Multi-Resource Packing for Cluster Schedulers

Robert Grandl
Aditya Akella
Srikanth Kandula
Ganesh Ananthanarayanan
Sriram Rao
Diverse Resource Requirements

Tasks need varying amounts of each resource
  - E.g., Memory [100MB to 17GB]
    CPU [2% of a core to 6 cores]

Demands for resources are not correlated
  - Correlation coefficient across resource demands \( \in [-0.11, 0.33] \)

Need to match tasks with machines based on resource
Current Schedulers do not Pack

Slots allocated purely on fairness considerations

Cluster [18 Cores, 36 GB] / Job: [Task Prof.], # tasks

<table>
<thead>
<tr>
<th></th>
<th>A (1 Core, 2 GB), 18</th>
<th>B (3 Cores, 1 GB), 6</th>
<th>C (3 Cores, 1 GB), 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6 tasks</td>
<td>6 tasks</td>
<td>6 tasks</td>
</tr>
<tr>
<td>B</td>
<td>2 tasks</td>
<td>2 tasks</td>
<td>2 tasks</td>
</tr>
<tr>
<td>C</td>
<td>2 tasks</td>
<td>2 tasks</td>
<td>2 tasks</td>
</tr>
</tbody>
</table>

Durations:

Current Schedulers

A: 3t
B: 3t
C: 3t

Packer Schedulers

A: t
B: 2t
C: 3t

33% improvement

Resources used: DRF share = 1/3

Current Schedulers

Packer Schedulers

Resources used: Packer

36 GB 6 GB 6 GB
It is all about packing?

Multi-dimensional bin packing is NP-hard for #dimens. ≥ 2
- Several heuristics proposed
- But they do not apply here ...
  \[
  \text{size of the ball, contiguity of allocation, resource demands are elastic in time}
  \]

Will perfect packing suffice?

Competing objectives:
- Cluster utilization vs.
- Job completion times vs.
- Fairness
Intuition behind the solution

Something reasonably simple and which can be applied

Cluster efficiency

Job completion time

Performance

Fairness
Tetris

Pack tasks along multiple resources

- Cosine similarity between task demand vector and machine resource vector

Multi-resource version of SRTF

- Favor jobs with small remaining duration and small resource consumption

Incorporate Fairness

- Fairness knob $f \in (0, 1]$
  
  $f \to 0$ close to perfect fairness
  
  $f = 1$ most efficient scheduling

(simplified) Scheduling procedure

1: while (resources R are free)
2:  among $[FJ]$ jobs furthest from fair share
3:    score (j) =
4:    \[ \max_{\text{task } t \in j, \text{ demand}(t) \leq R} A(t, R) + \varepsilon T(j) \]
5:  pick $j^*, t^* = \text{argmax score}(j)$
6:  $R = R - \text{demand}(t^*)$
7: end while
Task Requirements and resource usages

Learning task requirements

- From tasks that have finished in the same phase
- Coefficient of variation $\in [0.022, 0.41]$
- Collecting statistics from recurring jobs
- Peak usage demands estimates for tasks

Resource Tracker

- measure actual usage of resources
- enforce allocations
- aware of activities on the cluster other than tasks assignment: *ingest and evacuation*
Evaluation

Prototype atop Hadoop 2.3
- Tetris as a pluggable scheduler to RM
- Implement RT as a NM service
- Modified AM/RM resource allocation protocol

Large scale evaluation
Cluster capacity: 250 Nodes
4 hour synthetic workload
60 jobs with complementary task demands

Reduces average job duration by up to 40%
Reduces makespan by 39%
Evaluation

**Trace-driven simulation**

Facebook production traces analysis

- **Fairness knob**: fewer than 6% of jobs slow down; by not more than 8% on average
- Knob value of 0.75 offers nearly the best possible efficiency with little unfairness

- Fairness knob: fewer than 6% of jobs slow down; by not more than 8% on average
- Knob value of 0.75 offers nearly the best possible efficiency with little unfairness
Conclusion

Identify the importance of scheduling all relevant resources in a cluster

- Resource Fragmentation + Over-allocation and Interference

New scheduler that pack tasks along multiple resources

- Reduce makespan + Job Completion Time

Enable a trade-off between packing efficiency and fairness

Fairness Knob

Come and see our poster!