CuSH: Cognitive Scheduler for Heterogeneous High Performance Computing System


Reproduced by: Vanamala Venkataswamy, Swaroopa Dola

12/06/2019
Motivation

Resource Management is everywhere

- Cluster scheduling
- Video streaming
- Congestion Control
- Internet telephony
- Virtual machine placement

source: [6]
Motivation

An Example online multi-resource allocation problem, e.g. CPU, memory.

Many factors to consider: NP-hard

source: [6]
Motivation

- Increased complexity of resource management algorithms.
- Resource manager/scheduler should account for data locality, levels of parallelism, frequency of collective synchronizations etc.
- Cost model complexity for a job scheduler
  - job dimensions
  - queue size
  - execution time and available resources
  - job locality - shared datasets
Motivation

Current approach

- Assume a simple system model
- Come up with a set of heuristics
- Iteratively test and tune the heuristics in real system

Alternative approach

- Reinforcement Learning: Learning via interacting with the environment.
Background - Schedulers

- Job placement policies
  - First Come First Serve (FCFS)
  - Shortest Job First (SJF)
  - Dominant Resource Fairness (DRF)
  - Least Attained Service (LAS)
  - more...

- Traditional Heuristics based schedulers use a combination of these policies to maximize/minimize some objective function.
In RL, the data is not **Independent and Identically Distributed**. The outcome depends on the previous state(s) and action(s).

**Reward Hypothesis**: All goals can be described as maximising expected cumulative reward.
Related Work

- **DeepRM**: Uses RL for cluster scheduling by modeling the cluster state using image-like representation. (2016)

- **Gandiva**: Utilizes domain-specific knowledge to improve latency and efficiency of training DL models in a GPU cluster. (2018)

- **Decima**: Uses RL for scheduling job in Tensorflow like framework. Decima heavily focuses on DL jobs that have DAG like dependencies, optimizing for placing DAG tasks on the cluster. (2019)
Claim / Target Task

• **Claim**: Reinforcement Learning agent can learn better scheduling policies for given cluster constraints than heuristics based schedulers.

• **CuSH**
  – Employs DNN and Reinforcement Learning to achieve optimal performance.
  – Learns to make better scheduling choices by training on a dataset that contains jobs history, available resources and performance characteristics.
An Intuitive Figure Showing WHY Claim

Figure 1: CuSH hierarchical agents trained via RL process.

Reference: CuSH paper
Proposed Solution

- Scheduler as two level system i.e two separated DNNs for job selection and policy allocation.

- The reward function dynamically adjusts based on application.

- RL environment as cluster with
  - N - no. of jobs
  - R - resources
  - Sr - Fixed no.of resource per node
  - Q - jobs to be scheduled
Proposed Solution

• Two allocation policies
  – Depth-first policy: assign requested resources utilizing as few nodes as possible.
  – Breadth-first policy: assign requested resources utilizing as many nodes as possible

• Two different workloads
  – Compute intensive
  – Network intensive
Cluster scheduling problem setting

- Allocate multiple resources
- Resource requests are known
- Non-preemptive jobs

**Goal**: Minimize averaged normalized turnaround time

source: [6]
Two main scheduler modules

1. Job selector module (JSM)
2. Policy selector module (PSM)

Figure 1: CuSH hierarchical agents trained via RL process.
RL - Observe

Figure 1: CuSH hierarchical agents trained via RL process.

source: [6]
Naively considering all possible state/action pairs will be exponential cost.

Solution: Sequential allocation
Naively considering all possible state/action pairs will be exponential cost.

Solution: Sequential allocation
Get Reward:

A penalty of \((-1/j_{ob\_len})\) for every unfinished job
The $\theta_1$ are network parameters of the JSM and $\theta_2$ parameters of the policy selector.

Then $\pi\theta_1$ and $\pi\theta_2$ as the JSM and PSM networks, respectively.
Experimental Strategy

- CuSH: Code not open sourced

- DeepRM [2][6] by MIT is open source. CuSH is based on this work.
## CuSH vs. DeepRM

<table>
<thead>
<tr>
<th>Differences</th>
<th>CuSH</th>
<th>DeepRM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td>JSM and PSM</td>
<td>JSM</td>
</tr>
<tr>
<td><strong>Key Metric</strong></td>
<td>averaged normalized turnaround time</td>
<td>average job slowdown</td>
</tr>
<tr>
<td><strong>Input format</strong></td>
<td>Wait time for jobs in queue</td>
<td>Binary matrices</td>
</tr>
<tr>
<td><strong>Job duration</strong></td>
<td>Bounded</td>
<td>Unbounded</td>
</tr>
<tr>
<td><strong>Resource Locality</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Workload Type</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Experimental Results

Experimental Results

Conclusion and Future Work

• Resource management using traditional Heuristics based scheduling does not always give best schedule.

• Better job scheduling subject to constraints can be achieved using DNN and RL.

• CuSH - an RL implementation outperforms the best heuristic-based approaches, delivering up to 19% lower normalized turnaround time.
<table>
<thead>
<tr>
<th>Paper selection</th>
<th>Vanamala, Prof. Qi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper review slides</td>
<td>Equal contribution</td>
</tr>
<tr>
<td>Experiments</td>
<td>Individually ran DeepRM code and generated results/graphs</td>
</tr>
<tr>
<td>Slides with results</td>
<td>Equal contribution</td>
</tr>
</tbody>
</table>
Questions?

Thank You for your attention!
References


Data Summary

Input for the NN

– Merge cluster and queue into one matrix representation.
– The cluster nodes are concatenated together along the x-axis, forming R matrices of $N \times Sr \times T$ size, which is the same size of the representation of the waiting jobs.
– The input of the job scheduler module is a three-dimensional matrix of size $(R N \cdot Sr) \times (T) \times (Q + 1)$
1. Job selector module (JSM)
   - Current state as input image: jobs in the cluster, waiting jobs, resources.
   - a CNN using 16 2x2 filters, stride=1 and without padding followed by a ReLU, batch normalization and a softmax layer to predict probability for each action.
2. **Policy selector module (PSM)**

- goal of selecting which policy to use to allocate a job that has to be scheduled.
- The module is trained with policy gradient.
- The value return $v_t$ is only based on the local action and its reward value $r_t$.
- The return is the locality penalty ($v_t = r_t = p_j$), that is calculated using the projected workloads data.
Figure 3: Example of environment (state) representation. This representation shows a cluster of 2 nodes (each one with 2 CPUs and 4 GPUs) and a queue (job slots) of 3 waiting jobs.

source: cuSH paper
Conclusion and Future Work

- Current model requires workload type to be specified by the user.
- Better approach would be to use dynamic scheduling
  - unspecified job types can be classified as “unknown”
  - After few executions, the job type can be automatically classified with a ML model
DeepRM Architecture

Figure 1: Reinforcement Learning with policy represented via DNN.