Cloud WorkBench

A Web-Based Framework for Benchmarking Cloud Services

Joel Scheuner

University of Zurich, Switzerland
Cloud Computing -
Essential Characteristics

1. On-demand Self-service

Launch Instance

```
ec2-run-instances
ami-896c96fe
-t t1.micro
-k my-key-pair
```
Cloud Computing - Essential Characteristics

1. On-demand Self-service
2. Rapid Elasticity

3 days

70x
Cloud Computing - Essential Characteristics

1. On-demand Self-service
2. Rapid Elasticity
3. Utility-based Pricing
Cloud Computing - Essential Characteristics

1. On-demand Self-service
2. Rapid Elasticity
3. Utility-based Pricing
4. Resource Pooling
Cloud Computing - Essential Characteristics

1. On-demand Self-service
2. Rapid Elasticity
3. Utility-based Pricing
4. Resource Pooling
5. Broad Network Access
Infrastructure-as-a-Service (IaaS)

- Processing, storage, networks
- Virtual Machines (VMs)

>23 Instance Types

Google Compute Engine

Amazon EC2

Microsoft Azure

GoGrid

Rackspace

Joyent
Differences between IaaS Services

- Performance
- Hardware being served
- Reliability
- Costs

Even for services with the same specification!
Benchmark

- Performance test
- Types
  - Micro-Benchmarks
  - Application Benchmarks
Cloud WorkBench

Open Source

https://github.com/sealuzh/cloud-workbench
Research Questions

I. How can common IaaS cloud benchmarks from literature be defined in a modular and portable manner?

II. How can benchmarks from RQ1 be periodically scheduled and reproducibly executed in cloud environments without manual interaction?
Overall Architecture

- **CWB Server**
  - **Web Interface**
  - **REST**
  - **Business Logic**
    - **VM Environment Manager**
      - **Provider Plugin**
      - **Core**
    - **Scheduler**
  - **Relational Database**

- **Access**
  - **Web Interface**

- **Experimenter**
Overall Architecture
Benchmark Anatomy

- **Timeout**
  - 1
  - *

- **Benchmark Definition**
  - *

- **Schedule**
  - 1
Benchmark Anatomy

```
config.vm.provider :aws do |aws|
  aws.region = "eu-west-1"
  aws.ami = "ami-896c96fe"
  aws.instance_type = "t1.micro"
end

# Update package index
include_recipe 'apt'
# Install benchmark via package manager
package 'sysbench'
```
Benchmark Execution

1. **Experimenter / Scheduler**
   - Trigger Execution

2. **CWB Server**
   - Acquire Resources
   - Provision VM

3. **Provider API**
   - VM Provisioning Completed
   - Start Benchmark Run
   - Notify Benchmark Completed
   - Submit Metric(s)

4. **Cloud VM**
   - Fetch VM Configurations
   - Apply VM Configurations
   - Run Benchmark
   - Postprocess Results

5. **Provisioning Service**
   - Release Resources
Case Study

- Raw sequential write speed
- HDD vs. SSD storage
- Different instance types
Questions

1. When do larger instance types perform better than smaller instance types?

2. When should larger instance types be preferred over the better block storage type?

3. How do instance types and block storage types influence performance variability?
Experiment Setup

- Amazon EC2 Ireland (eu-west-1)
- Ubuntu 14.04 (trusty)
- FIO benchmark
- 8 - 12 repetitions

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>Price per Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1.micro</td>
<td>$0.020</td>
</tr>
<tr>
<td>m1.small</td>
<td>$0.047</td>
</tr>
<tr>
<td>m3.medium</td>
<td>$0.077</td>
</tr>
</tbody>
</table>
When do larger instance types perform better than smaller instance types?

- t1.micro: 750 KB/s (Standard EBS) + 4x
- m1.small: 3500 KB/s (Standard EBS) + 0x
- m3.medium: 3000 KB/s (Standard EBS)
When should larger instance types be preferred over the better block storage type?
How do instance types and block storage types influence performance variability?
How do instance types and block storage types influence performance variability?

<table>
<thead>
<tr>
<th></th>
<th>t1.micro</th>
<th>m1.small</th>
<th>m3.medium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard EBS</td>
<td>20% (20-50%)</td>
<td>20% (10-20%)</td>
<td>30% (15-60%)</td>
</tr>
<tr>
<td>General Purpose EBS</td>
<td>10% (20-40%)</td>
<td>10% (5-15%)</td>
<td>10% (5-10%)</td>
</tr>
</tbody>
</table>

![Graph showing performance variability across different instance types and block storage types.](image)
Conclusion I

How can common IaaS cloud benchmarks from literature be defined in a modular and portable manner?

- Entirely define benchmarks by means of code
- Apply common software engineering techniques
- Make components and benchmarks configurable
How can benchmarks from RQ1 be periodically scheduled and reproducibly executed in cloud environments without manual interaction?

- Use system utilities and existing tools to build a fully automated benchmark execution environment
- Eliminate any error-prone human interactions threatening reproducibility
Outlook

- Large-scale evaluation for a World Wide Web Conference paper
- Extend CWB in a master project
- Web-based benchmarking studio
- Support entire lifecycle