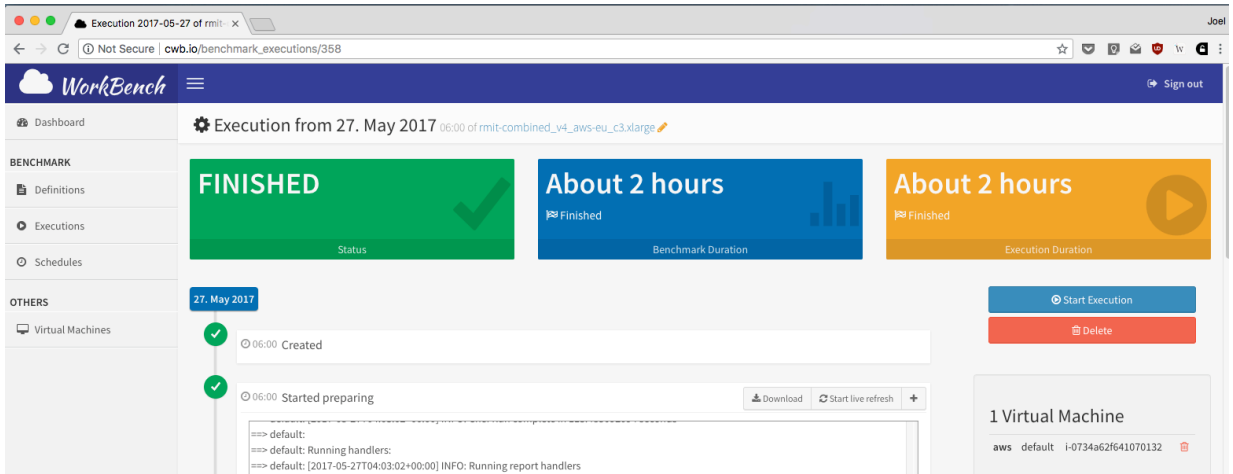
CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

ICET LAB

WASP WALLENBERG AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Performance Benchmarking with Cloud Workbench (CWB)



Presenters

Joel Scheuner

Philipp Leitner

ICET LAB

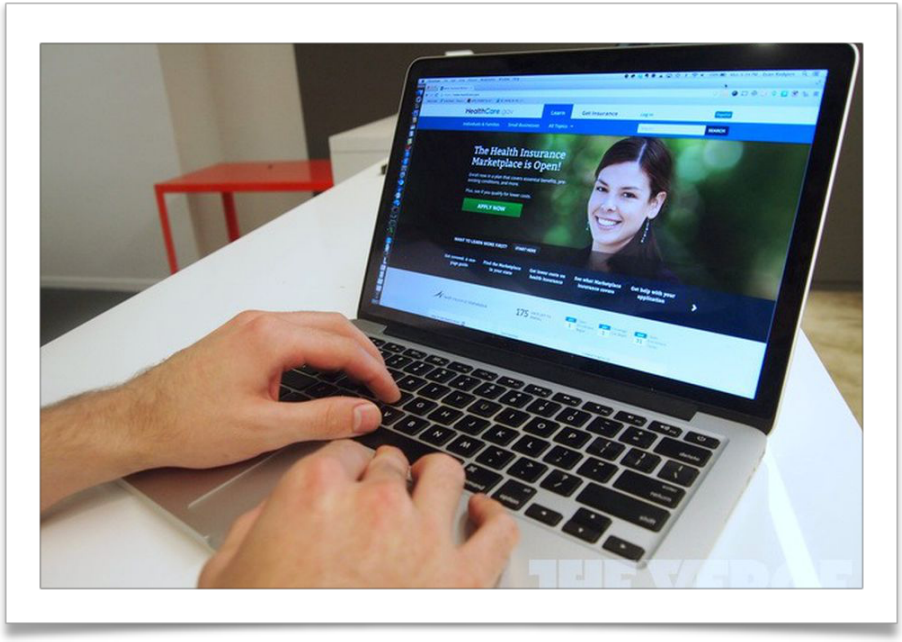
https://icet-lab.eu

@IcetLab

AVANCEZ 1829

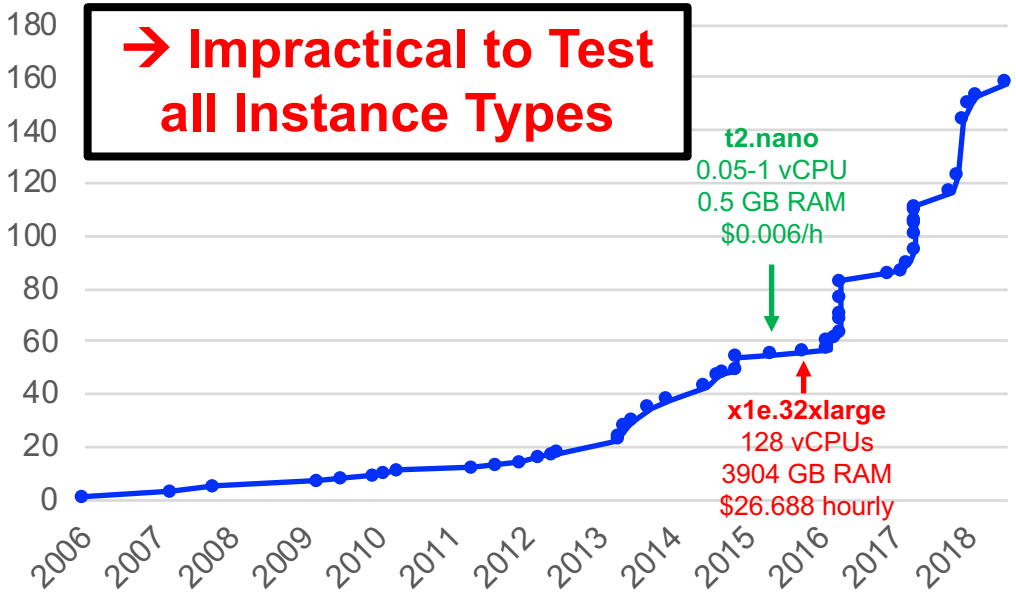
Universität Zürich[UZH]

Performance Matters

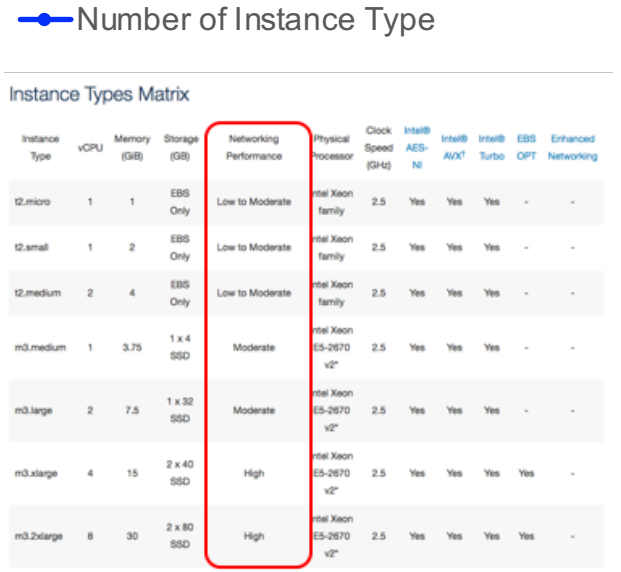


Benchmarking IaaS Clouds

Capacity Planning in the Cloud is hard



Source: https://aws.amazon.com/blogs/aws/ec2-instance-history/



Capacity Planning in the Cloud is hard

NETFLIX

"The instance type itself is a very major tunable parameter"

@brendangregg re:Invent'17

https://youtu.be/89fYOo1V2pA?t=5m4s

What cloud provider should I choose?

Should I go for many small or few large instances?

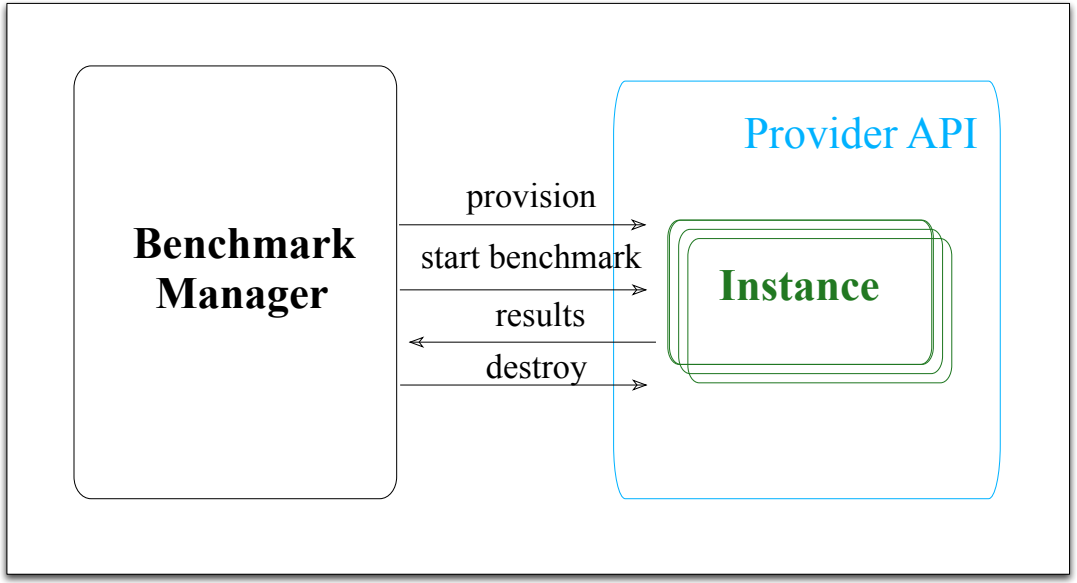
*General-purpose or *-optimized?*

Pay for better IOPS or not?

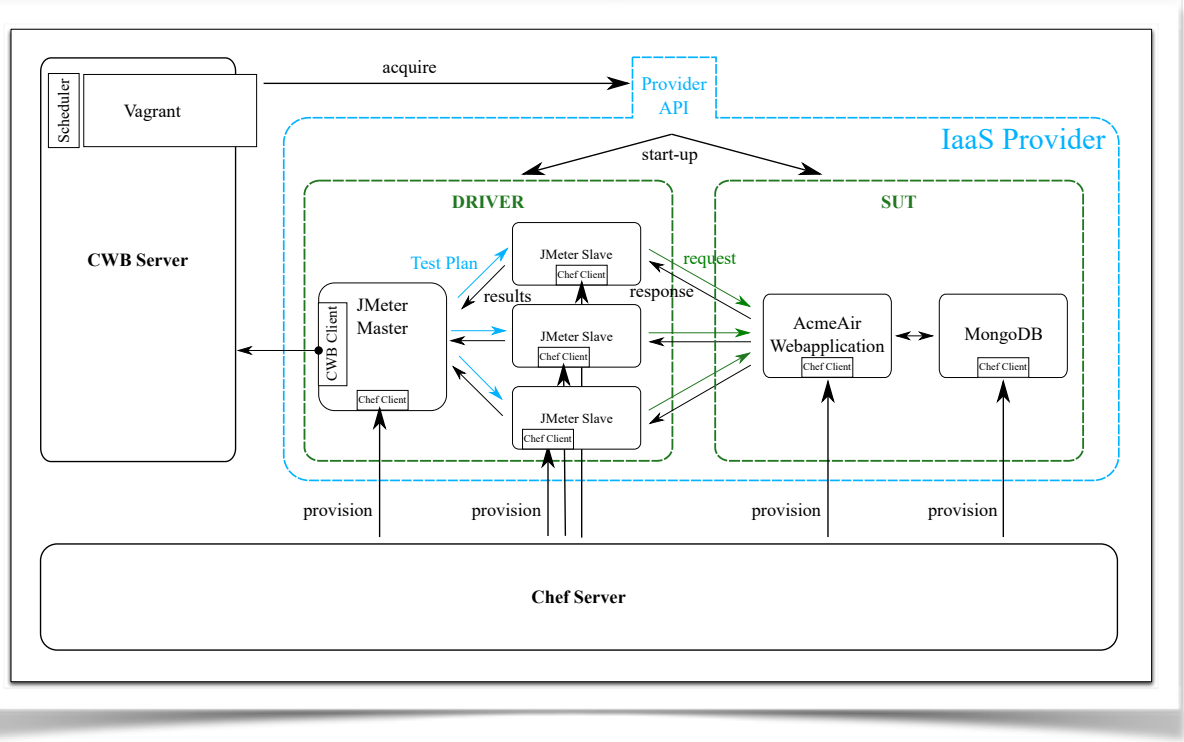
……..……..

➡ **Need for Benchmarking**

Basic Cloud Benchmarking Approach

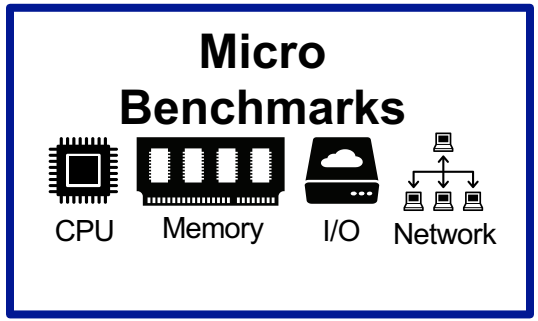


Basic Cloud Benchmarking Approach



CCGrid 2017 "An Approach and Case Study of Cloud Instance Type Selection for Multi-Tier Web Applications"

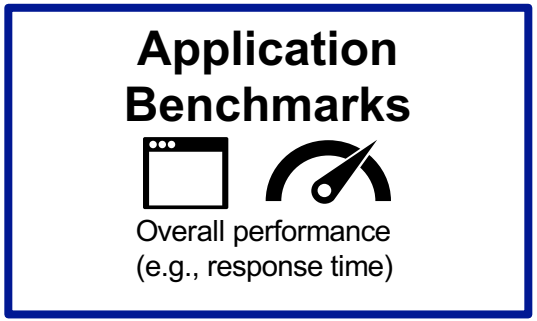
Benchmark Types



Generic

Artificial

Resource-specific



Specific

Real-World

Resource-heterogeneous

Micro Benchmark Examples

File I/O: 4k random read

1) Prepare

2) Run

3) Extract Result

3.5793 MiB/sec

4) Cleanup

Bandwidth

Server

Client

Result

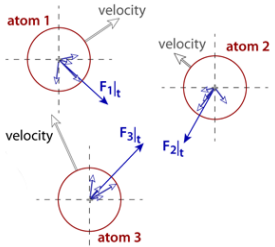
972 Mbits/sec

Application Benchmark Examples

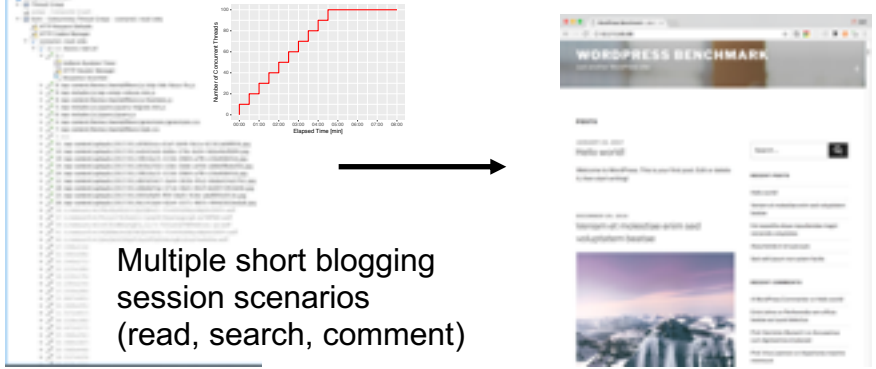
Application Benchmarks

Overall performance
(e.g., response time)

Molecular Dynamics Simulation (MDSim)

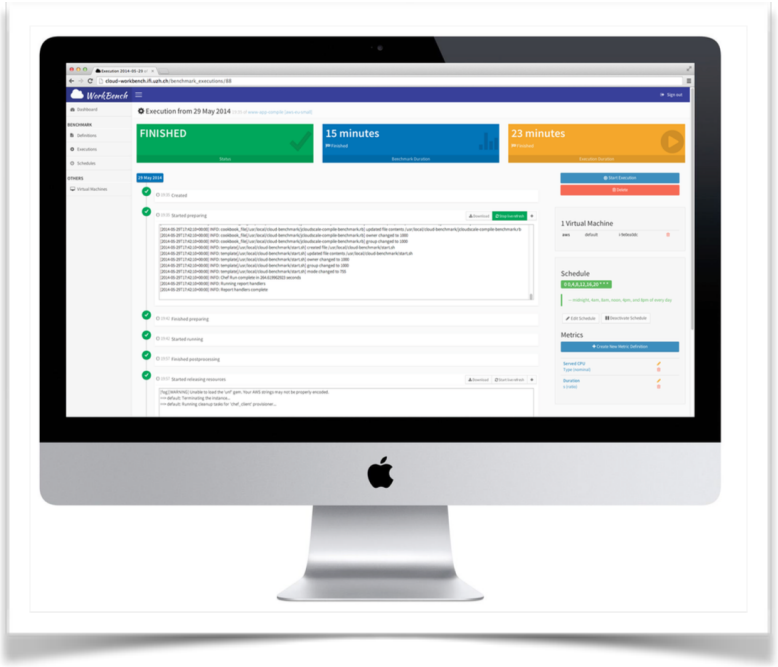


APACHE JMeter™ WordPress Benchmark (WPBench)



Multiple short blogging session scenarios (read, search, comment)

Cloud Workbench



Tool for **scheduling** cloud experiments

Demo:
https://www.youtube.com/watch?v=0yGFGvHvobk

Code:
https://github.com/sealuzh/cloud-workbench

CloudCom 2014 "Cloud Work Bench - Infrastructure-as-Code Based Cloud Benchmarking"

Planned Schedule

- My First CWB Benchmark [~ 30 mins]
- CWB Architecture and Selected Previous Results [~ 30 mins]
- ~ Coffee Break ~ 🎉
- Building and Running a Benchmark from Ground Up [~ 90 mins]
- Wrap-Up and Outlook [5 mins]

My First CWB Benchmark

Interactive Session

Online Material

http://bit.ly/cwb-tutorial

Benchmarking with CWB

1. Write benchmark config to setup environment (optional for simple benchmarks)
2. Declare IaaS resources and parametrize benchmark config
3. Trigger execution or define periodic schedule
4. Download metrics as CSV file
5. Analyze results

1. Write benchmark config

Setup environment

default.rb

```
1   # Chef resources: https://docs.che
2   # Community cookbooks: https://sup
3
4   ### Install
5
6   # apt_update
7   # package 'sysbench'
8
9   ### Configure
10
11  # Declare a CWB benchmark defined
12  cwb_benchmark 'fio-yourname'
13
```

Write CWB execution hook

fio_yourname.rb

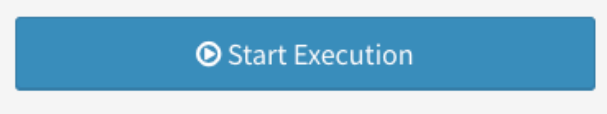
```
1   require 'cwb'
2
3   class FioYourname < Cwb::Benchmark
4     def execute
5       @cwb.submit_metric('cpu_model_name', timestamp, cpu_model_name)
6       result = `date`
7       metric = extract(result)
8       @cwb.submit_metric(metric_name, timestamp, metric)
9     end
```

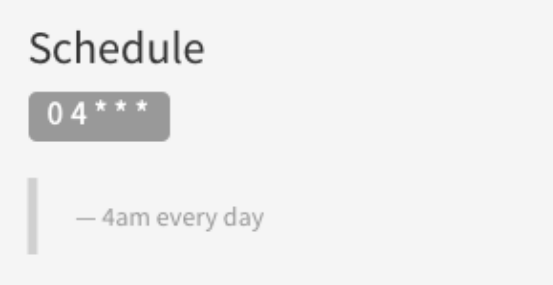
2. Declare IaaS resources and parametrize BM config

```
2   config.vm.provider :aws do |aws, override|
3     aws.region = 'ap-south-1'
4     # Official Ubuntu from Canonical: https://c
    .com/locator/ec2/
5     # 18.04 LTS hvm:ebs-ssd amd64 ap-south-1
6     aws.ami = 'ami-0a63d6cdb8b90bdae'
7     # https://www.ec2instances.info/
8     # Options: t3.nano, t2.nano, t3.micro, t2.m
    .small
9     aws.instance_type = 't3.nano'
10  end
```

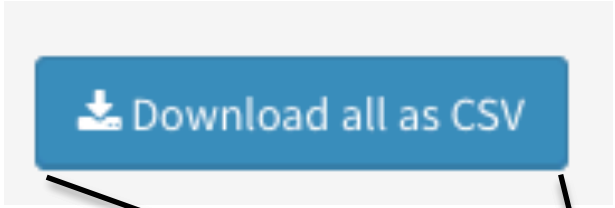
```
12  config.vm.provision 'cwb', type: 'chef_client' do |chef|
13    chef.add_recipe 'fio-joe4dev'
14    chef.json =
15    {
16      'fio' => {
17        'version' => '3.13'
18      },
19    }
20  end
```

3. Trigger or schedule execution





4. Download metrics as CSV file

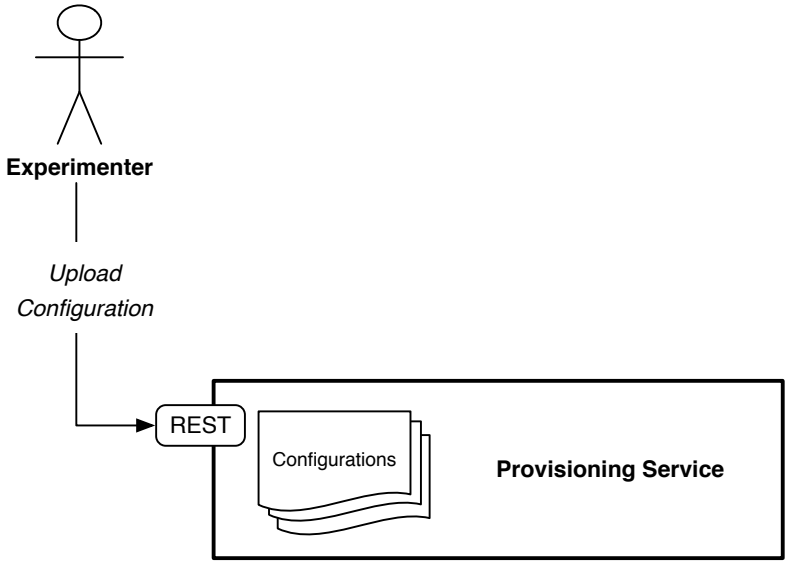


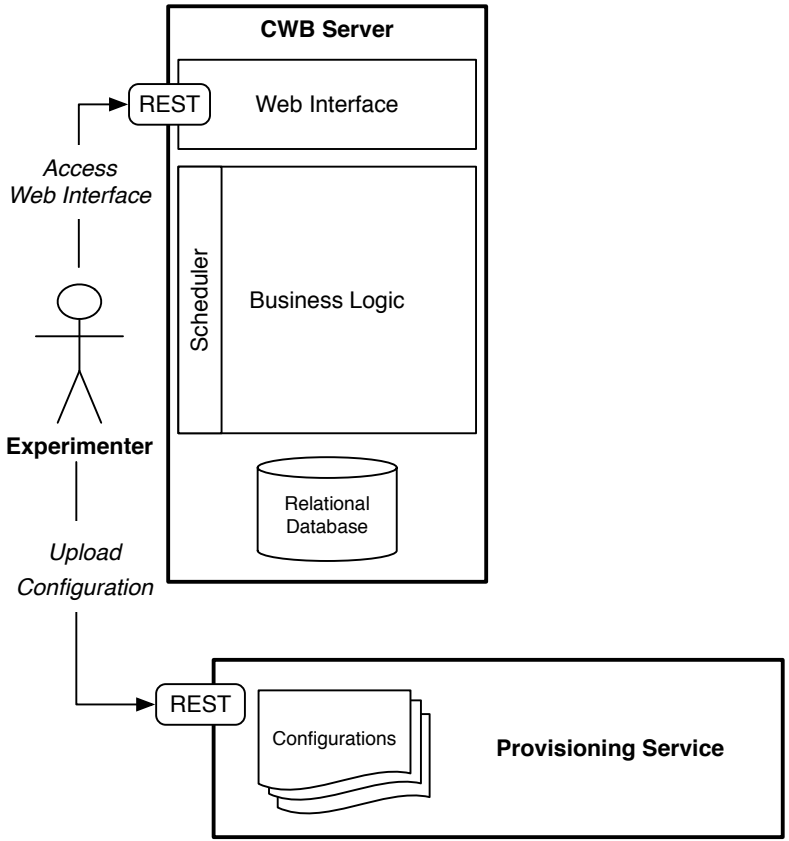
Metric Observations of events_per_second () belonging to execution of 2019-04-02 14:48 of icpe-simple-joe4dev and having *nominal* scale

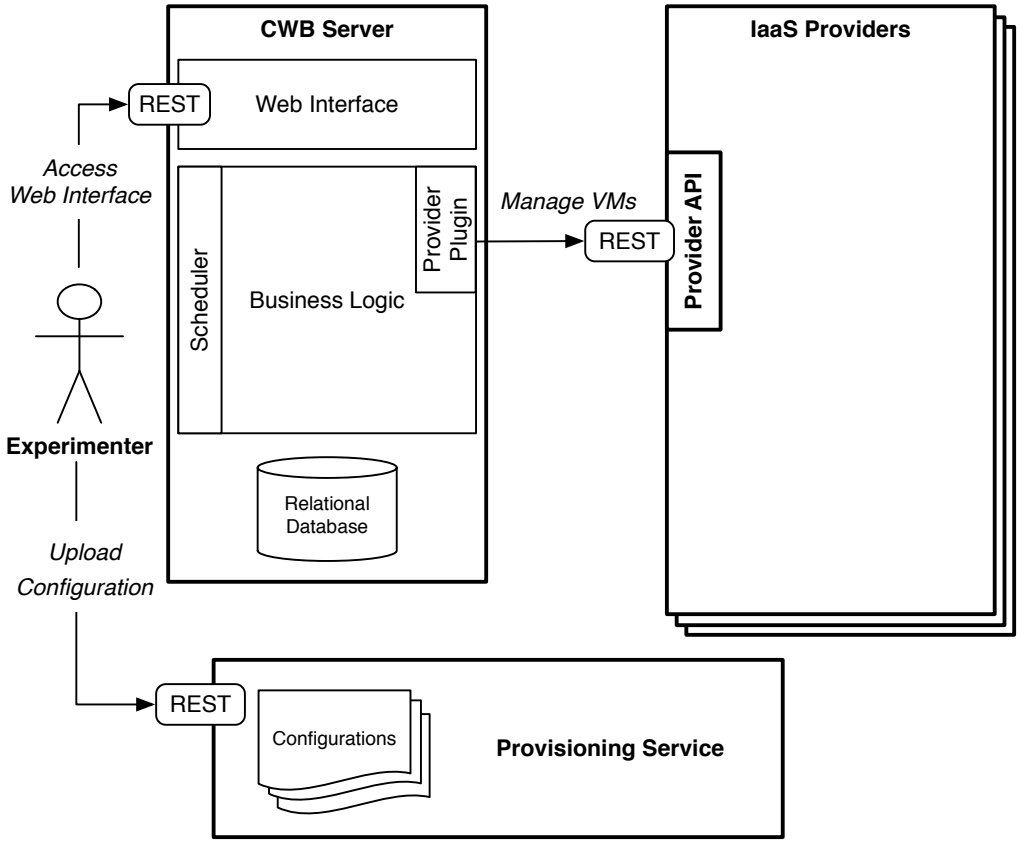
Download all as CSV

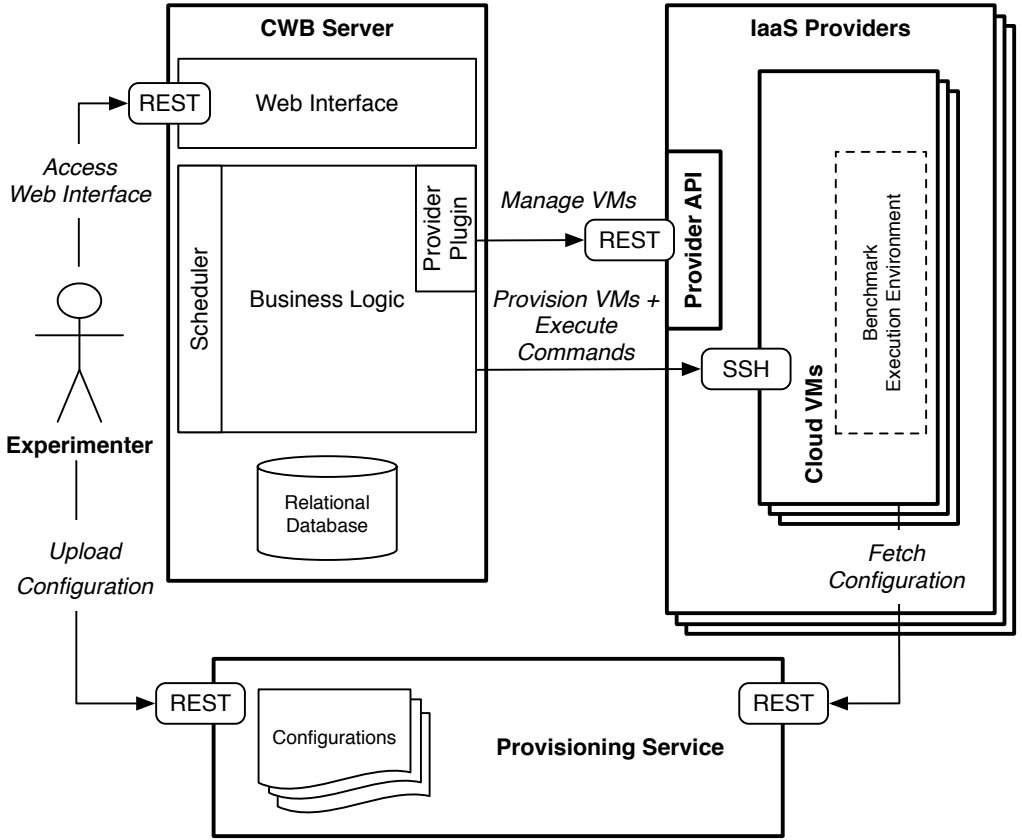
Benchmark Start Time	Provider Name	Provider id	VM Role	Time	Value ()
2019-04-02 12:51:39	aws	i-03c1024580fcea2f1	default	1554209512	335.98
2019-04-02 12:51:39	aws	i-03c1024580fcea2f1	default	1554209522	334.44

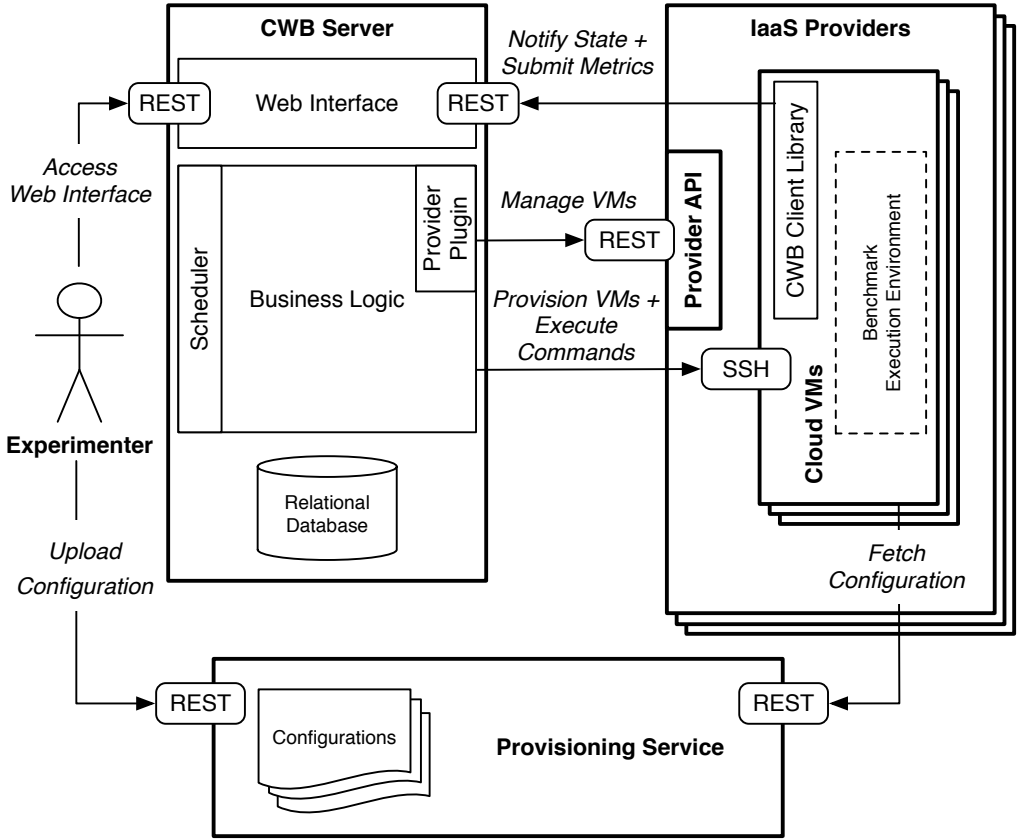
CWB Architecture


Experimenter
Upload
Configuration
REST
Configurations
Provisioning Service


CWB Server
REST
Web Interface
Access
Web Interface
Scheduler
Business Logic
Experimenter
Relational
Database
Upload
Configuration
REST
Configurations
Provisioning Service

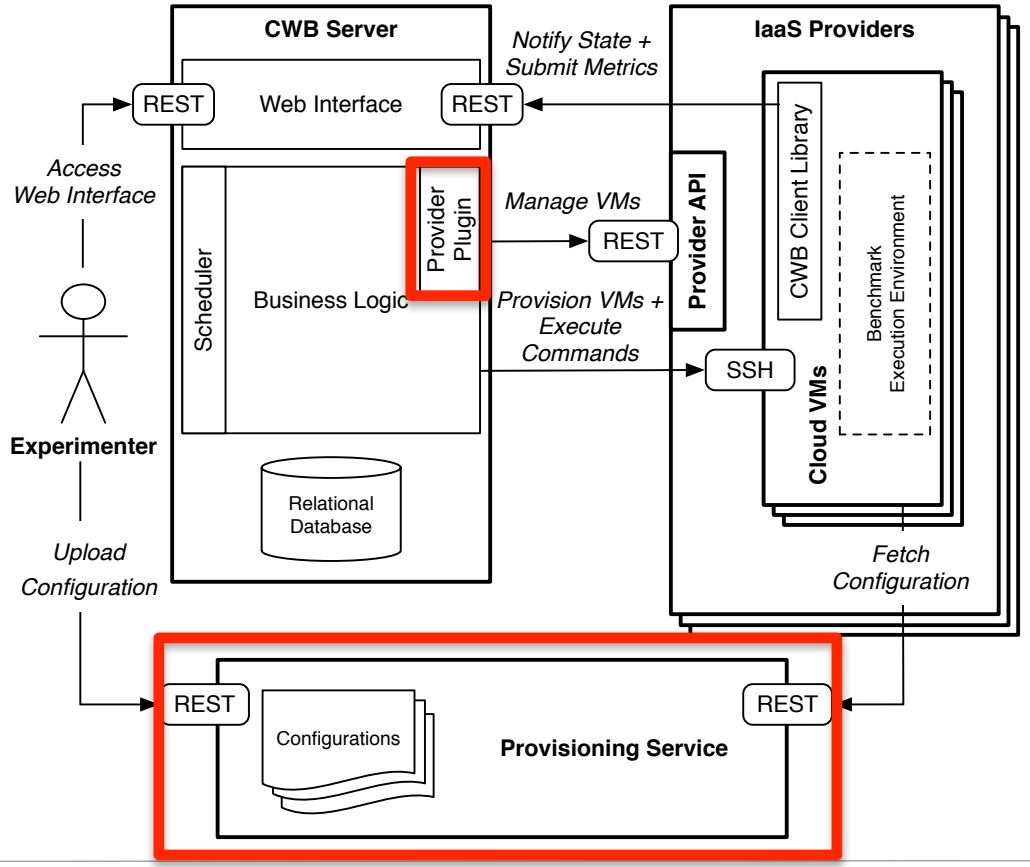

CWB Server
REST
Web Interface
Access
Web Interface
Scheduler
Business Logic
Provider
Plugin
Manage VMs
REST
IaaS Providers
Provider API
Experimenter
Relational
Database
Upload
Configuration
REST
Configurations
Provisioning Service


CWB Server
REST
Web Interface
Access
Web Interface
Scheduler
Business Logic
Provider
Plugin
Experimenter
Relational
Database
Upload
Configuration
Manage VMs
REST
Provision VMs +
Execute
Commands
SSH
IaaS Providers
Provider API
Cloud VMs
Benchmark
Execution Environment
Fetch
Configuration
REST
Configurations
Provisioning Service
REST


CWB Server
Web Interface
REST
REST
Scheduler
Business Logic
Provider Plugin
Relational Database
Notify State + Submit Metrics
Manage VMs
REST
Provision VMs + Execute Commands
SSH
IaaS Providers
Provider API
CWB Client Library
Benchmark Execution Environment
Cloud VMs
Fetch Configuration
Access Web Interface
Experimenter
Upload Configuration
REST
Configurations
Provisioning Service
REST

HashiCorp Vagrant

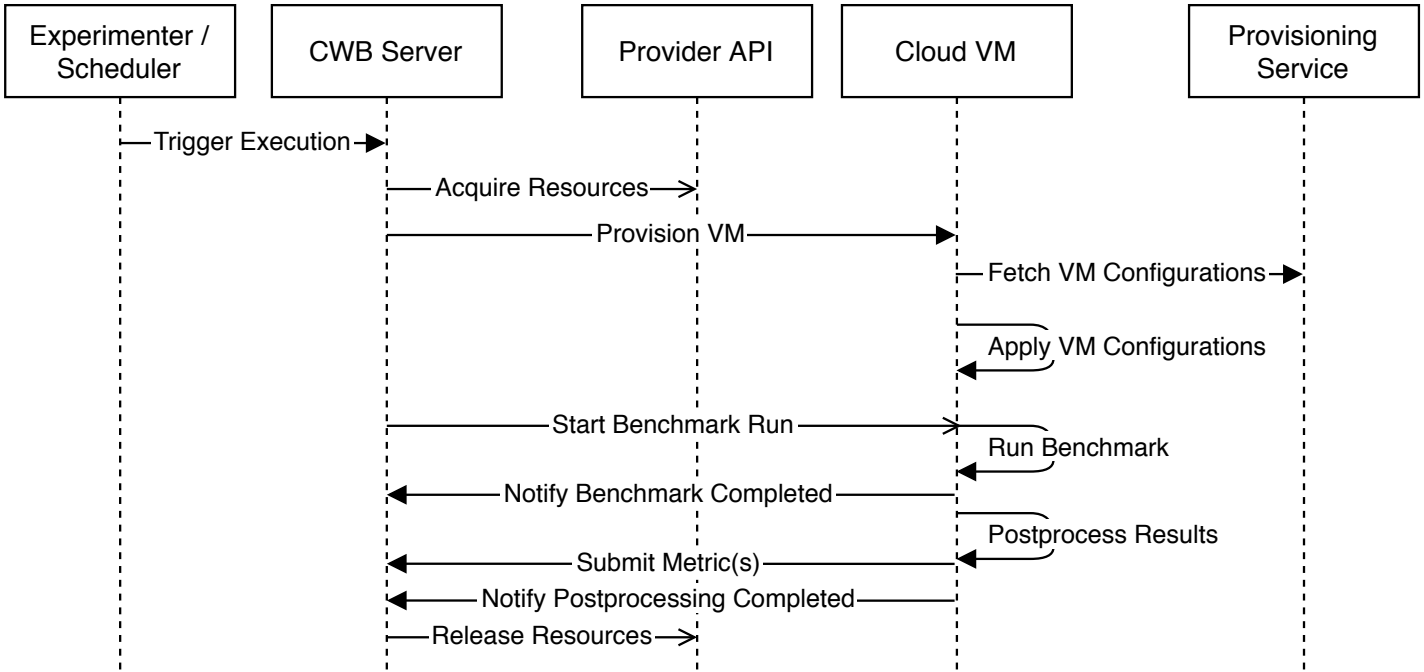
Ruby DSL for defining infrastructure (mostly VMs)



CHEF

Ruby DSL for configuring machines

Benchmark Execution Lifecycle



Selected Previous Results

Example Study 1 - Performance Testing of the Cloud

Study setup

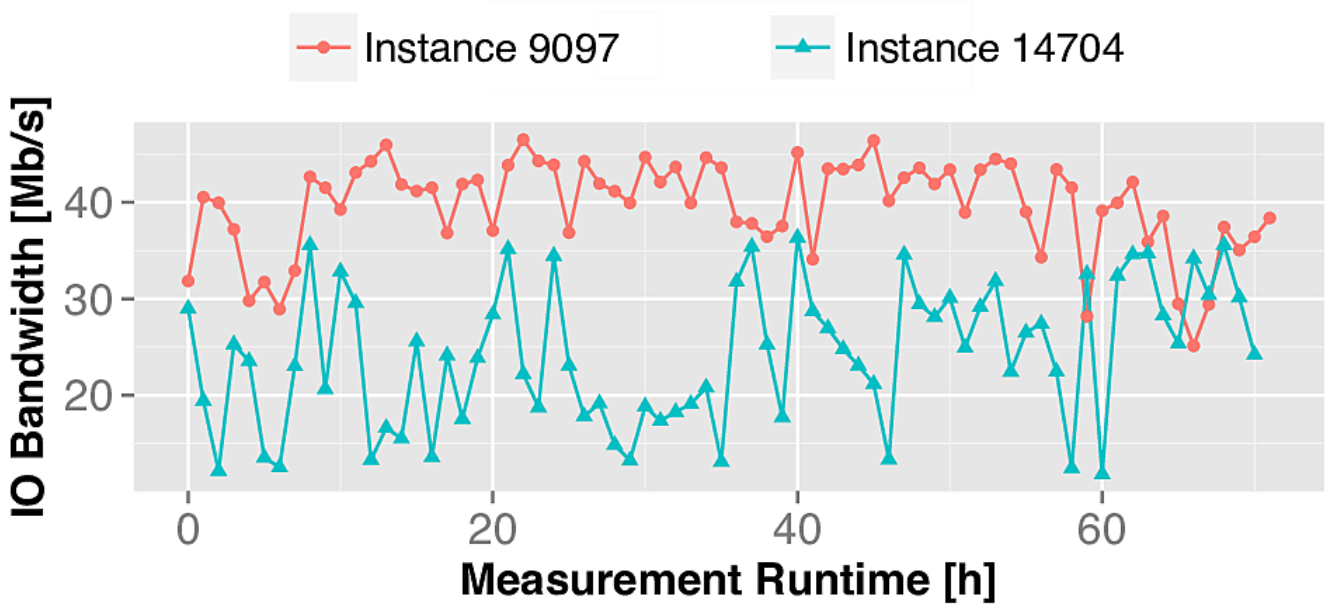
Benchmarked **22 cloud configurations** using **5 benchmarks**

Two types of experiments

Isolated: 300 - 500 repetitions

Continuous: 15 repetitions per configuration

TOIT 2016 "Patterns in the Chaos - A Study of Performance Variation and Predictability in Public IaaS Clouds"


Instance 9097
Instance 14704
IO Bandwidth [Mb/s]
40
30
20
0
20
40
60
Measurement Runtime [h]

Results Summary

			CPU-Bound			IO-Bound	
		Type	CPU	MEM	Java	IO	OLTP
EC2	eu	t1.micro	12.14	17.67	30.63	71.33	30.66
		m1.small	3.19	3.77	3.17	88.49	13.02
		m3.large	0.13	2.07	7.22	35.53	21.26
		c3.large	0.21	8.60	6.42	58.88	21.31
		i2.xlarge	0.12	11.92	8.44	20.07	12.28
	na	t1.micro	20.28	26.40	59.32	70.08	32.18
		m1.small	12.81	26.18	5.34	94.47	15.68
		m3.large	0.16	4.46	9.23	49.02	37.10
GCE	eu	f1-micro	5.28		8.36	3.06	
		n1-standard-1	2.54		6.99	3.36	
		n1-standard-2	1.71		6.96	1.33	
	na	f1-micro	5.13		7.17	9.47	
		n1-standard-1	2.05		8.31	10.39	
		n1-standard-2	1.16		9.53	4.88	
Azure	eu	ExtraSmall	18.38		16.88	61.92	
		Small	18.23		8.37	59.01	
		Medium	17.81		11.91	47.14	
	na	ExtraSmall	18.13		15.96	49.01	
		Small	19.11		6.62	44.01	
		Medium	18.28		10.96	48.31	
SL	na	1 CPU / 2048 MB	0.11		6.65	13.01	
		2 CPUs / 4096 MB	0.11		7.14	6.27	

TOIT 2016 "Patterns in the Chaos - A Study of Performance Variation and Predictability in Public IaaS Clouds"

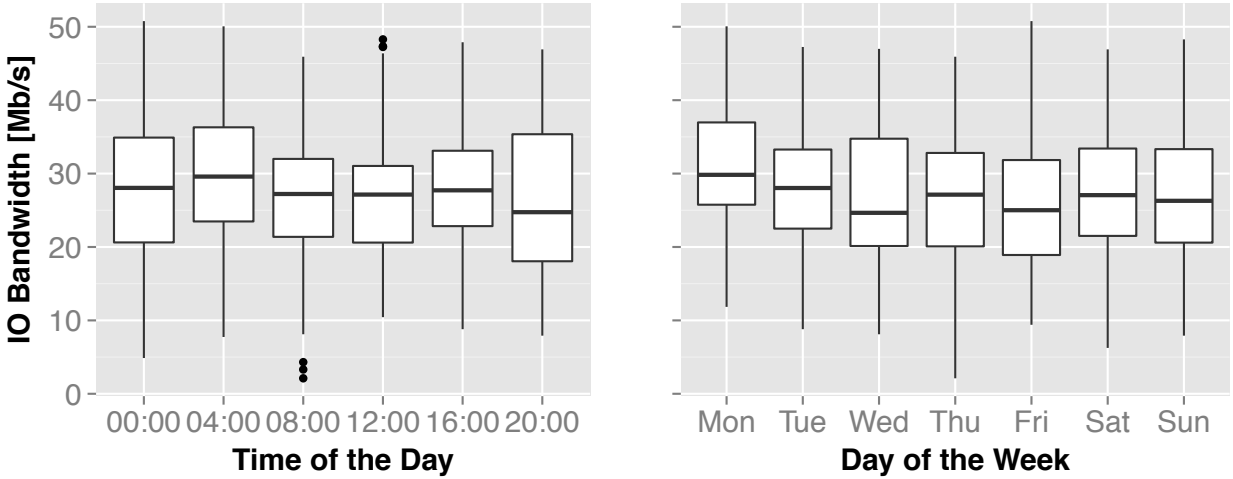
Observed CPU Models

	Model	#
EC2	Intel E5-2650	1962
	Intel E5430	364
	Intel E5645	293
	Intel E5507	84
	Intel E5-2651	32
	AMD 2218 HE	9
Azure	AMD 4171 HE	782
	Intel E5-2673	595
	Intel E5-2660	189

(for m1.small and Azure Small in North America)

TOIT 2016 "Patterns in the Chaos - A Study of Performance Variation and Predictability in Public IaaS Clouds"

Impact of Different Days / Times



(for m3.large in Europe)

TOIT 2016 "Patterns in the Chaos - A Study of Performance Variation and Predictability in Public IaaS Clouds"

		Type	CPU-Bound CPU	CPU-Bound MEM	CPU-Bound Java	IO-Bound IO	IO-Bound OLTP
EC2	eu	t1.micro	12.14	17.67	30.63	71.33	30.66
		m1.small	3.19	3.77	3.17	88.49	13.02
		m3.large	0.13	2.07	7.22	35.53	21.26
		c3.large	0.21	8.60	6.42	58.88	21.31
		i2.xlarge	0.12	11.92	8.44	20.07	12.28
	na	t1.micro	20.28	26.40	59.32	70.08	32.18
		m1.small	12.81	26.18	5.34	94.47	15.68
		m3.large	0.16	4.46	9.23	49.02	37.10
GCE	eu	f1-micro	5.28		8.36	3.06	
		n1-standard-1	1.5		6.99	3.36	
		n1-standard-2	0.77		6.96	1.33	
	na	f1-micro	2.13		7.17	9.47	
		n1-standard-1	2.05		8.31	10.39	
		n1-standard-2	1.16		9.53	4.88	
Azure	eu	ExtraSmall	18.38		16.88	61.92	
		Small	18.23		8.37	59.01	
		Medium	17.81		11.91	47.14	
	na	ExtraSmall	18.13		15.96	49.01	
		Small	19.11		6.62	44.01	
		Medium	18.28		10.96	48.31	
SL	na	1 CPU / 2048 MB	0.11		6.65	13.01	
		2 CPUs / 4096 MB	0.11		7.14	6.27	

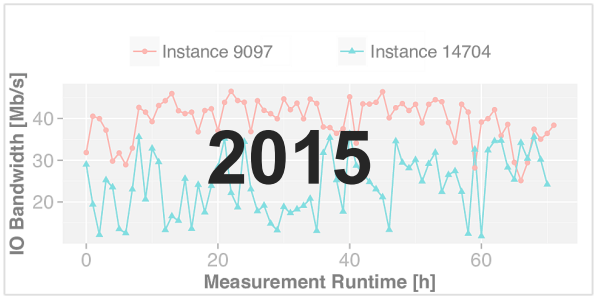
2015

Recent Results

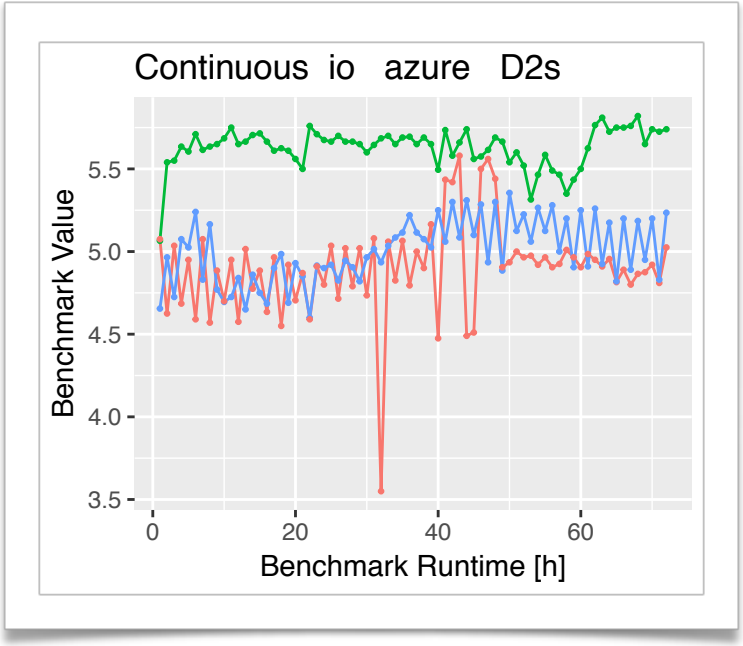
(Feb 2019)

	Type	CPU	IO	MEM
EC2	t3-nano	2.33	13.58	1.64
	t3-medium	3.85	11.09	2.18
	m5-large	1.59	12.77	1.55
	c5-large	0.15	13.94	2.56
GCE	f1-micro	8.65	7.21	16.96
	standard-1	0.57	7.33	18.4
	standard-2	1.3	29.37	5.27
	highcpu-1	1.39	7.39	4.3
AZ	B1s	4.43	7.3	5.33
	D2s	6.82	33.29	6.67
	F2s	6.55	31.78	6.26

(unpublished data)



Instance Runtime (Feb 2019)



(unpublished data)

Example Study 2 - Estimating Application Performance from Microbenchmarks

Research question:

How accurately can we predict application performance from system-level microbenchmarks?

Study setup:

2 applications (Wordpress, Molecular Dynamics Simulation)

23 microbenchmarks

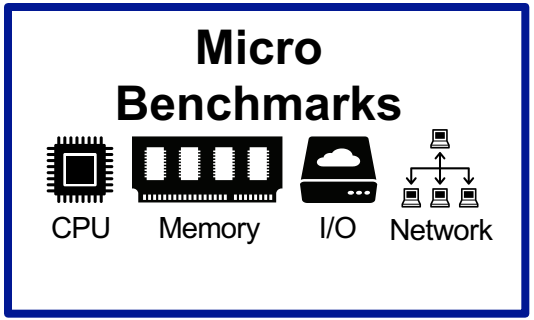
Study executed in AWS (11 instance types)

Linear regression for prediction

Joel Scheuner, Philipp Leitner (2018). Estimating Cloud Application Performance Based on Micro-Benchmark Profiling. IEEE CLOUD.

Micro Benchmarks

Broad resource coverage and specific resource testing



CPU
- sysbench/cpu-single-thread
- sysbench/cpu-multi-thread
- stressng/cpu-callfunc
- stressng/cpu-double
- stressng/cpu-euler
- stressng/cpu-ftt
- stressng/cpu-fibonacci
- stressng/cpu-int64
- stressng/cpu-loop
- stressng/cpu-matrixprod

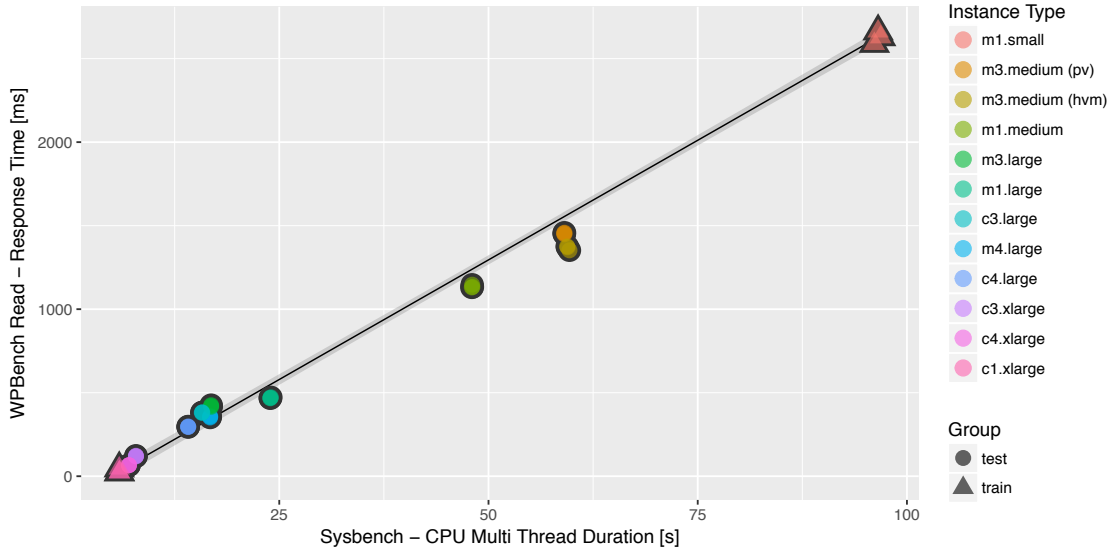
Memory
- sysbench/memory-4k-block-size
- sysbench/memory-1m-block-size

I/O
- [file I/O] sysbench/fileio-1m-seq-write
- [file I/O] sysbench/fileio-4k-rand-read
- [disk I/O] fio/4k-seq-write
- [disk I/O] fio/8k-rand-read

Network
- iperf/single-thread-bandwidth
- iperf/multi-thread-bandwidth
- stressng/network-epoll
- stressng/network-icmp
- stressng/network-sockfd
- stressng/network-udp

Joel Scheuner, Philipp Leitner (2018). Estimating Cloud Application Performance Based on Micro-Benchmark Profiling. IEEE CLOUD.

Example Results: Predicting Wordpress



Joel Scheuner, Philipp Leitner (2018). Estimating Cloud Application Performance Based on Micro-Benchmark Profiling. IEEE CLOUD.

Example Study 3 - Software Performance Testing in the Cloud

Research question:

Executed 19 software performance tests in different environments

How small performance regressions can we find?

Study setup:

4 open source projects in Java and Go

Study executed in AWS, Azure, Google

Baseline: baremetal server in Softlayer / Bluemix

Christoph Laaber, Joel Scheuner, Philipp Leitner (2019). Software Microbenchmarking in the Cloud. How Bad is it Really? Empirical Software Engineering (EMSE). To appear.

Background - JMH Microbenchmarks

```
@State(Scope.Thread)
public class ComputationSchedulerPerf {

    @State(Scope.Thread)
    public static class Input
      extends InputWithIncrementingInteger {
        @Param({ "100" })
        public int size;
    }

    @Benchmark
    public void observeOn(Input input) {
        LatchedObserver<Integer> o =
          input.newLatchedObserver();
        input.observable.observeOn(
          Schedulers.computation()
        ).subscribe(o);
        o.latch.await();
    }
}
```

Listing 1: JMH example from the *RxJava* project.

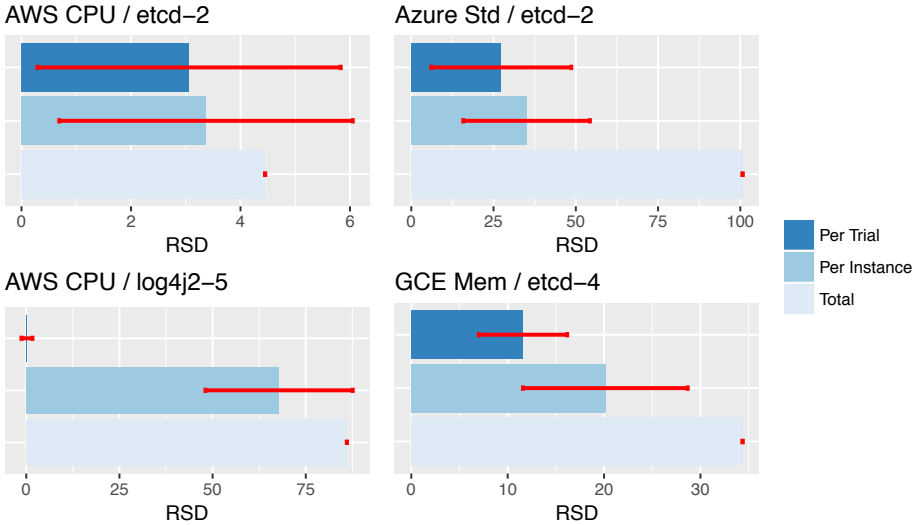
MSR'18. An Evaluation of Open-Source Software Microbenchmark Suites for Continuous Performance Assessment.

Summary - Variability of Software Benchmark Results

Benchs	AWS			GCE			Azure			BM
	Std	CPU	Mem	Std	CPU	Mem	Std	CPU	Mem	
log4j2-1	45.41	42.17	48.53	41.40	43.47	44.38	46.19	40.79	51.79	41.95
log4j2-2	7.90	4.89	3.92	10.75	9.71	11.29	6.18	6.06	11.01	3.83
log4j2-3	4.86	3.76	2.53	10.12	9.18	10.15	13.89	7.55	15.46	3.02
log4j2-4	3.67	3.17	4.60	10.69	9.47	10.52	17.00	7.79	19.32	6.66
log4j2-5	76.75	86.02	88.20	83.42	82.44	80.75	82.62	86.93	82.07	77.82
rxjava-1	0.04	0.04	0.05	0.04	0.04	0.04	0.05	0.05	0.27	0.03
rxjava-2	0.70	0.61	1.68	5.73	4.90	6.12	9.42	6.92	13.38	0.49
rxjava-3	2.51	3.72	1.91	8.16	8.28	9.63	6.10	5.81	10.32	4.14
rxjava-4	4.55	4.18	7.08	8.07	10.46	8.82	17.06	10.22	21.09	1.42
rxjava-5	5.63	2.81	4.04	14.33	11.39	13.11	61.98	64.24	21.69	1.76
bleve-2	1.57	1.32	4.79	5.56	6.09	5.78	5.97	5.48	13.29	0.27
bleve-3	1.13	7.53	7.77	10.08	10.74	14.42	7.62	6.12	14.41	0.18
bleve-4	4.95	4.38	5.17	11.24	12.00	14.52	8.18	7.11	15.24	0.62
bleve-5	10.23	9.84	8.18	57.60	58.42	59.32	52.29	46.40	52.74	10.16
etcd-1	1.03	3.17	1.56	6.45	5.21	7.62	6.36	4.89	11.46	0.15
etcd-2	4.06	4.45	6.28	66.79	69.07	69.18	100.68	94.73	90.19	29.46
etcd-3	1.25	0.69	1.24	7.15	6.57	9.27	4.95	4.31	9.89	0.14
etcd-4	6.80	6.00	7.34	34.53	34.34	34.37	12.28	12.39	22.92	8.09
etcd-5	43.59	22.46	43.44	27.21	27.86	27.17	30.54	31.40	24.98	23.73

Christoph Laaber, Joel Scheuner, Philipp Leitner (2019). Software Microbenchmarking in the Cloud. How Bad is it Really? Empirical Software Engineering (EMSE). To appear.

Sources of Variability



Christoph Laaber, Joel Scheuner, Philipp Leitner (2019). Software Microbenchmarking in the Cloud. How Bad is it Really? Empirical Software Engineering (EMSE). To appear.

Example Study 4 - Credit-based Bursting Instance Types

Research question:

How do t2 instance types perform in terms of CPU and IO speed in comparison to other instances?
When are t2 bursting instance types more cost-efficient than other instance types?
How do t2 instance types perform in comparison to the previous generation (t1) types?

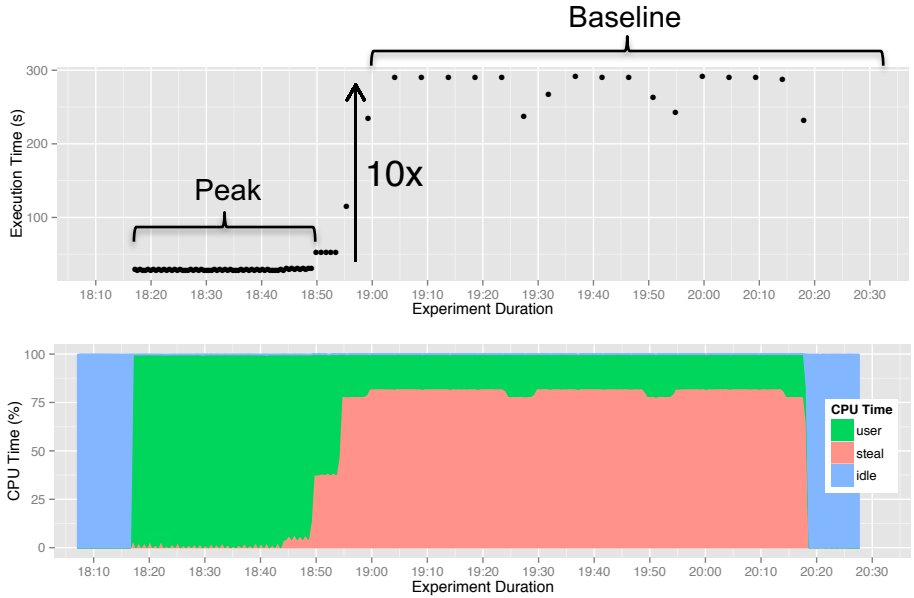
Study setup:

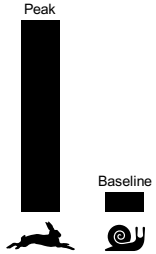
Sysbench CPU and IO benchmarks

Study executed in AWS
3 bursting vs non-bursting instance types

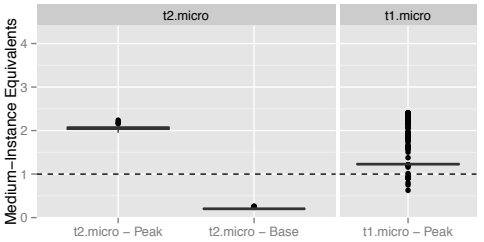
Philipp Leitner, Joel Scheuner (2015). Bursting With Possibilities – an Empirical Study of Credit-Based Bursting Cloud Instance Types (UCC).

Example Study 4 - Credit-based Bursting Instance Types





Burstable T2.* highly predictable unlike previous generation T1.*



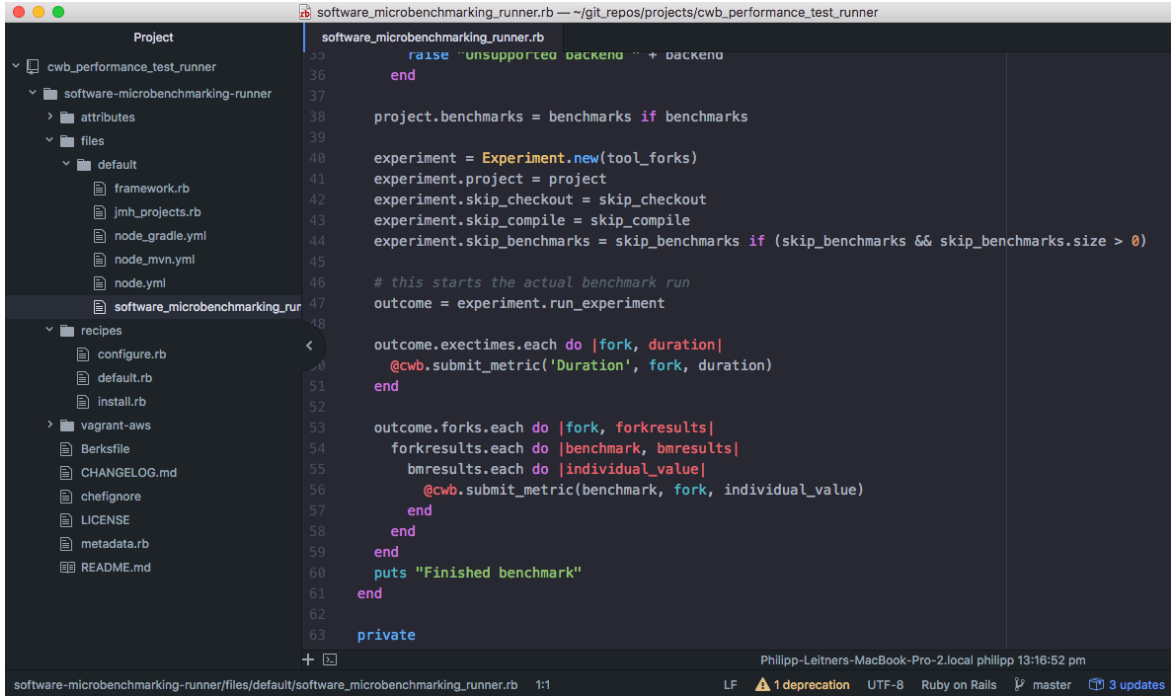
Building and Running a Benchmark from Ground Up

Interactive Session

http://bit.ly/cwb-tutorial

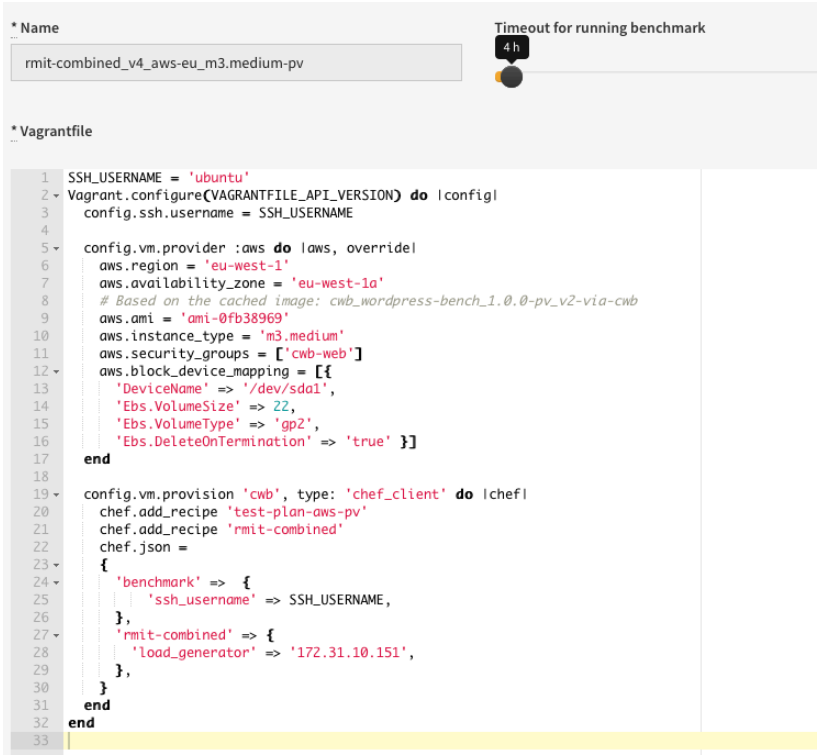
Wrap-Up and Outlook

Three Steps Towards a Benchmark

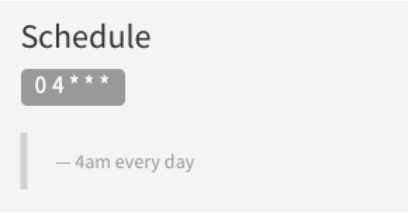


(Optional) Step 1: Write Chef Cookbook

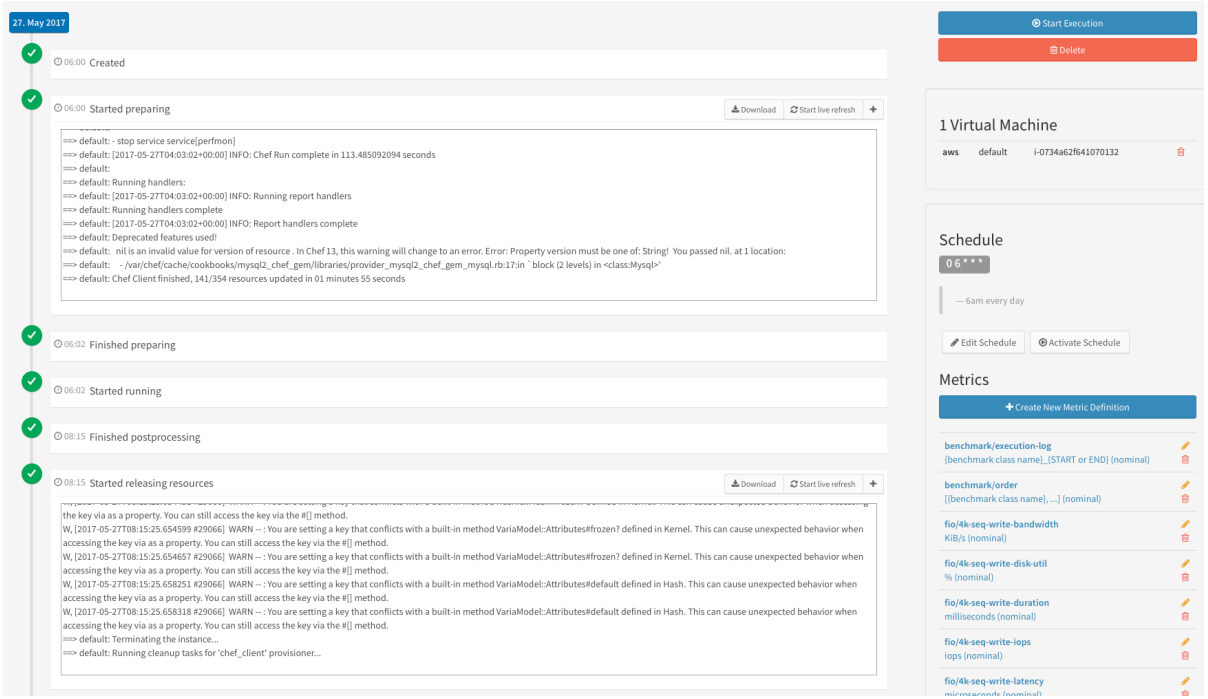
Three Steps Towards a Benchmark



Step II: Define IaaS config and schedule

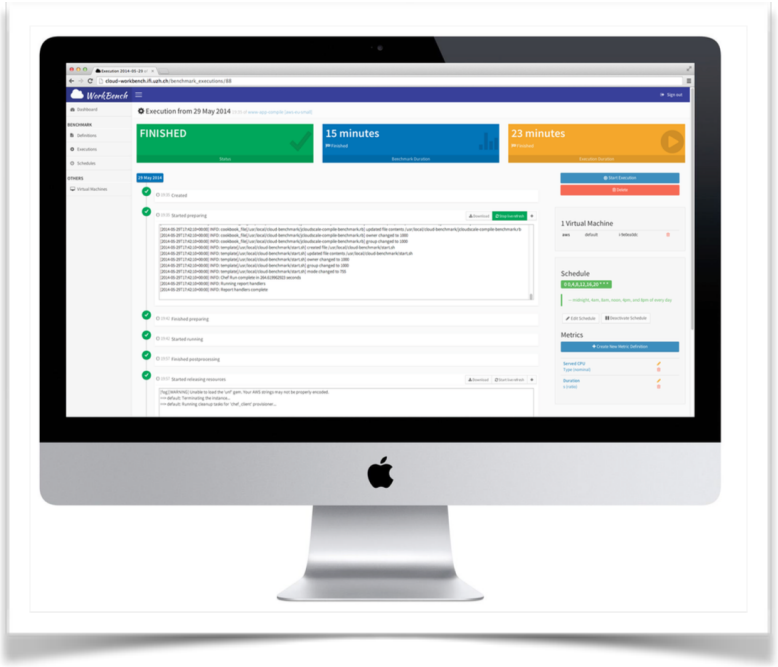


Three Steps Towards a Benchmark



Step III: Execute and download result CSV

Cloud Workbench



Tool for **scheduling** cloud experiments

Demo:
https://www.youtube.com/watch?v=0yGFGvHvobk

Code:
https://github.com/sealuzh/cloud-workbench

CloudCom 2014 "Cloud Work Bench - Infrastructure-as-Code Based Cloud Benchmarking"