BIAS Autoscaler: Leveraging Burstable Instances for Cost-Effective Autoscaling on Cloud Systems

Jaime Dantas
jaimecjd@yorku.ca

Dr. Hamzeh Khazaei
hkh@yorku.ca

Dr. Marin Litoiu
mlitoiu@yorku.ca

7th International Workshop on Serverless Computing (WoSC7)
December 7, 2021

Jaime Dantas, Hamzeh Khazaei and Marin Litoiu. "BIAS Autoscaler: Leveraging Burstable Instances for Cost-Effective Autoscaling on Cloud Systems", In *7th International Workshop on Serverless Computing, 2021, doi: 10.1145/3493651.3493667*

https://www.serverlesscomputing.org/wosc7/papers/p2

Performant and Available Computing Systems (PACS) Lab

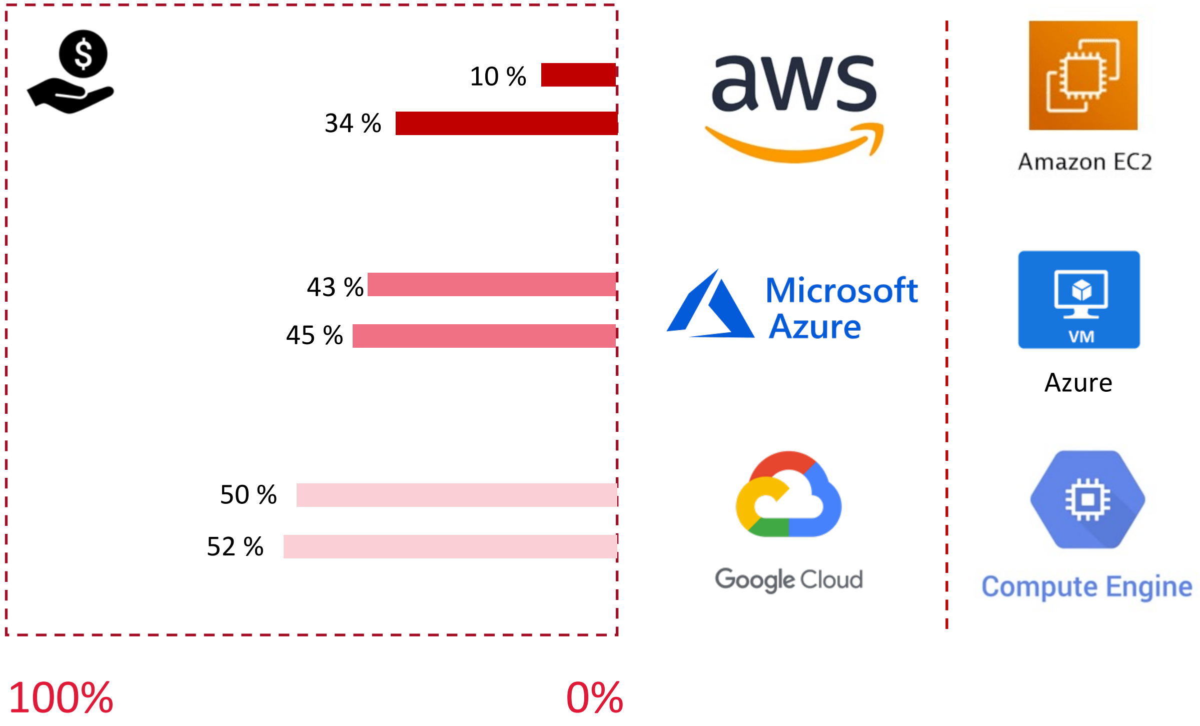
PACS

YORK UNIVERSITÉ UNIVERSITY

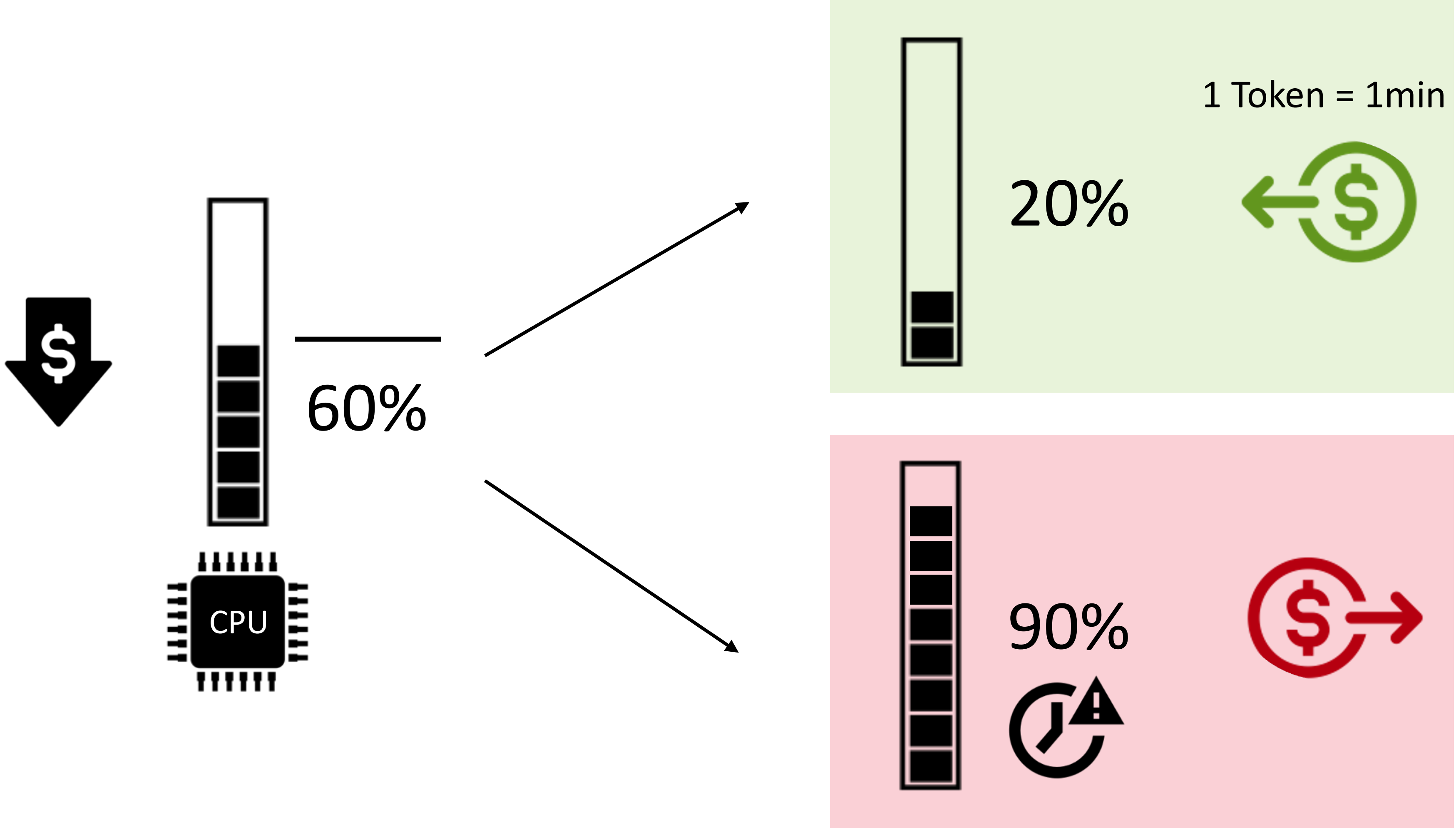
Problem & Motivation

- Underutilization of cloud resources
- Many instance types to choose
- Difference performance and pricing for each instance type (up to 10 times)
- No open-sourced autoscaler available for Google Cloud or Microsoft Azure for burstable instances

Burstable vs regular instances



Burstable Instances



60%

CPU

1 Token = 1min

20%

90%

aws

Google Cloud

Microsoft Azure

System Design

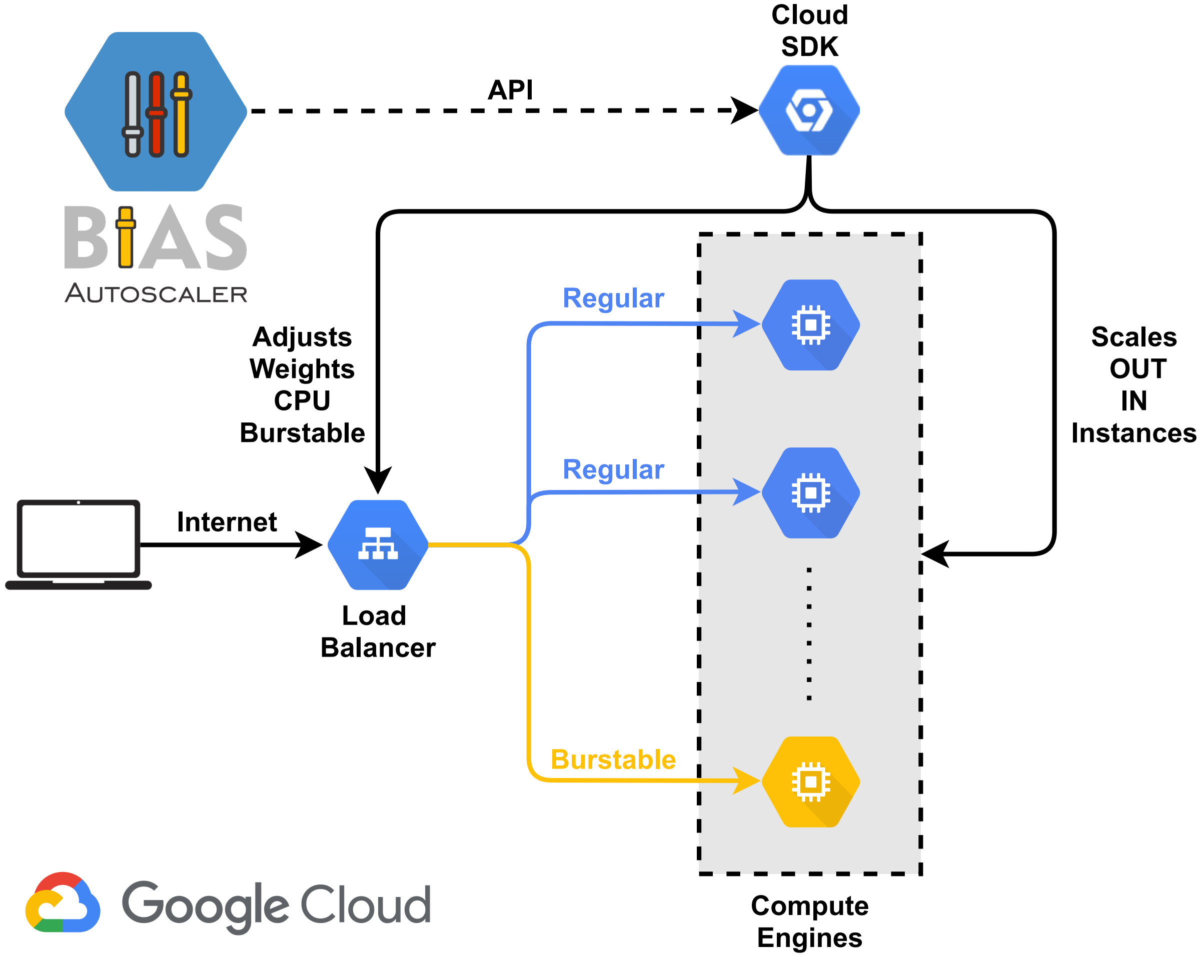
- Uses the GCP Load Balancer
- Supports customized scaling policies
- 4 internal APIs for manual controlling
- Can be extended to Google Kubernetes Engine to manage container-based applications
- Open source

Stack:

Java

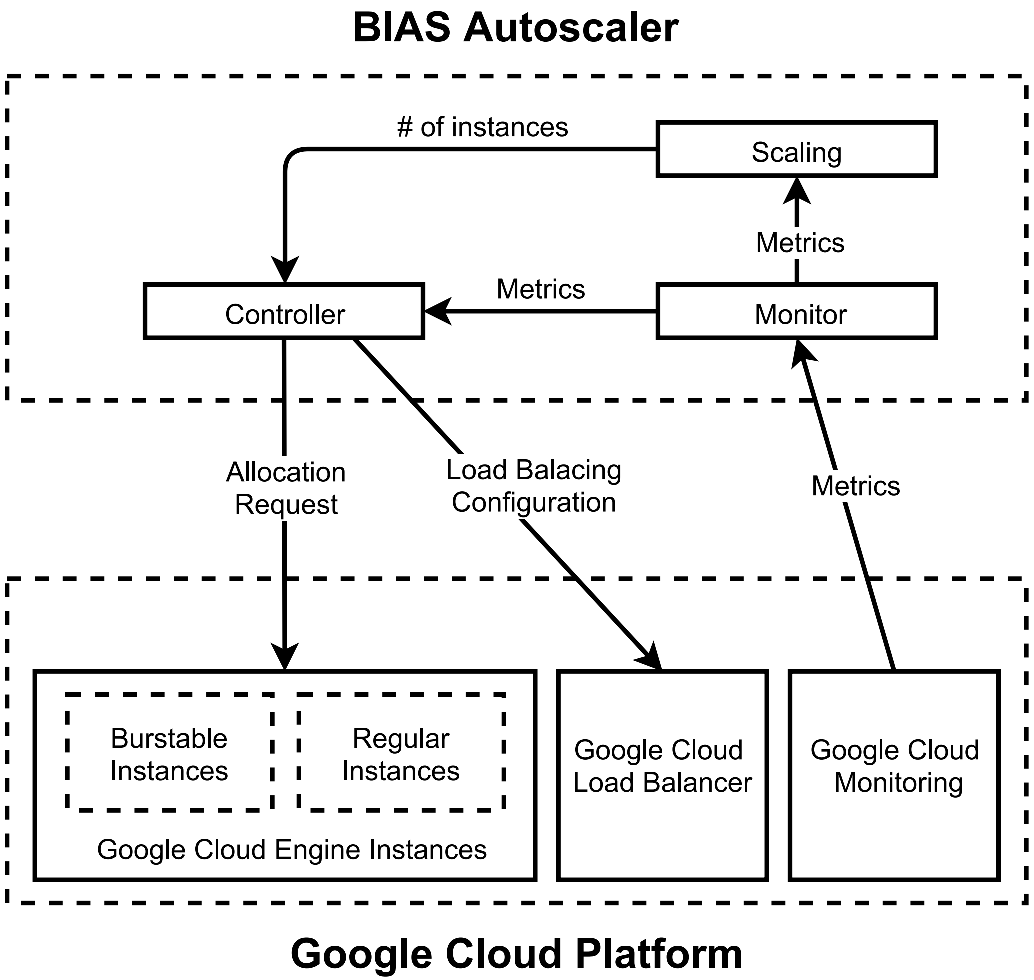
MICRONAUT

Cloud SDK

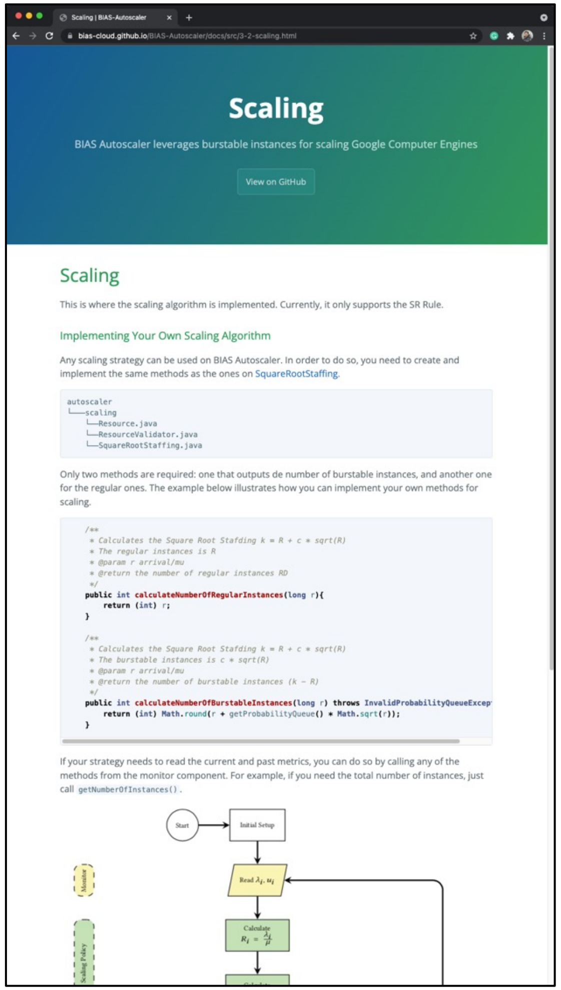


System Design

- The **scaling** interface provides an easy way to implement any scaling algorithm
- The **monitoring** module can be integrated with other monitoring agents (e.g. Prometheus)



Complete Documentation


Scaling | BIAS-Autoscaler
bias-cloud.github.io/BIAS-Autoscaler/docs/src/3-2-scaling.html
View on GitHub
Start
Initial Setup
Monitor
Read λ_i, u_i
Scaling Policy
Calculate $R_i = \frac{\lambda_i}{\mu}$

Scaling

BIAS Autoscaler leverages burstable instances for scaling Google Computer Engines

Scaling

This is where the scaling algorithm is implemented. Currently, it only supports the SR Rule.

Implementing Your Own Scaling Algorithm

Any scaling strategy can be used on BIAS Autoscaler. In order to do so, you need to create and implement the same methods as the ones on SquareRootStaffing.

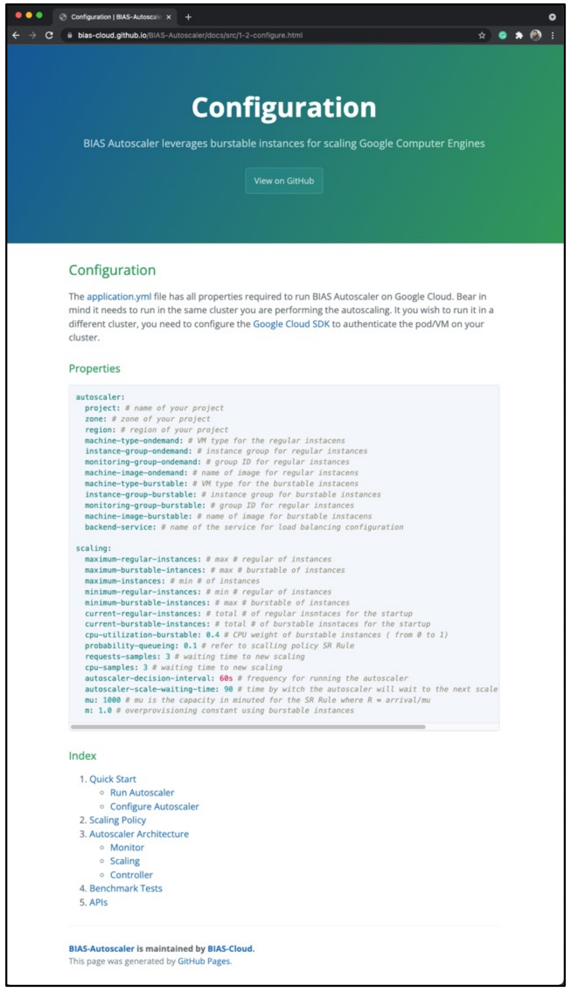
```
autoscaler
└──scaling
    └──Resource.java
    └──ResourceValidator.java
    └──SquareRootStaffing.java
```

Only two methods are required: one that outputs de number of burstable instances, and another one for the regular ones. The example below illustrates how you can implement your own methods for scaling.

```
    /**
     * Calculates the Square Root Stafding k = R + c * sqrt(R)
     * The regular instances is R
     * @param r arrival/mu
     * @return the number of regular instances RD
     */
    public int calculateNumberOfRegularInstances(long r){
        return (int) r;
    }

    /**
     * Calculates the Square Root Stafding k = R + c * sqrt(R)
     * The burstable instances is c * sqrt(R)
     * @param r arrival/mu
     * @return the number of burstable instances (k - R)
     */
    public int calculateNumberOfBurstableInstances(long r) throws InvalidProbabilityQueueExcep
        return (int) Math.round(r + getProbabilityQueue() * Math.sqrt(r));
    }
```

If your strategy needs to read the current and past metrics, you can do so by calling any of the methods from the monitor component. For example, if you need the total number of instances, just call `getNumberOfInstances()`.


Configuration | BIAS-Autoscaler
bias-cloud.github.io/BIAS-Autoscaler/docs/src/1-2-configure.html
View on GitHub

Configuration

BIAS Autoscaler leverages burstable instances for scaling Google Computer Engines

Configuration

The application.yml file has all properties required to run BIAS Autoscaler on Google Cloud. Bear in mind it needs to run in the same cluster you are performing the autoscaling. It you wish to run it in a different cluster, you need to configure the Google Cloud SDK to authenticate the pod/VM on your cluster.

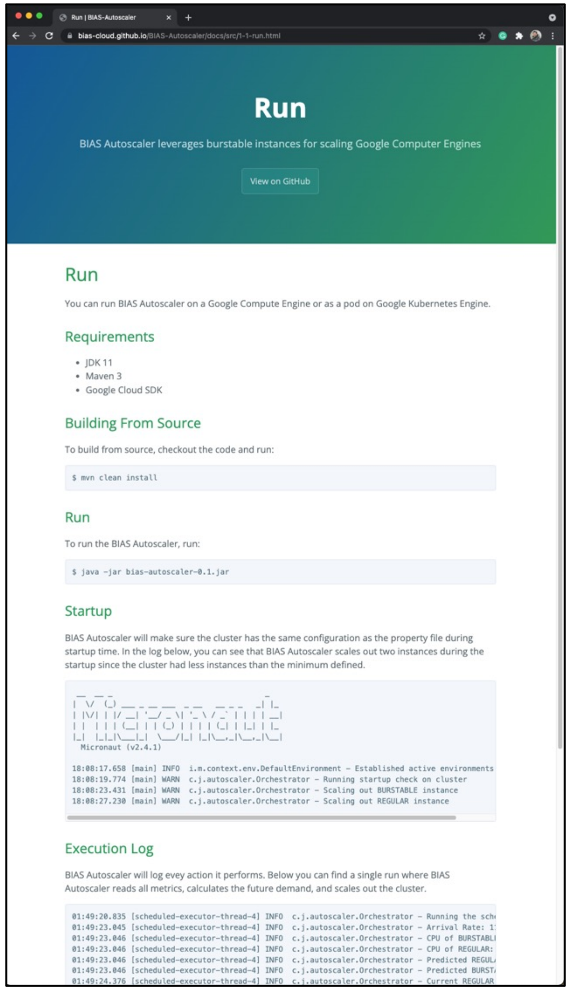
Properties

```
autoscaler:
  project: # name of your project
  zone: # zone of your project
  region: # region of your project
  machine-type-ondemand: # VM type for the regular instacens
  instance-group-ondemand: # instance group for regular instances
  monitoring-group-ondemand: # group ID for regular instances
  machine-image-ondemand: # name of image for regular instacens
  machine-type-burstable: # VM type for the burstable instacens
  instance-group-burstable: # instance group for burstable instances
  monitoring-group-burstable: # group ID for regular instances
  machine-image-burstable: # name of image for burstable instacens
  backend-service: # name of the service for load balancing configuration

scaling:
  maximum-regular-instances: # max # regular of instances
  maximum-burstable-intances: # max # burstable of instances
  maximum-instances: # min # of instances
  minimum-regular-instances: # min # regular of instances
  minimum-burstable-instances: # max # burstable of instances
  current-regular-instances: # total # of regular insntaces for the startup
  current-burstable-instances: # total # of burstable insntaces for the startup
  cpu-utilization-burstable: 0.4 # CPU weight of burstable instances ( from 0 to 1)
  probability-queueing: 0.1 # refer to scalling policy SR Rule
  requests-samples: 3 # waiting time to new scaling
  cpu-samples: 3 # waiting time to new scaling
  autoscaler-decision-interval: 60s # frequency for running the autoscaler
  autoscaler-scale-waiting-time: 90 # time by witch the autoscaler will wait to the next scale
  mu: 1000 # mu is the capacity in minuted for the SR Rule where R = arrival/mu
  m: 1.0 # overprovisioning constant using burstable instances
```

Index

BIAS-Autoscaler is maintained by BIAS-Cloud.
This page was generated by GitHub Pages.



Run

BIAS Autoscaler leverages burstable instances for scaling Google Computer Engines

Run

You can run BIAS Autoscaler on a Google Compute Engine or as a pod on Google Kubernetes Engine.

Requirements

- JDK 11
- Maven 3
- Google Cloud SDK

Building From Source

To build from source, checkout the code and run:

```
$ mvn clean install
```

Run

To run the BIAS Autoscaler, run:

```
$ java -jar bias-autoscaler-0.1.jar
```

Startup

BIAS Autoscaler will make sure the cluster has the same configuration as the property file during startup time. In the log below, you can see that BIAS Autoscaler scales out two instances during the startup since the cluster had less instances than the minimum defined.

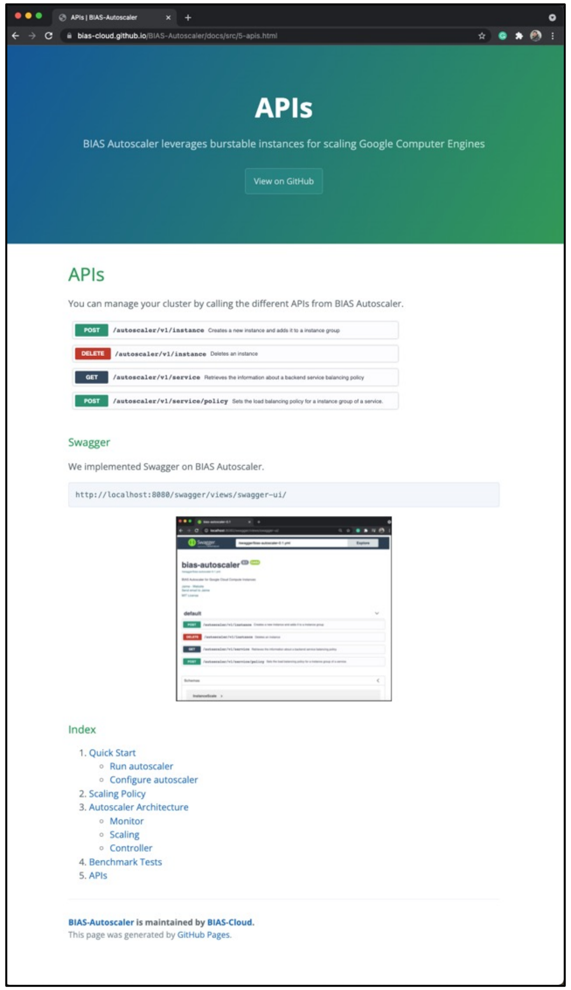
```
  Micronaut (v2.4.1)

18:08:17.658 [main] INFO  i.m.context.env.DefaultEnvironment - Established active environments
18:08:19.774 [main] WARN  c.j.autoscaler.Orchestrator - Running startup check on cluster
18:08:23.431 [main] WARN  c.j.autoscaler.Orchestrator - Scaling out BURSTABLE instance
18:08:27.230 [main] WARN  c.j.autoscaler.Orchestrator - Scaling out REGULAR instance
```

Execution Log

BIAS Autoscaler will log evey action it performs. Below you can find a single run where BIAS Autoscaler reads all metrics, calculates the future demand, and scales out the cluster.

```
01:49:20.835 [scheduled-executor-thread-4] INFO  c.j.autoscaler.Orchestrator - Running the sch
01:49:23.045 [scheduled-executor-thread-4] INFO  c.j.autoscaler.Orchestrator - Arrival Rate: 1
01:49:23.046 [scheduled-executor-thread-4] INFO  c.j.autoscaler.Orchestrator - CPU of BURSTABL
01:49:23.046 [scheduled-executor-thread-4] INFO  c.j.autoscaler.Orchestrator - CPU of REGULAR:
01:49:23.046 [scheduled-executor-thread-4] INFO  c.j.autoscaler.Orchestrator - Predicted REGUL
01:49:23.046 [scheduled-executor-thread-4] INFO  c.j.autoscaler.Orchestrator - Predicted BURST
01:49:24.376 [scheduled-executor-thread-4] INFO  c.j.autoscaler.Orchestrator - Current REGULAR
```



APIs

BIAS Autoscaler leverages burstable instances for scaling Google Computer Engines

APIs

You can manage your cluster by calling the different APIs from BIAS Autoscaler.

POST	`/autoscaler/v1/instance`	Creates a new instance and adds it to a instance group
DELETE	`/autoscaler/v1/instance`	Deletes an instance
GET	`/autoscaler/v1/service`	Retrieves the information about a backend service balancing policy
POST	`/autoscaler/v1/service/policy`	Sets the load balancing policy for a instance group of a service.

Swagger

We implemented Swagger on BIAS Autoscaler.

```
http://localhost:8080/swagger/views/swagger-ui/
```

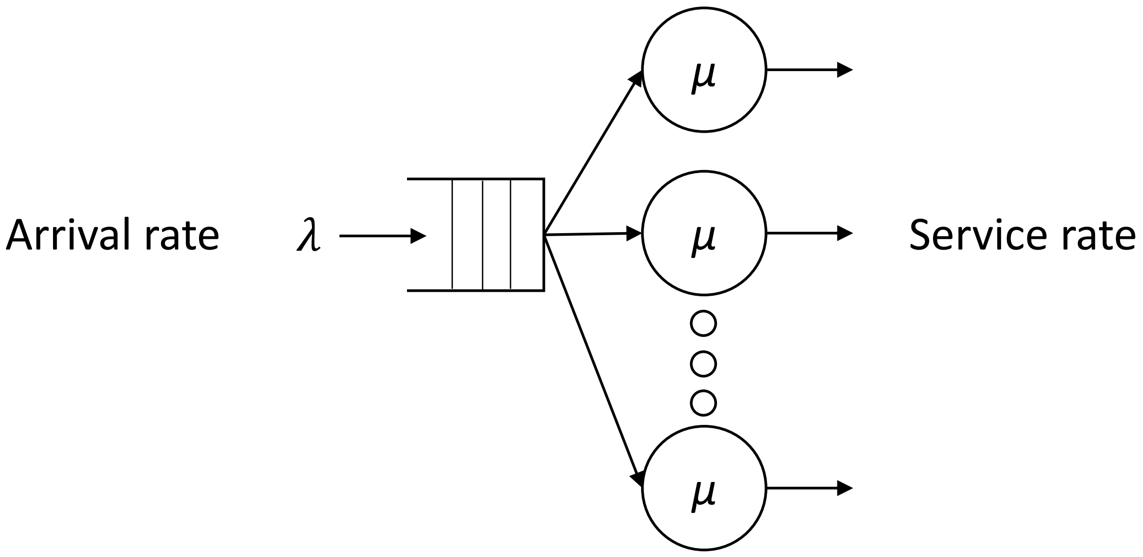
Index

BIAS-Autoscaler is maintained by BIAS-Cloud.
This page was generated by GitHub Pages.

https://bias-cloud.github.io/BIAS-Autoscaler

Scaling Policy

- Reactive Autoscaler
- M/M/k queueing system
- Square-Root Staffing Rule (SR Rule)



Load

$$\rho = \frac{\lambda}{k\mu}$$

Resource utilization

$$R = \frac{\lambda}{\mu}$$

# of servers

$$k_{\propto} = R + \mathrm{c}\sqrt{R}$$

Waiting time

$$E[T] = \frac{1}{\lambda} \times P_Q \times \frac{\rho}{1-\rho} + \frac{1}{\mu}$$

Upper bound on the probability of queueing $\boldsymbol{\alpha}$

10%

Experimental Evaluation

- Set the average and the 95th percentile for SLOs
- Consumes a RESTFul API
- Scaling interval of 1min

Load Generator:

LOCUST

https://locust.io

Web API:

Load
Microservice

https://bias-cloud.github.io/Load-Microservice

Regular

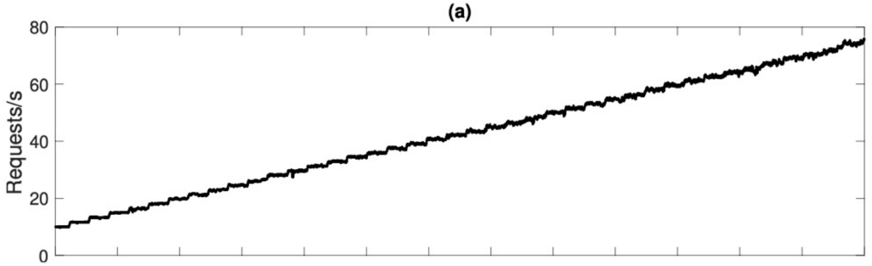
N1 standard 1
3.75 GB RAM

Burstable @ 50%

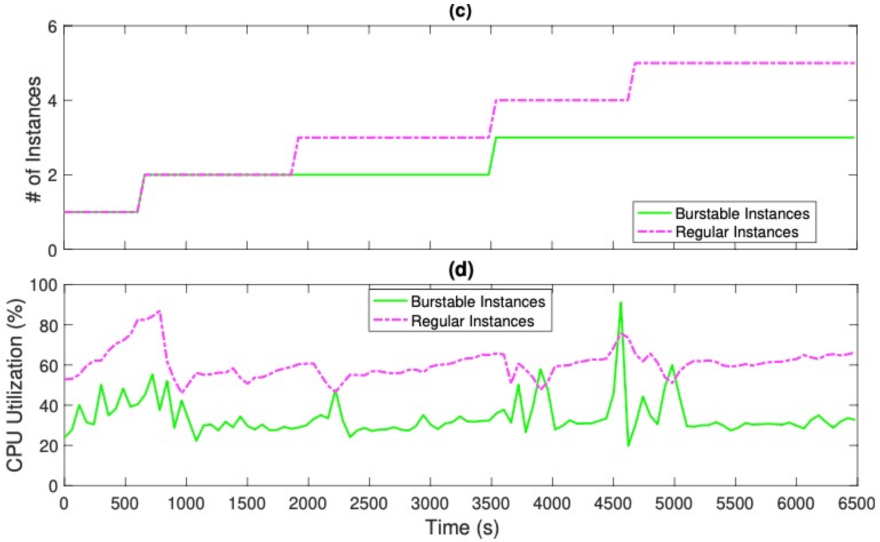
N1 shared-core g1-small
1.7 GB RAM
(52% cheaper)

Transient Queueing

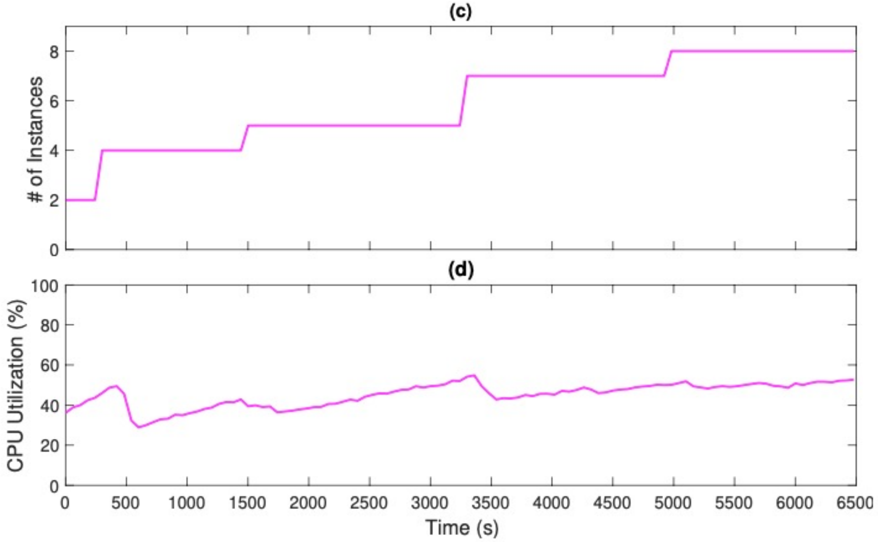
Load



Burstable and Regular Instances

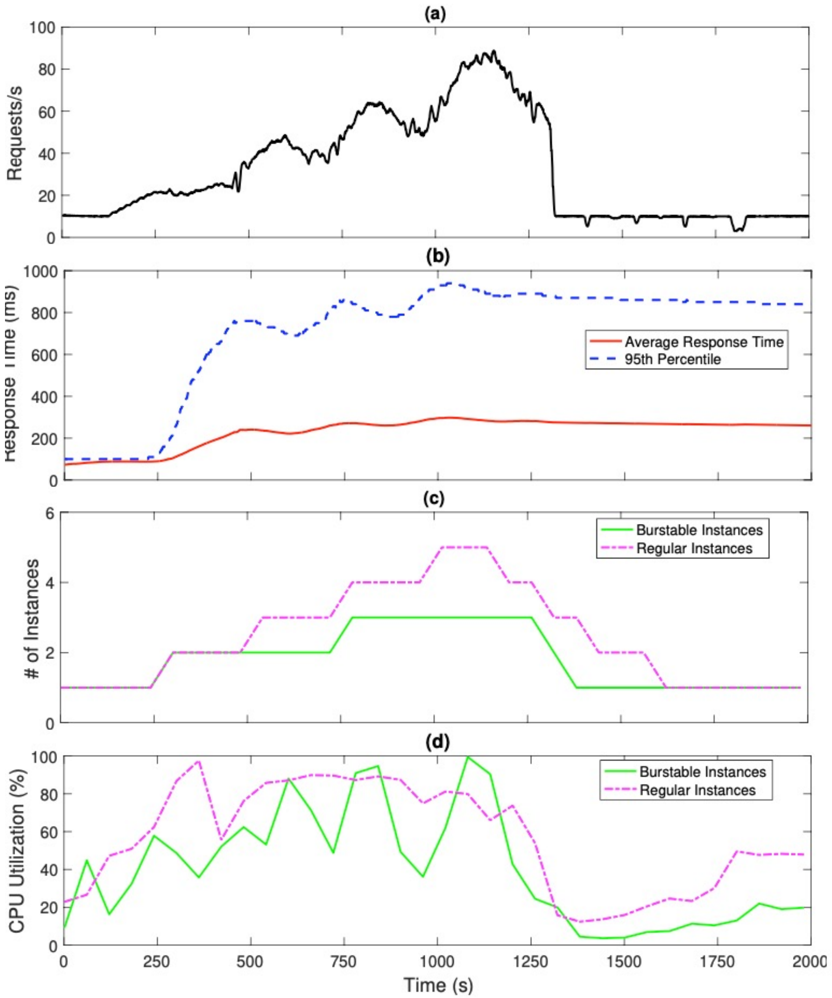


Regular Instances Only



Test Scenario	Average Response Time (ms)	Maximum 95th Percentile (ms)	Cost (10^{-3} USD)
Regular instances only	110	210	493
Rule-based GCP autoscaler	108	220	450
Burstable and regular instances	118	280	371
Burstable instances only	120	220	218

Flash Crowds



Burstable and Regular Instances

- Savings of up to 25%
- Increased resource efficiency by 42%

Conclusion

- Created BIAS Autoscaler, an autoscaler that leverages burstable instances on the public cloud.
- Validated our application on Google Cloud with Compute Engines.
- Ran BIAS Autoscaler under a transient queueing and a flash crowd experiment.
- Achieved promising results of 25% in savings and 42% increase in resource efficiency without interfering with the quality of the service when using burstable instances.
- BIAS Autoscaler can be modified to be used with container scaling or other cloud services providers.

Thank you!