Sustainability in Computing

Energy Efficient Placements of Kubernetes Workloads

Kaiyi Liu
Software Engineer Intern

Parul Singh
Senior Software Engineer
Who We Are

● Community based initiatives on environmental sustainability

● Proposal: **CNCF TAG Environmental Sustainability**

● **Carbon Aware Scaling with KEDA**
  ○ a community based initiative; investigates how to use electricity carbon intensity to make workload scaling decisions.

● **CLEVER**:
  ○ Container Level Energy-efficient VPA Recommender for Kubernetes
Agenda

● Background
● Introduce our Sustainability stack
  ○ Kepler
  ○ Model Server
● Demo
Background

According to Gartner, “In 2021, an ACM technology brief estimated that the information and communication technology (ICT) sector contributed between 1.8% and 3.9% of global carbon emissions.”
Background

- How to measure energy consumption indirectly?
- How to measure energy consumption of workloads?
- How to attribute power on share resources to processes, containers or Pods?
Introducing the Cloud Native Sustainability Stack

1. Kepler
2. Kepler Model Server
Energy Consumption Attribution Methodology

Kepler

Kubernetes based Efficient Power Level Exporter
Kepler: Kubernetes based Efficient Power Level Exporter

Uses software counters to measure power consumption by hardware resources and exports as Prometheus metrics
- Per Pod level energy consumption reporting, including CPU/GPU, RAM
- Support **bare metal** as well as VM
- Support **Prometheus**
- Per Pod level energy consumption reporting, including **CPU/GPU, RAM**
- Support **bare metal** as well as **VM**
- Support **Prometheus**

- Reduced computational resource used by the probe
- Using **eBPF**
- Per Pod level energy consumption reporting, including CPU/GPU, RAM
- Support bare metal as well as VM
- Support Prometheus

- Reduced computational resource used by the probe
- Using eBPF

- Support ML models to estimate energy consumption
- Science based approach
About Kepler Model Server

- Default: Kepler uses supported power meter tools to measure node level energy metrics (CPU core, DRAM, Pod Energy)
- Problem: No supported power meter for Kepler
- Model Server Goal: Provide Trained Models for Kepler that use Software Counters/Performance metrics to predict missing energy metrics
- Current Tech Stack: Tensorflow Keras, Scikit, Flask, Prometheus
Kepler Model Server’s Models

- CPU Core Energy Consumption Model: Linear Regression
  - Label: CPU Core Energy Consumption
  - Features: cpu_architecture, curr_cpu_cycles, curr_cpu_instructions, curr_cpu_time
- Online Learning
Kepler Model Server’s Models

Cont

- Dram Energy Consumption Model: Linear Regression
  - Label: DRAM Energy Consumption
  - Features: `cpu_architecture`, `curr_cache_misses`, `memory_working_set`
- Online Learning
Model Server and Kepler

Training Phase

Kepler Collector

- Export Node Metrics

Kepler Model Server
- Scrape Node metrics from Kepler Prometheus
- Set up training, testing, validation datasets
- Train, Evaluate, and save Regression models if error is not too high
  (Tensorflow’s fit and evaluate methods)
- Export Models

Node

Kepler Power Estimate Consumption Agent

Pod
Pod

Node

Kepler Power Estimate Consumption Agent

Pod
Pod
# Model Server and Kepler

## Exporting Phase

<table>
<thead>
<tr>
<th>Kepler Model Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Scrape Node metrics from Kepler Prometheus</td>
</tr>
<tr>
<td>- Set up training, testing, validation datasets</td>
</tr>
<tr>
<td>- Train, Evaluate, and save Regression models if error is not too high (Tensorflow's fit and evaluate methods)</td>
</tr>
<tr>
<td>- Export Models</td>
</tr>
</tbody>
</table>

![Diagram](image)
Carbon Intensity Aware Scheduling
Use Cases

Solar Power vs Fossil Fuel

Control Carbon Intensity
Use Case Premise

- Multi-node cluster
- Nodes in different zones
- Long running batch/ML workloads
Demo Set Up

- OpenShift Cluster
- Monitoring: Prometheus
- Taints/Tolerations/NodeSelectors
- Carbon Intensity Forecaster
Carbon Intensity Forecaster

- Exporter scrapes from Public Energy APIs (ex. Electricity Map) and exports as Prometheus metrics
- Scrapes Prometheus metrics from the exporter to update models for each node
- Carbon Intensity Forecaster and Exporter are extendable interfaces
Periodically assign node labels according to forecasted carbon intensity of the zone the node is present.

```
kubectl label nodes ip-10-0-169-34.ec2.internal carbon_intensity=green
```
tolerationSeconds means that if this pod is running and a matching taint is added to the node, then the pod will stay bound to the node for 5 seconds, and then be evicted.
Tainting nodes ensure pods are evicted by the nodes if pods have no tolerations for the taint.
Demo
Overview

Release
v1alpha1

Features:

● Pre-requisite Cgroup v2
● Follows Kepler v0.4
● Deploy’s Kepler on Kubernetes and OpenShift
  ○ Pre-configuration for OpenShift (MachineConfig and SCC)
● Uses offline model
  ○ Uses local linear regression estimator in Kepler main container with offline trained model weights.

https://github.com/sustainable-computing-io/kepler-operator
Demo - Lessons Learnt

- Finding Zone Carbon Intensity Data
  - Some time points are missing
Demo - Lessons Learnt

- Finding Zone Carbon Intensity Data
- Need to support multiple query types
  - It is easy to query threshold friendly metric on Prometheus (e.g. what is the current or average carbon intensity in zone XYZ?), but hard on others (no threshold or more complicated logic)
    - Which zone has the lowest carbon intensity?
    - Is the current carbon intensity low, e.g. within the past 24 hours?
Demo - Lessons Learnt

- Finding Zone Carbon Intensity Data
- Need to support multiple query types
  - It is easy to query threshold friendly metric on Prometheus (e.g. what is the current or average carbon intensity in zone XYZ?), but hard on others (no threshold or more complicated logic)
    - Which zone has the lowest carbon intensity?
    - Is the current carbon intensity low, e.g. within the past 24 hours?
- Need to support multiple electricity carbon emission providers
  - Improve and integrate with Green Software Foundation carbon-aware SDK
Road Ahead

- Apply to multi-cluster
  - Explore approach with kcp

- Integrate carbon-intensity awareness in kubernetes-sigs/scheduler-plugins
  - Use Trimaran TargetLoadPacking profile and integrate carbon-intensity awareness in the scheduler
  - Tune Trimaran for energy efficiency.
References

- How to use performance counters to estimate power consumption by cpu, memory, etc

- Kepler:
  https://github.com/sustainable-computing-io/kepler

- The Model Server:
Thank You