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
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

- Amazon EC2:
  - 34% burstable
  - 10% regular

- Microsoft Azure:
  - 43% burstable
  - 45% regular

- Google Cloud:
  - 50% burstable
  - 52% regular
Burstable Instances

- CPU: 60%
- 20% (1 Token = 1 min)
- 90% (Token cost)

Logos: 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:

![Stack Diagram]

- Java
- Micronaut
- Cloud SDK
- Google Cloud

![Diagram]

API

Cloud SDK

Cloud Load Balancer

Regular

Burstable

Internet

- Scales OUT
- IN
- Instances
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

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

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

Resource utilization

\[ R = \frac{\lambda}{\mu} \]

Waiting time

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

Load

\[ \rho = \frac{\lambda}{k\mu} \]

Upper bound on the probability of queueing \( \alpha \)

\[ k_\alpha = R + c\sqrt{R} \]
Experimental Evaluation

- Set the average and the 95\textsuperscript{th} 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

### Burstable and Regular Instances

**Load**

(a) Requests/s

(b) # of Instances

(c) CPU Utilization (%)

---

### Regular Instances Only

(d) Requests/s

(e) # of Instances

(f) CPU Utilization (%)

---

<table>
<thead>
<tr>
<th>Test Scenario</th>
<th>Average Response Time (ms)</th>
<th>Maximum 95th Percentile (ms)</th>
<th>Cost ($10^{-3}$ USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular instances only</td>
<td>110</td>
<td>210</td>
<td>493</td>
</tr>
<tr>
<td>Rule-based GCP autoscaler</td>
<td>108</td>
<td>220</td>
<td>450</td>
</tr>
<tr>
<td>Burstable and regular instances</td>
<td>118</td>
<td>280</td>
<td>371</td>
</tr>
<tr>
<td>Burstable instances only</td>
<td>120</td>
<td>220</td>
<td>218</td>
</tr>
</tbody>
</table>
Flash Crowds

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

Burstable and Regular Instances
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!