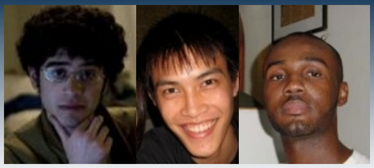
Rain A Sophisticated Workload Generation Toolkit for Cloud Computing Applications

Aaron Beitch, Timothy Yung, Rean Griffith



Overview of workload generation

Managing **workload variations** is key to **dynamic resource allocation** in the Cloud.

We identify three classes of workload variation:

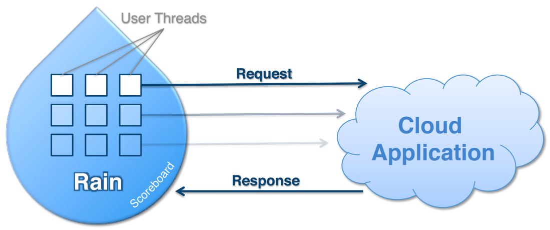
- **Load Amount** (e.g. diurnal patterns, ramps, etc.)
- **Operation Mix** (e.g. changes in feature-popularity or transitions between read- and write-heaviness)
- **Data popularity** (e.g. hotspots)

Optimized dynamic resource allocation requires:

- Scaling up/down to meet changes in load and avoid SLA penalties.
- Reducing power consumption and costs.
- Consolidating work to improve cluster utilization.

Rain workload generation toolkit

Rain is a statistics-based workload generation toolkit that uses parameterized and empirical distributions to model the different classes of workload variations.

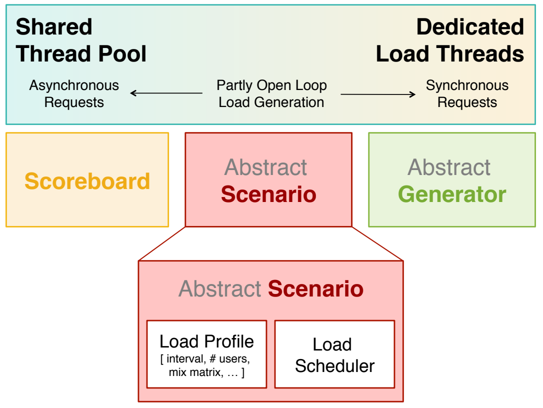


Problems with existing tools

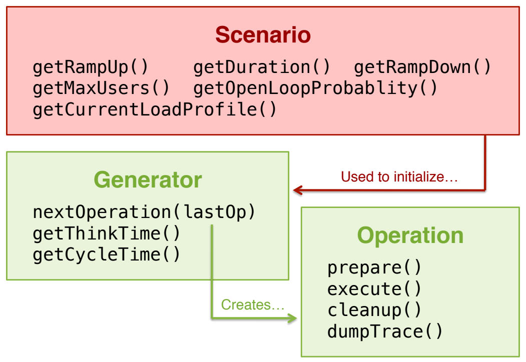
Current workload generation tools fall short of allowing application developers to explore how these workload variations affect their systems and how they can improve their system's reactions.

	TPC-W	Rubis	Faban (1.0)	Rain
Variable load	No	No	Yes	Yes
Variable mix	No	No	No	Yes
Variable data	No	No	No	Yes

Benchmark architectural diagram



APIs of core classes



Scenario

```
getRampUp()     getDuration()  getRampDown()
getMaxUsers()   getOpenLoopProbablity()
getCurrentLoadProfile()
```

Generator

```
nextOperation(lastOp)
getThinkTime()
getCycleTime()
```

Operation

```
prepare()
execute()
cleanup()
dumpTrace()
```

Features and design decisions

- Modular design of the harness
  - Generators can easily use **parameterized** or **empirical** probability distributions to characterize different variations in load
- Supports closed, open, and partly open loop workload generation
  - Separation of request generation and execution
  - Use of the Command pattern to implement Operations (GoF)

Preliminary Results

Implemented Rain prototype

- ~1813 lines of Java code

Created Cloudstone Workload Generator

- Ported Olio UI Driver previously used in Faban
- ~2841 lines of Java code

Abs. Generator

`CloudstoneGenerator`

Abs. Operation

```
CloudstoneOperation
– HomePageOperation
– LoginOperation
– TagSearchOperation
– EventDetailOperation
– PersonDetailOperation
– AddPersonOperation
– AddEventOperation
```

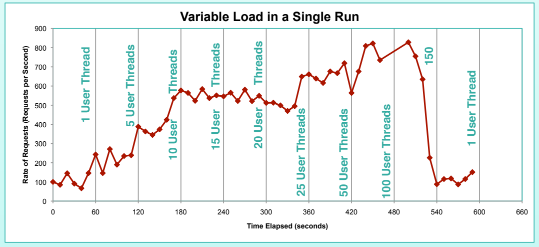
*Olio is a Web 2.0 social-events application. The implementation used here is written in Ruby.

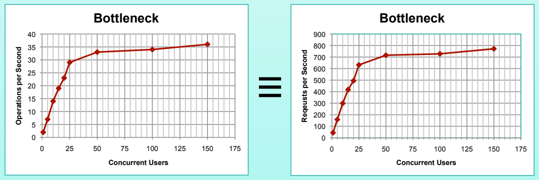
Experiments and results

Cluster Configuration (Amazon EC2):

1 Rails (c1.xlarge), 1 MySQL (c1.xlarge), 1 Rain (c1.xlarge)
1 Nginx/HAProxy (m1.small)

1 – 150 Users, Maximum Load (0 Think Time)





Future Work

Find traces to characterize interesting load variations.
Generate foreground and background workloads.
A Webrat-like library to create new workloads.
Integrate with SCADS and MapReduce request generators.