Proactive Horizontal Auto-Scaling for Kubernetes

Rajesh Manedi, Rohit Mohan, Vinay Vasudev

Introduction

- Elasticity is the key property for cloud computing to become popular.
- Kubernetes is a well known container orchestrator for cloud deployed applications.
- Kubernetes offers Horizontal and Vertical Pod Autoscaling

Motivation

- Default approach used in kubernetes is reactive HPA
- Scaling is done only after change is observed
- Issue with this approach is over-provisioning and under-provisioning of resources
- Solution is to move from reactive to proactive auto scaling.

Related Work

- RPPS: A Novel Resource Prediction and Provisioning Scheme in Cloud Data Center
- Machine learning-based auto-scaling for containerized applications
- Fisher: An Efficient Container Load Prediction Model with Deep Neural Network in Clouds

Proposed Approach

Metrics:

- System Level Metrics : Average CPU and Memory Utilization
- Application Level Metric : Number of HTTP Requests.

Granularity considered for the metric collection and decision making is 15 seconds.

Proposed Approach

MultiVariate Time-Series Prediction Models:

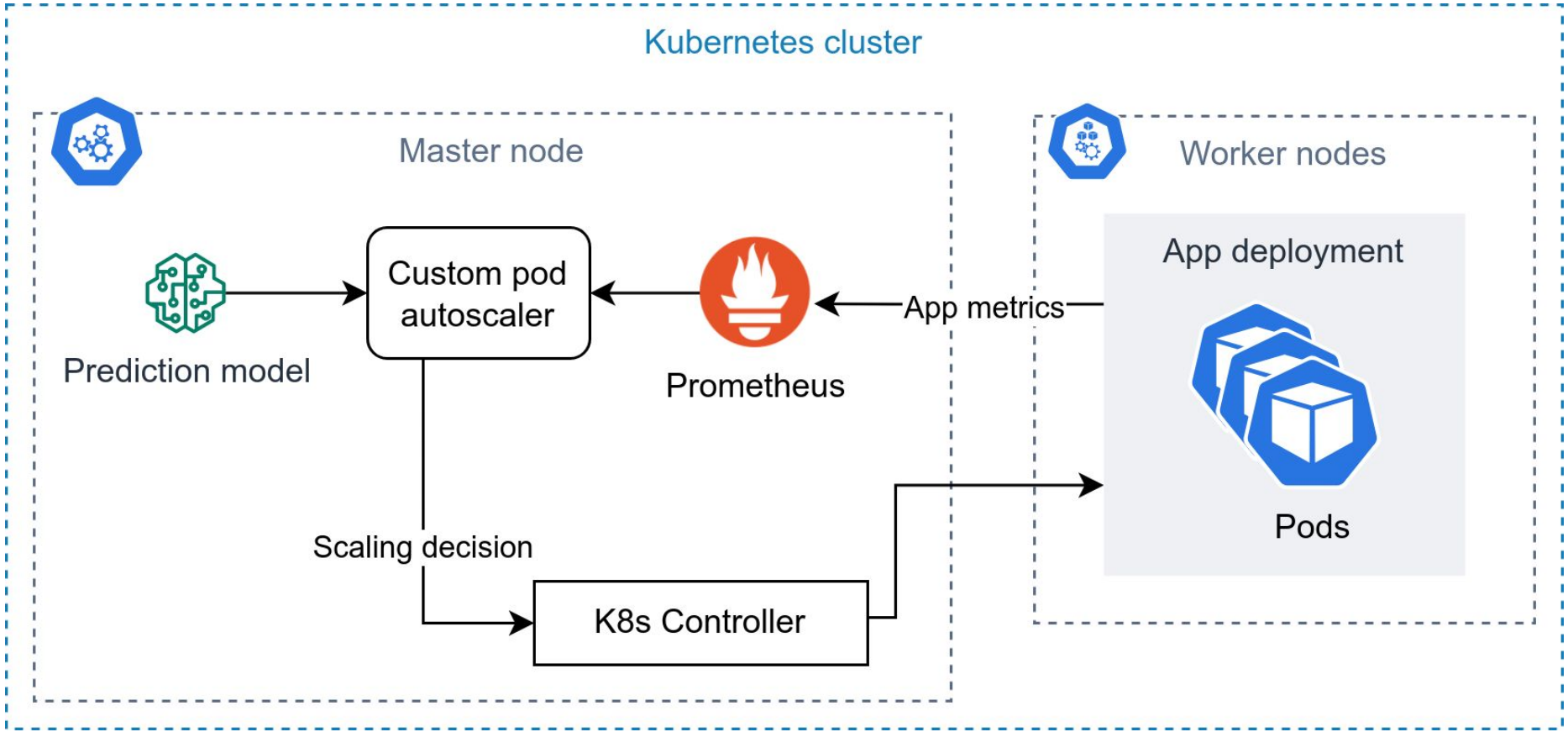
- Autoregressive Integrated Moving Average (ARIMA)
- Bidirectional Long-Short Term Memory (Bi-LSTM)
- Gated Recurrent Unit (GRU)

Formula to be used :

$DesiredReplicas_{mx} = ceil[CurrentReplicas * (PredictedMetricValue_{mx}/DesiredMetricValue_{mx})$

mx = {Avg CPU Utilization, Avg Memory Utilization, Number of HTTP Requests}

$ConfiguredReplicas = max(DesiredReplicas_{mx})$


Kubernetes cluster
Master node
Worker nodes
Custom pod autoscaler
Prediction model
Prometheus
App metrics
App deployment
Pods
Scaling decision
K8s Controller

Experiment Setup

- Generate load using Locust with Wikipedia access trace patterns
- We will use a stateless HTTP server application as the container
- Container deployed as a service on kubernetes

Evaluation

We will compare the results from the following experiments:

1. Behavior and metrics from the container with no auto scaling
2. Baseline HPA with CPU and memory utilization, request rate metrics with appropriate thresholds
3. Custom auto-scaler using the different prediction models (ARIMA, GRU, Bi-LSTM)

Evaluation metrics

- Root Mean Square Error
- % deviation from the expected replicas

Plan of Work

- Environment setup and data collection - Rohit
- Model training and evaluation - Rajesh
- Custom autoscaler development - Vinay
- Experiments - Team
- Evaluation - Team

References

- Fang, Wei and Lu, ZhiHui and Wu, Jie and Cao, ZhenYin
  https://ieeexplore.ieee.org/document/6274197
- Imdoukh, Mahmoud and Ahmad, Imtiaz and Alfailakawi, Mohammad Gh Alfailakawi
  Machine learning-based auto-scaling for containerized applications
- Tang, Xuehai and Liu, Qiuyang and Dong, Yangchen and Han, Jizhong and Zhang, Zhiyuan
  Fisher: An Efficient Container Load Prediction Model with Deep Neural Network in Clouds