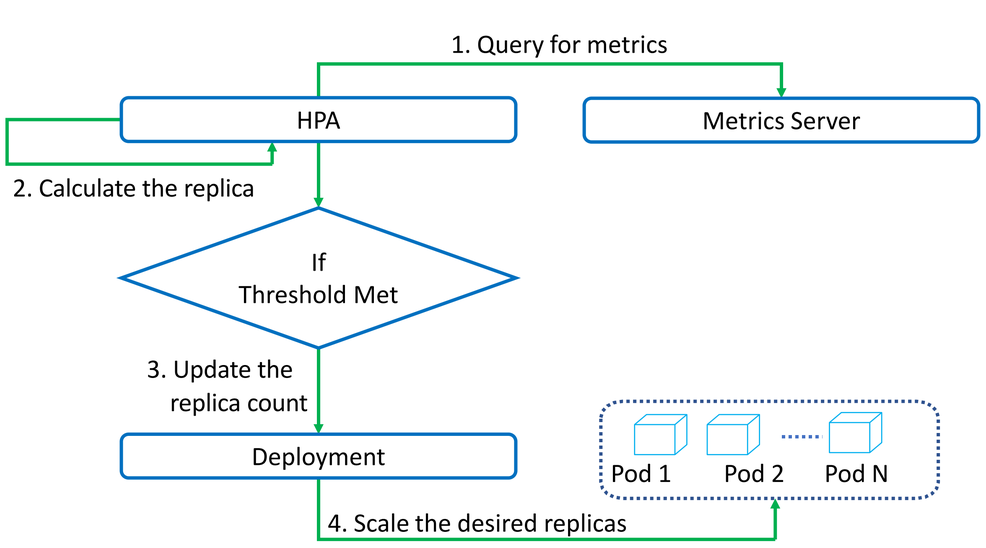
Towards An Efficient Autoscaling Decision for Cloud Applications

Nancy Chahal, Kenneth B. Kent
Faculty of Computer Science, University of New Brunswick
Joran Siu — IBM Canada
Michael Dawson — Red Hat Canada
{nancy.chahal, ken}@unb.ca, joransiu@ca.ibm.com, midawson@redhat.com

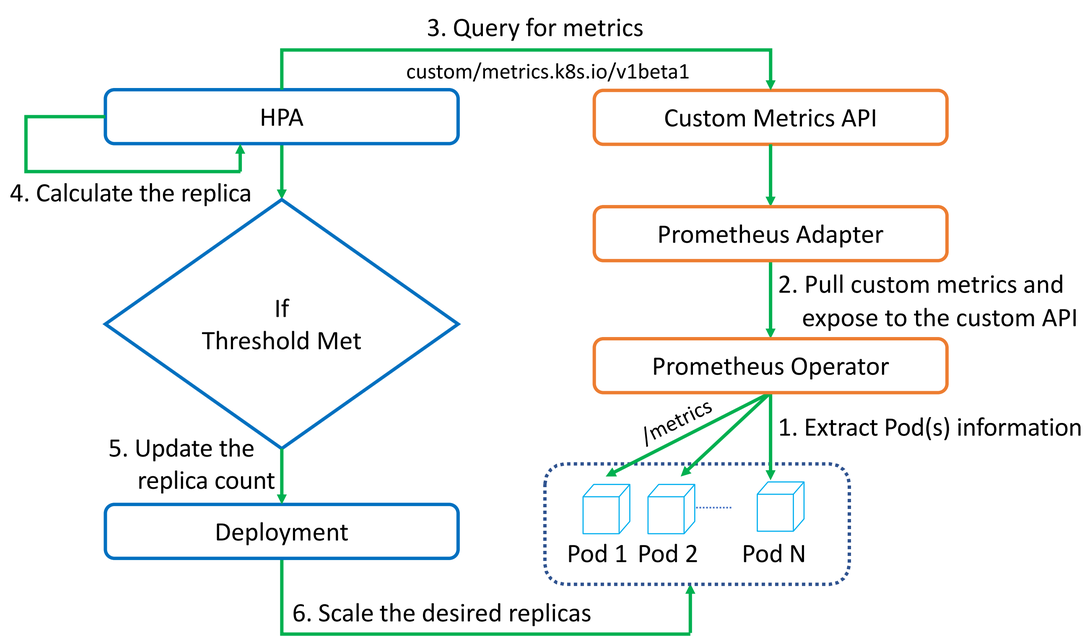
Motivation

- In many cases, a single metric cannot efficiently predict the resource requirements of an application, leading to higher computation costs, suboptimal scaling of pods and reduced performance.
- Analyzing more than one metric can capture a more detailed view of the current state of the application.
- Scaling in and out might not always be the best way towards performance increase of the application.
- Exploring the suitability and impact of various metrics on the throughput during autoscaling.

Autoscaling in Kubernetes



Kubernetes HPA using Custom Metrics



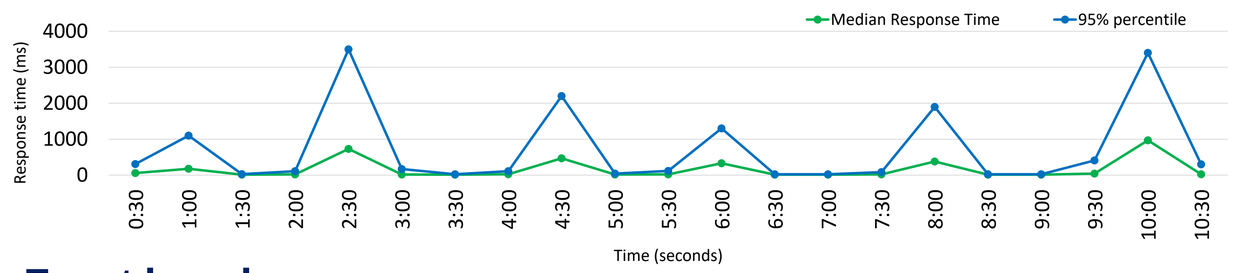
Methodology

- Multi-node Kubernetes cluster; three servers (one master, two worker nodes)
- CPU intensive and I/O intensive applications
- Linearly increasing load

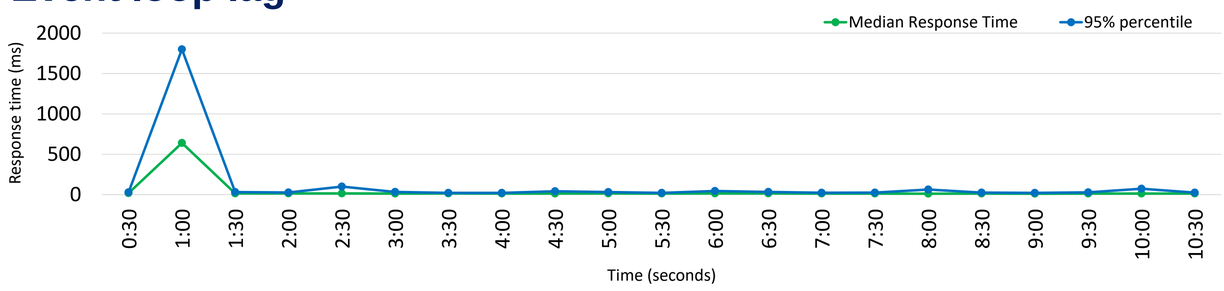
Analyzed Metrics	Description
CPU Utilization	CPU capacity required to handle the cumulative workload on the service
Event loop lag	Describes the delay or lag of event loop in seconds (language runtime-specific metric)
HTTP Requests	Average number of HTTP requests per second
Garbage Collection Count	Refers to garbage collection by kind {major, minor or incremental}

Response Time while Autoscaling based on:

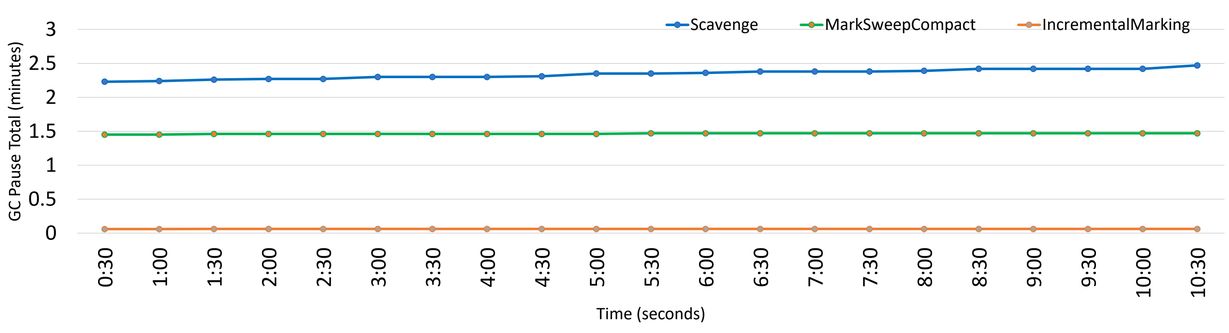
CPU Utilization



Event loop lag



Garbage Collector Pause Metric



Analysis & On-going Work

- Event loop lag-based autoscaling provide better response time and performance than CPU utilization.
- Garbage collection metrics gave some interesting insights towards CPU usage.
- Our goal is to find an alternative towards scaling in and out as part of autoscaling for the underlying load.

CAS–Atlantic