Phoenix: A Constraint-aware Scheduler for Heterogeneous Datacenters

Prashanth Thinakaran, Jashwant Gunasekaran, Bikash Sharma, Mahmut Kandemir, Chita Das
Executive Summary

› Problem: Heterogeneity agnostic datacenter schedulers leads to poor placement choices of jobs
  • Schedulers Ignoring hardware and application level heterogeneity

› Constraints are used as a medium
  • To express task level heterogeneity (Eg., Latency sensitive, Batch)
  • To expose hardware level heterogeneity (Eg., ISA, Clock speed, Accelerators)
  • To ensure task performance guarantees to ensure QoS

› Phoenix is a constraint-aware scheduler that is:
  • Heterogeneity-aware and hybrid hence scalable
  • Uses real-time CRV metric for task reordering at peak congestions optimizing for tail latencies
  • Improves the 99th percentile (tail) latency by 1.9x across production cluster traces
Outline

- Scheduler Design Paradigm
- Motivation
- Modeling and synthesizing task constraints
- Phoenix architecture
- Results
Scheduler Design Paradigm

- **Centralized**
  - Mesos
  - Borg

- **Distributed**
  - Yacc-D

- **Hybrid Schedulers**
  - Mercury
  - Choosy

- **Constraint aware**
  - Constraint unaware

- **Number of jobs executed per day**
  - 10M
  - 100M
  - 1B
  - 10B
  - 100B

- **Task binding to Queue**
  - Early
  - Late

- **Control Plane**
  - Early to Late
Outline

• Scheduler Design Paradigm
• Motivation
• Modeling and synthesizing task constraints
• Phoenix architecture
• Results
Constraint share in Google traces

- ISA (x86, ARM): 74%
- Number of cores: 17%
- Maximum Disks: 8%
- Minimum Disks: 1%
Task placement constraints

- Constraint-based Job Requests in Cloud Schedulers
  - More than 50% of all tasks subscribe to task constraints
  - Eg., A job may request two server nodes belonging to x86 with at least 1 Gbps of network speed between them
  - Constraint subscription surges
    - Impact other unconstrained tasks
    - Root cause for tail-latencies
Outline

• Scheduler Design Paradigm
• Motivation
• Modeling and synthesizing task constraints
• Phoenix architecture
• Results
Synthesizing Task Constraints

- Publicly available Google’s cluster workload traces [1]
  - Hashed constraint values were correlated with constraint frequency vector proposed in [2]

- Yahoo, Cloudera synthetically generated constraints
  - Benchmarking model proposed in [1] to characterize and generate constraints for tasks
  - Cross-validated & accuracy is close to 87%

- 50% of tasks are constrained

- 33% of jobs demand two constraints but only 12% of it could be satisfied

- As incoming jobs demand more constraints it become difficult to satisfy all of them.
Job Queuing Delays

- High tail latency for resource constrained tasks
  - Average 2 to 2.5x at tail incase of Eagle and Yacc-d
- High volume of scheduling requests demands distributed scheduling
Job response times vs Cluster Load

- 99th percentile job response times shooting up for all traces
- More the system utilization more the response time degradation

Need for a scalable scheduler that could handle tasks with multiple constraints

- More the system utilization more the response time degradation
Outline

• Scheduler Design Paradigm
• Motivation
• Modeling and synthesizing task constraints
• Phoenix architecture
• Results
Phoenix architectural overview
SRPT fails at tail of constrained jobs at higher utilization
utilization > threshold

CRV Monitor

DS

Worker 1
Worker 2
Worker 3
Worker 4

CRV reordering
• CRV monitor keeps track of Constraint Resource Vector (CRV)

• Demand and supply ratio of every constraint at every machine is updated for every heartbeat interval

• P-K based queue waiting time estimators for admission control

• When CRV increase beyond a set threshold CRV based reordering is initiated
Outline

• Scheduler Design Paradigm
• Motivation
• Modeling and synthesizing task constraints
• Phoenix architecture
• Results
Phoenix compared to Eagle

*Lower the better
Phoenix compared to Hawk/Sparrow

*Lower the better

---

Google trace- Hawk

Google trace- Sparrow
Summary

• Phoenix is a hybrid constraint-aware scheduler

• Dynamically adapts itself at high resource demands using CRV metric based reordering

• Improves tail-latency by an average of 1.9x for heavily resource constrained tasks

• Not affecting long job response times and fairness of other unconstrained tasks
prashanth@cse.psu.edu

http://www.cse.psu.edu/hpcl/index.html
CRV statistics

- Number of task reordering is reliant on inter arrival patterns of jobs

- Average utilization of the cluster was 80%

<table>
<thead>
<tr>
<th>Workload</th>
<th>Nodes</th>
<th>Constrained Tasks</th>
<th>Unconstrained Tasks</th>
<th>Reordered tasks</th>
<th>Short jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yahoo</td>
<td>5000</td>
<td>251404</td>
<td>263240</td>
<td>5227</td>
<td>91.56%</td>
</tr>
<tr>
<td>Cloudera</td>
<td>15000</td>
<td>1972428</td>
<td>1925052</td>
<td>817470</td>
<td>95%</td>
</tr>
<tr>
<td>Google</td>
<td>15000</td>
<td>6602875</td>
<td>6265616</td>
<td>1717145</td>
<td>90.2%</td>
</tr>
</tbody>
</table>
Constraint Modeling

• Types of constraints
  • Hard constraints -> Eg., Minimum memory, No. of cores
  • Soft constraints -> Clock speed, Network bandwidth
  • Affinity constraints -> HDFS Data locality, MPI tasks

• Constraint support in existing schedulers
  • Mesos - Locality preferences of tasks
  • Kubernetes - It is on their roadmap to support soft & hard

• Affinity constraints impact scheduling delays 2x to 4x
Scheduling Optimization Metrics

• Job response Times
  • Hybrid schedulers uses SRPT for job turnaround times
  • Comes at the cost of fairness of the other unconstrained jobs

• Admission Control
  • Negotiating for Jobs with multiple resource constraints
  • Hard to soft relaxation of constraints

• Late Binding
  • Avoiding early commit and reducing queue waiting times especially for short jobs.

• Load Balancing
  • Job stealing techniques improves the overall resource utilization but not always the case.
  • Task migration overheads and constraint preferences violation should be taken into account
Queuing delays of Google jobs using SRPT

- Sporadic peaks and valleys of job submission pattern
  - At peaks, heavy tail latency leads to QoS violations of short jobs
  - Queuing delays cascade in to other unconstrained tasks
  - Naive SRPT based queue management fails to deliver
- Constrained tasks scheduled by Hawk and Yacc-d also experience 2 to 2.5x queueing delays (repetition of information)
Evaluation Methodology

• Trace-driven simulator built on top of Eagle and Sparrow

• Three production datacenter traces were used for evaluation
  • Yahoo, Cloudera & Google

• Cluster inter arrival rate is bursty and unpredictable with peak to median ratio from 9:1 to 260:1
Impact on unconstrained jobs

![Bar chart showing impact on unconstrained jobs](chart.png)