HyPPO: HYBRID PERFORMANCE-AWARE POWER-CAPPING ORCHESTRATOR


Summarized By:
Vaidehi Barevadia
OUTLINE

• Introduction
• Related Work and Motivation
• Methodology
  • Observe
  • Decide
  • Act
• Evaluation
  • Experimental Setup
  • Experimental Results
• Limitations and Future Work
• Conclusion
INTRODUCTION

• Some critical services require the servers to remain up regardless the current traffic intensity in order to scale up quickly if needed.
• Reserved resources are needed for such services to guarantee Service Level Agreement(SLA) in case of sudden increase in requests.
• Average Utilization of the system is found to be around 75% in case of batch workloads, but drops to 10%-50% in case of mixed workloads.
• How to reduce Total Cost of Ownership(TCO) with all these problems?
INTRODUCTION

• As power consumption is accounted for the 20% of the TCO, Energy Proportionality is the key to reduce TCO.

• HyPPO, an orchestrator based on a distributed ODA control loop able to introduce energy proportionality in containerized infrastructure by exploiting the opportunity gap of cloud workloads.

• It leverages Kubernetes resource requests to dynamically adjust node power consumption, while respecting the SLA defined by containerized application owners.
RELATED WORK AND MOTIVATION

- Related work in the direction to achieve energy proportionality,
  - Minimize the overall power consumption of the data center.
  - Work only with batch workloads.
  - Conducted tests only in single node mode.
  - Proposed approach only for services composed by homogeneous tasks.
Fig. 2. HyPPO ODA loop: we observe the status of each server, centralizing power and performance metrics. Then, the decision phase computes a new strategy towards the system goals, which is enforced by the actuators.
METHODOLOGY - OBSERVE

Power and performance monitoring

Fig. 3. Monitoring back-end structure: The system receives samples of metrics and Kubernetes status from each monitoring agent in the GRPC collector. Kubernetes samples are unpacked and stored in a MongoDB database for later use. Monitoring samples are unpacked inside the Metrics workers and aggregated depending on containers and pods. The aggregated metrics are stored in an InfluxDB database, which is queried by the monitoring frontend to show real-time data. Finally, the REST endpoint serves requests of other ODA loop components.
METHODOLOGY - DECIDE

Performance-aware power allocation in 3 steps:

• Retrieving information by polling
• Exploiting received data to take a decision
• Communicating the outcome to actuator

• Controlling policy for each node -

\[
power_n = P_{idle} + \sum_{c=0}^{C} (power_{n,c} + i(c)) \quad (1)
\]

\[
i(c) = \begin{cases} 
(cpu_{usage,c} - cpu_{request,c}) \times P & \text{if } \exists cpu_{usage,c} \\
0 & \text{if } \not\exists cpu_{usage,c}
\end{cases} \quad (2)
\]
METHODOLOGY - ACT

Enforcing power allocation

• Leveraging RAPL to define power cap by using its package domain’s power limit and energy status interfaces, as they allow to set power cap and check if it is correctly enforced.

• They use 3 Model Specific Registers (MSR) –
  • Power unit MSR – to retrieve units of measure
  • Power limit MSR – to enforce power cap
  • Energy status MSR – to retrieve measured power
EVALUATION - EXPERIMENTAL SETUP

• Kubernetes version 1.10 was installed directly on the two nodes used in the experiment. Each workload ran inside a Docker container with a CPU request of 5 CPUs and a limit of 10 CPUs.

• The authors used 6 benchmarks from the phoronix test suite to evaluate the proposed orchestrator –
  • pts/apache-1.6.1
  • pts/ngnix-1.1.0
  • pts/dbench-1.0.0
  • pts./fio-1.4.0
  • pts/postmark-1.1.0
  • pts/iozone-1.8.0
EVALUATION - EXPERIMENTAL RESULTS

• For CPU intensive workloads, this technique leads to an increased overall execution time, which can lead to an increase in overall energy consumption.
EVALUATION - EXPERIMENTAL RESULTS

- Orchestrator discovers that pts/dbench requires more than requested CPU according to eq. (2) and introduces a positive contribute to the power cap on the node.
EVALUATION - EXPERIMENTAL RESULTS

- No exploitation of the opportunity gap, since these workload are strongly dependent on IO and disk performance.
- They are able to save power by capping core resources but at the cost of delaying the completion of workloads.
EVALUATION - EXPERIMENTAL RESULTS

• Now, to validate the behavior of the orchestrator in case of two containers each running an instance of pts/apache.

• Orchestrator is able to exploit an opportunity gap in both the containers resulting in power saving of the node.
EVALUATION - EXPERIMENTAL RESULTS

• To discover the impact of proposed hybrid orchestrator on the system under two different perspectives: power consumed and CPU utilized.

• All the components of the orchestrator present a negligible impact in terms of CPU utilized in the system.

<table>
<thead>
<tr>
<th>Component</th>
<th>CPU [%]</th>
<th>Power [mW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor</td>
<td>12.97</td>
<td>3028.69</td>
</tr>
<tr>
<td>Controller</td>
<td>0.61</td>
<td>295.29</td>
</tr>
<tr>
<td>Actuator</td>
<td>0.13</td>
<td>145.76</td>
</tr>
</tbody>
</table>
LIMITATIONS AND FUTURE WORK

- HyPPO strongly depends on Intel RAPL and does not consider workload parallelism.
- Evaluation of monitoring and decision components on more application specific performance metrics instead of using only power consumption.
- Make HyPPO capable of targeting On-Line Data Intensive (OLDI) workloads.
- They plan to shift to pub/sub communications strategy as the current approach introduces delays due to its polling communication strategy.
CONCLUSION

• This paper presented a novel hybrid performance-aware power-capping orchestrator enabling better energy proportionality in a distributed containerized environment governed by Kubernetes.

• The proposed approach is based on an ODA control loop strategy composed by three stages:
  • 1) the monitoring task performed by the DEEP-mon agents
  • 2) a controller in charge to reduce power consumption while respecting the define SLA
  • 3) the actuation performed by the actuator agents in charge to enforce the power cap on the single nodes exploiting RAPL interfaces.

• The results obtained during the experimental evaluation show that the approach is able to consume less power in almost all the adopted benchmarks with a negligible impact on the system.