- ganglia-collector service

Multi-DB deploy
Multi-cloud API

Multi-DB API

1. d run-load-phase
2. e run-transactions

3. a/b collect data
3. c push data

Observation data store
Open Challenges & Lessons Learned

- Extensible design for multiple OSs and DBs
  - Hard-coding experiment plans into VMs is too static.
  - Java plus shell scripts: hard to extend and debug.
  - Better but not perfect: Ruby-based tool using Chef or Puppet

- Collaboration
  - Source code management system for experiment module collaboration, e.g. git and github.
  - Cloud storage with version control for data collection, e.g. S3.

- Shared virtualized infrastructure causes side-effects
  - Amazon EC2 network is a shared resource
  - Use large or extra-large instance types for somewhat predictable i/o
  - API unreliability and long provisioning times of some cloud providers other than AWS can prohibit efficient experimentation.
Related Work

Experiment Automation in Grid Computing

- The NMI Test & Build Laboratory for continuous integration of heterogeneous distributed computing systems [Pavlo 2006].
- DiPerF is an automated performance evaluation framework for experiments with client-server systems, such as the metadata and directory service of the Globus Toolkit [Dumitrescu et al. 2004].
- The Weevil framework combines a simulation-based workload generator with a model-based configuration management and system automation approach for experiments with distributed systems on PlanetLab [Wang 2005].

Experiment Automation in Cloud Computing

- A recent publication by IBM Research: CloudBench [Silva et al. 2013]
Conclusion & Outlook

Conclusion

- A basic approach and initial tooling for conducting experiments with distributed database systems in the cloud:
  - Capture experiment plans as Experiment Modules.
  - Run experiments using a multi-cloud and multi-system Elastic Lab.
  - Collaborate using source code management services and cloud storage services.

Outlook

- Develop a set of basic experiments that can be used out-of-the-box.
- Use Response Surface Methodology for more efficient experimentation.
Thank you!

markus.klems@kit.edu
https://github.com/myownthemepark/csde
Research group: eorganization.de