Introduction

What is Kubernetes?

- Automation of deployment and management
- Created by Google in 2015
- Software companies use own version now
- Scaling and load balancing increase performance of application
Outline

- Background Concepts
  - Kubernetes Hierarchy
  - Load-balance vs Scaling
  - Central Processing Unit (CPU)
- Methods
  - Load-balancing
  - Scaling
- Conclusion
Client Server Architecture

- User
- Internet
- Nodes
Containers:

- Run faster and more consistent between environments

Application runs inside of the containers
Containers run inside of pods
1 or more container per pod
Pods: smallest unit of execution
Two types of nodes in Kubernetes:
- Worker Node
- Master node
A cluster is made up of:
- containers, pods, and nodes
- Unique cluster IP
API server

Application Programming Interface (API)

Communicates with Kubelet and Kube-Proxy

Changing the number of nodes
Kubernetes provides Horizontal scaling

- Changing number of pods or nodes
- Distribute workload

Ex. CPU usage of a pod hits 80 percent of capacity
Load-Balancing Background

Distributes traffic to replicated applications

Bottleneck?
Central Processing Unit (CPU)

CPUs are measured in cores

- Measures the compute processing in a machine
- Most cases CPU usage in Kubernetes is measured in millicores
- Millicore is one thousandth of a core
- Ex. 2000 millicores (or 2000m) is equal to 2 cores
- This is standard for every unit in Kubernetes
Server Scalability Using Kubernetes


Study uses 3 services
- Get - get data
- Post - send data
- Delete - delete data

Evaluates CPU usage single vs. multiple server (Using scalability)

3 Kubernetes nodes with multiple servers
- 1 master node
- 2 worker nodes
Kubernetes Horizontal Pod Autoscaling

What is KHPA?

- Increase or decrease based on the number of concurrent users.
- KHPA takes the target input as a percentage of CPU usage
- The output is the pods target number
CPU Usage

Simulated behavior using a combination of services
1,650 requests in each task

<table>
<thead>
<tr>
<th>Task</th>
<th>CPU Usage Pod (in millicores) on Single Server</th>
<th>CPU Usage Pod (in millicores) on Multiple Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>576.00</td>
<td>209.00</td>
</tr>
<tr>
<td>2</td>
<td>526.00</td>
<td>369.00</td>
</tr>
<tr>
<td>3</td>
<td>498.00</td>
<td>333.00</td>
</tr>
<tr>
<td>Average</td>
<td>533.33</td>
<td>303.66</td>
</tr>
</tbody>
</table>
**CPU Usage**

Users send 8000 requests of each service

Evaluates CPU usage (millicores) of single vs. multiple servers

<table>
<thead>
<tr>
<th>Average</th>
<th>GET-1</th>
<th>625.33</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GET-2</td>
<td>595.33</td>
<td>233.66</td>
</tr>
<tr>
<td></td>
<td>POST</td>
<td>567.66</td>
<td>264.00</td>
</tr>
</tbody>
</table>

Averages of previous scenario:

| Average | 533.33 | 303.66 |
Scaling Results

Single Server
- In both scenarios the CPU usage was much higher
- When the number of requests went up the CPU usage increased

Multiple Servers (Scaling)
- Servers distribute workload
- CPU usage is smaller when there are more requests
Raft Algorithm

Leader selection algorithm for Kubernetes

Used to select pod leaders in a node

- Majority of traffic goes to leaders
- Candidates try to become the leader
- Followers are replica applications that don’t get the majority of traffic
Toward Highly Scalable Load-Balancing in Kubernetes Cluster


Concentrated vs. Balanced Leaders

- 5 leaders distributed to 3 nodes
- Balanced leaders uses the leader election
- Majority of traffic is sent to the leaders
Leaders

Concentrated vs. Balanced Leaders

- Concentrated: 5 leaders in node 1
- Balanced: 5 leaders distributed throughout 3 nodes
CPU Utilization

4 clients send requests for 150s
The requests are handled by the leader pods
Throughput

Clients increasing from 1 to 32
Clients send 2000 requests each
Throughput- Number of request per second the cluster can recieve
Load-Balancing Results

Concentrated leaders
- CPU utilization 60 percent higher in node 1 versus the other two nodes
- Once 4 clients send requests throughput is constant

Balanced leaders
- More balanced CPU utilization
- More requests per second
Conclusion

- Scaling and load-balancing increase performance
- Scaling focused on distributing to different nodes
- Load-balancing focused on distributing network traffic
I would like to thank Nic McPhee and Elena Machkasova for great feedback and guidance throughout this process. Also thanks to the CSCI department for great planning of this event.
Questions?