Tetris

Multi-Resource Packing for Cluster Schedulers

Robert Grandl, Ganesh Ananthanarayanan,

Srikanth Kandula, Sriram Rao, Aditya Akella

Microsoft Research

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Performance of cluster schedulers

We find that:

- *Resources are fragmented i.e. machines run below capacity*
- *Even at 100% usage, goodput is smaller due to over-allocation*
- *Pareto-efficient multi-resource fair schemes do not lead to good avg. performance*

Tetris

Up to 40% improvement in makespan[1] and job completion time with near-perfect fairness

[1] *Time to finish a set of jobs*

Findings from Bing and Facebook traces analysis

Applications have (very) diverse resource needs

- Tasks need ***varying amounts of each resource***
- ***Demands*** for resources ***are weakly correlated***

Multiple resources become tight

This matters, because no single bottleneck resource in the cluster:

- E.g., enough cross-rack network bandwidth to use all cores

Upper bound on potential gains

- ***Makespan reduces by ≈ 49%***
- ***Avg. job completion time reduces by ≈ 46%***

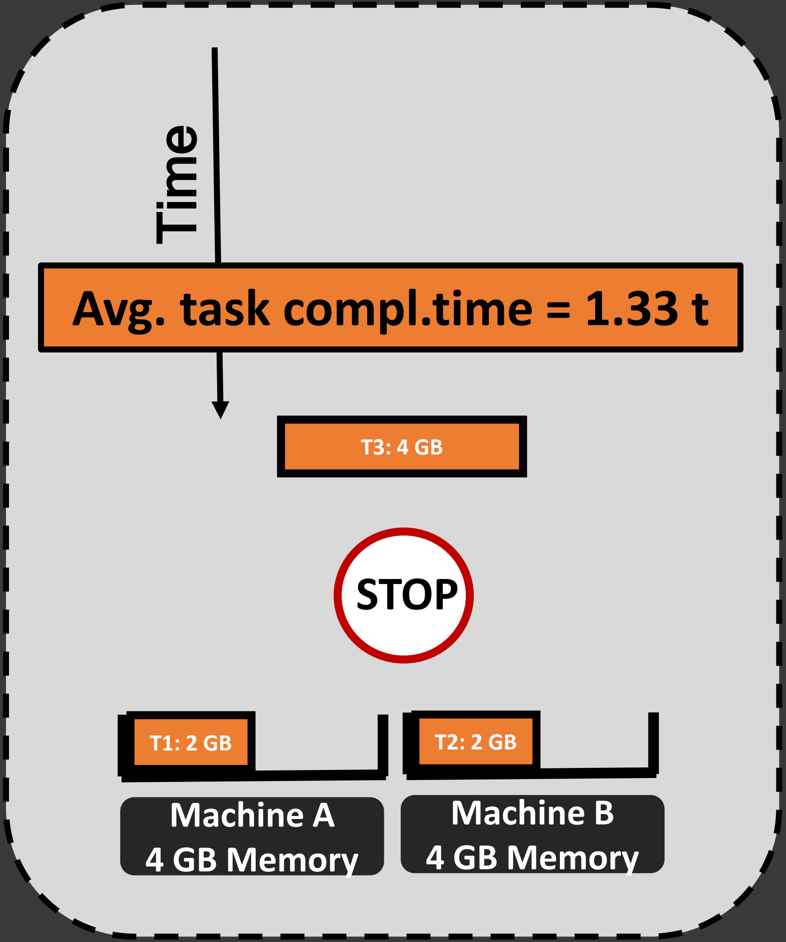
Why so bad #1

Production schedulers neither pack tasks nor consider all their relevant resource demands

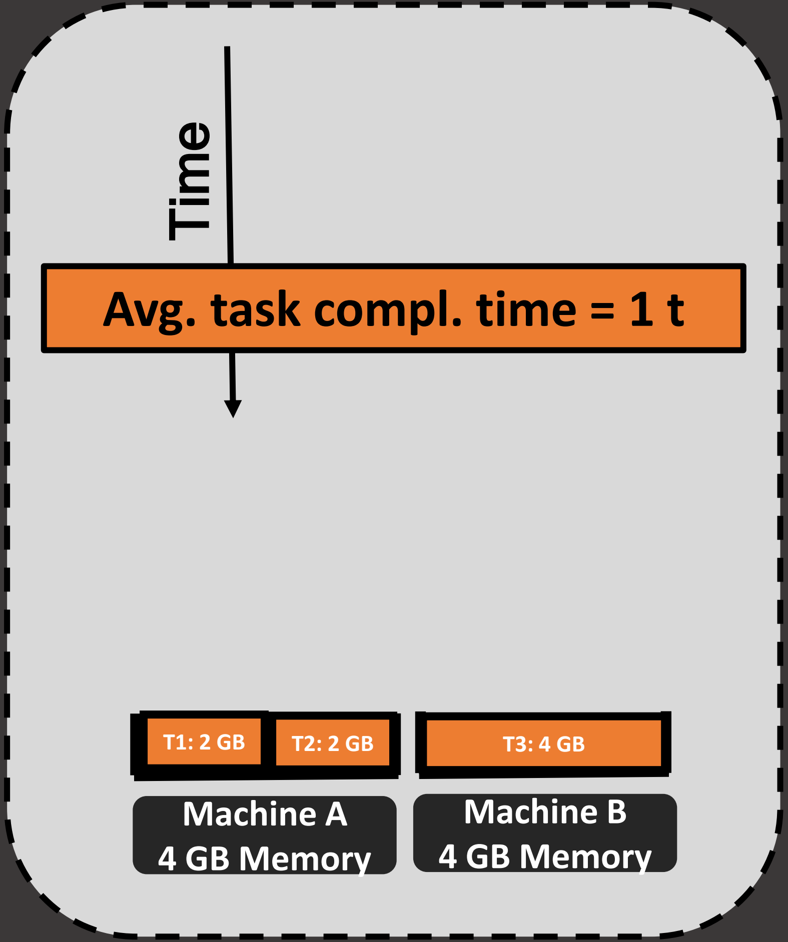
#1 Resource Fragmentation

#2 Over-allocation

Resource Fragmentation (RF)



Current Schedulers



"Packer" Scheduler

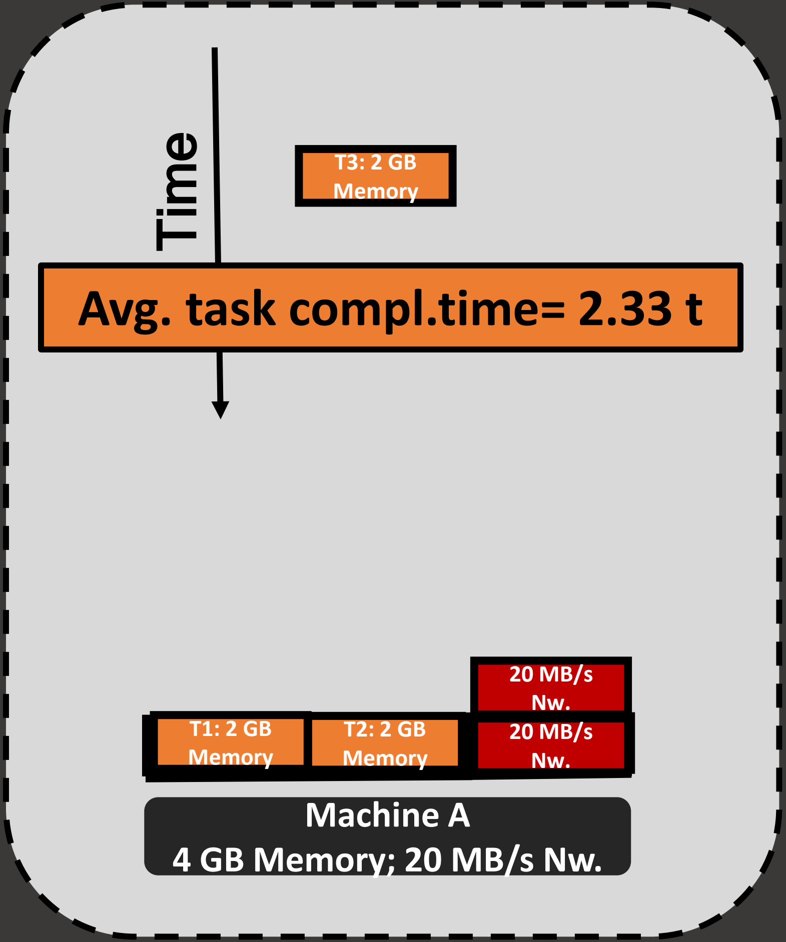
Current Schedulers

Are not explicit about packing.

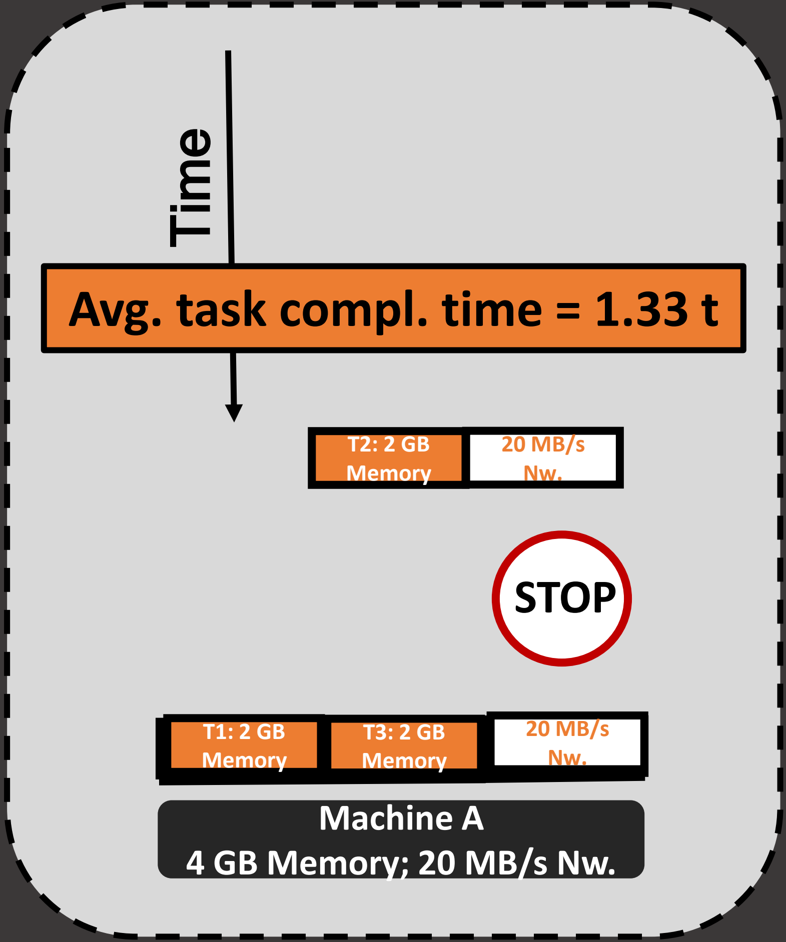
Allocate resources per **slots, fairness.**

RF increase with the number of resources being allocated !

Over-Allocation



Current Schedulers



“Packer” Scheduler

Current Schedulers

Not all of the resources are **explicitly allocated**

E.g.,**disk** and **network** can be **over-allocated**

Why so bad #2
Multi-resource Fairness Schemes do not solve the problem

Example in paper

Packer vs. DRF: makespan and avg. completion time improve by over 30%

Work Conserving != no fragmentation, over-allocation

Pareto[1] efficient != performant

- **Treat cluster as a big bag of resources**
  - *Hides the impact of* ***resource fragmentation***
- **Assume job has a fixed resource profile**
  - *Different tasks in the same job have different demands*
  - *How the job is scheduled impacts jobs' current resource profiles*
  - *Can schedule to create complementarity*

[1]no job can increase its share without decreasing the share of another

Current Schedulers

1. Resource Fragmentation
2. Over-Allocation
3. Fair allocations sacrifice performance

Competing objectives

Cluster efficiency

vs.

Job completion time

vs.

Fairness

Tetris

1

Pack tasks along multiple resources to improve cluster efficiency and reduce makespan

Theory

Multi-Resource Packing of Tasks

similar to

Multi-Dimensional Bin Packing

APX-Hard[1]

Avoiding fragmentation looks like:

- ❑ Tight bin packing
- ❑ Reduce # of bins → ***reduce makespan***

[1]APX-Hard is a strict subset of NP-hard

Balls could be tasks

Bin could be machine, time

Practice

Existing heuristics do not directly apply:

- ❑ Assume balls of a fixed size
- ❑ Assume balls are known apriori

- vary with time / machine placed
- elastic
- cope with online arrival of jobs, dependencies, cluster activity

Tetris

A packing heuristic

Fit

- Tasks resources demand vector
- Machine resource vector

<

Alignment score (A)

"A" works because:

1. Check for fit to ensure **no over-allocation**

✓ Over-Allocation

2. Bigger balls ***get bigger scores***

3. Abundant resources ***used first***

✓ Resource Fragmentation

1

Packing heuristic

Tetris

2

Faster average job completion time

Tetris

CHALLENGE

Q: What is the shortest "~~remaining time~~" ?

"**remaining work**" = **remaining # tasks**
&
tasks' resource demands
&
tasks' durations

2

Job Completion Time Heuristic

A job completion time heuristic

- Gives a score **P** to every job
- Extended SRTF to incorporate multiple resources

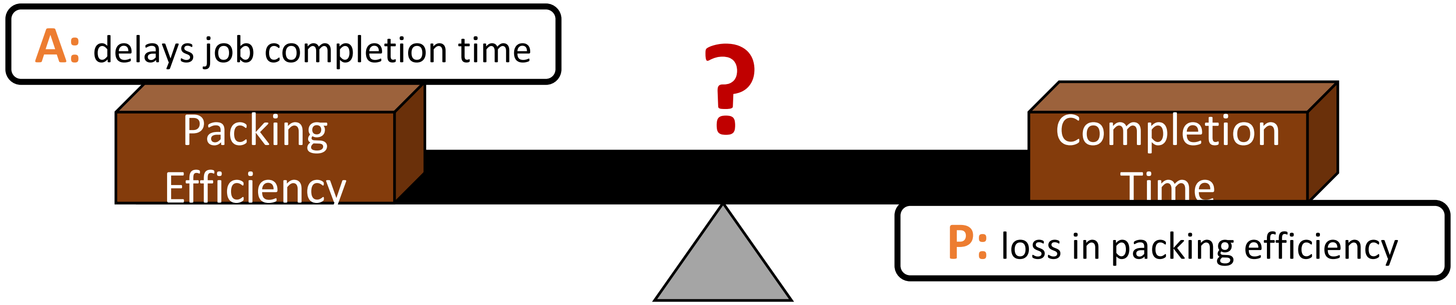
Shortest Remaining Time First[1] (SRTF)

schedules jobs in ascending order of their ***remaining time***

[1]SRTF – M. Harchol-Balter et al. Connection Scheduling in Web Servers [USITS'99]

Tetris

CHALLENGE



2

Job Completion Time Heuristic

Combine A and P scores !

```
among J runnable jobs
 score (j) = A(t, R)+ε P(j)
     max task t in j, demand(t) ≤ R (resources free)
pick j*, t* = argmax score(j)
```

Tetris

3

Achieve performance and fairness

Tetris

Performance and fairness do not mix well in general

But ….
We can get "perfect fairness" and much better performance

- **Packer** says: "***task T*** *should go next to improve* ***packing efficiency***"
- **SRTF** says: "schedule ***job*** *J to improve* ***avg. completion time***"
- **Fairness** says: "**this set of *jobs*** *should be scheduled next*"

Possible to satisfy all three
In fact, happens often in practice

3

Fairness Heuristic

Tetris

Fairness is not a tight constraint

- Lose a bit of fairness for a lot of gains in performance
- Long term fairness not short term fairness

Heuristic

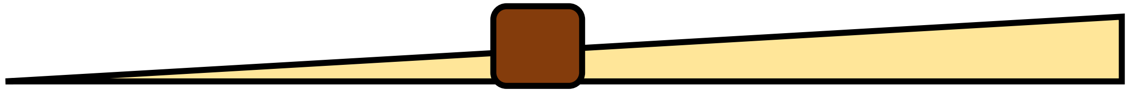
- Fairness Knob, F $\in [0, 1)$

Pick the ***best-for-perf.*** task from among

$\lceil 1-F \rceil$ fraction of jobs furthest from fair share

- ❑ Most unfair
- ❑ Most efficient scheduling

Close to perfect fairness



F = 0 F → 1

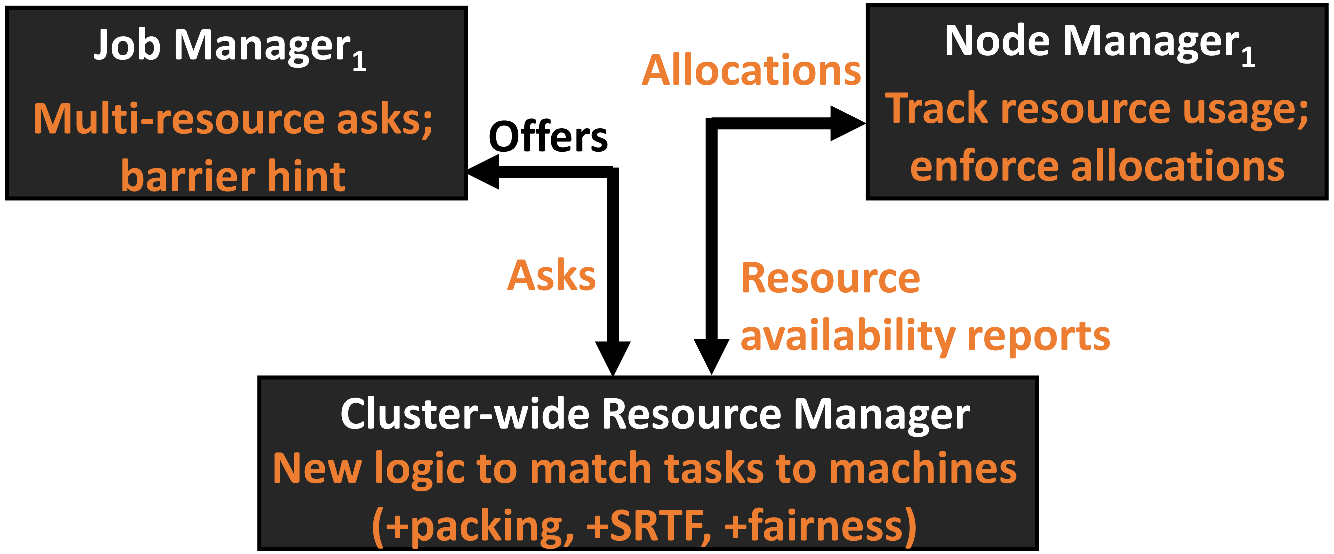
3

Fairness Heuristic

Putting it all together

We saw:

- Packing efficiency
- Prefer small remaining work
- Fairness knob



Yarn architecture

Changes to add Tetris(shown in orange)

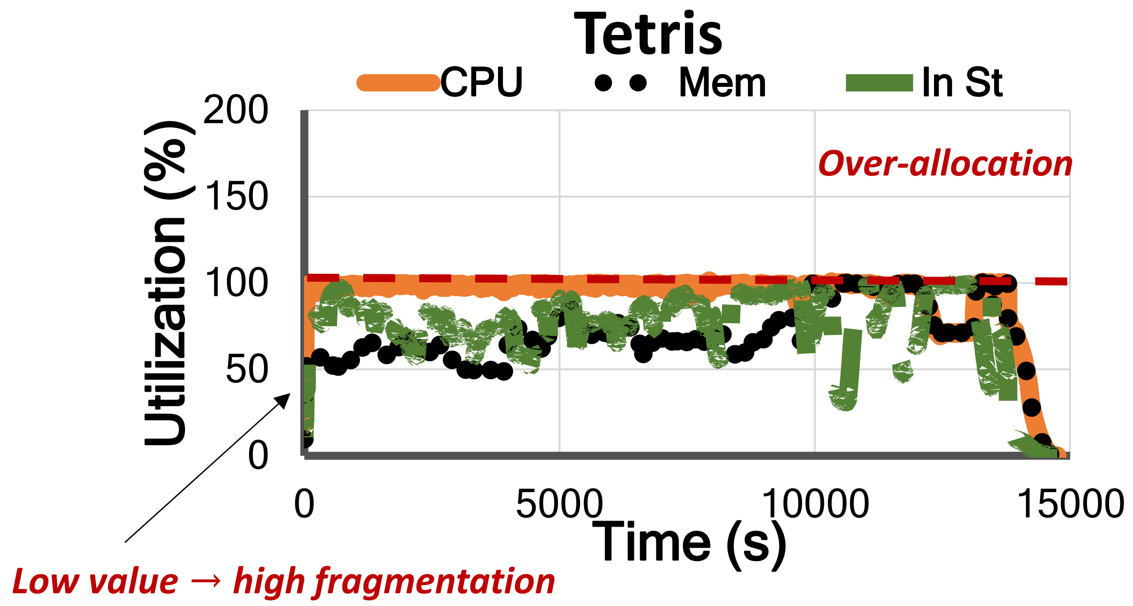
Other things in the paper:

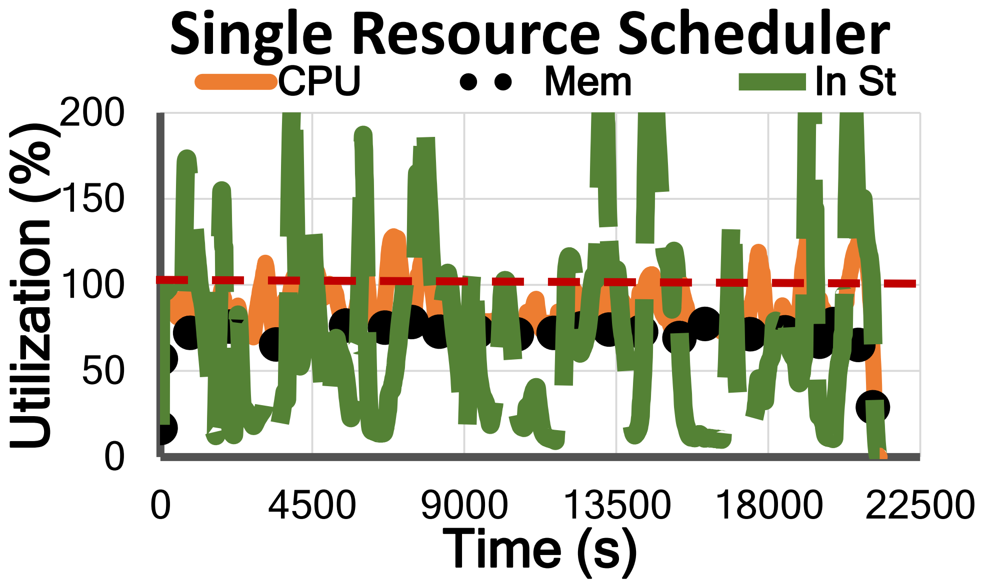
- Estimate task demands
- Deal with inaccuracies, barriers
- Other cluster activities

Evaluation

- Implemented in Yarn 2.4
- 250 machine cluster deployment
- Bing and Facebook workload

Efficiency





Tetris vs.	Makespan	Avg. Job Compl. Time
Single Resource Scheduler	29 %	30 %
Multi-resource Scheduler	28 %	35%

Gains from

- *avoiding fragmentation*
- *avoiding over-allocation*

Fairness

Fairness Knob

- *quantifies the extent to which Tetris adheres to fair allocation*

	Makespan	Job Compl. Time	Avg. Slowdown [over impacted jobs]
No Fairness F = 0	50 %	40 %	25 %
Full Fairness $F \rightarrow 1$	10 %	23 %	2 %
F = 0.25	25 %	35 %	5 %

Tetris

Pack efficiently along multiple resources

Prefer jobs with less “remaining work”

Incorporate Fairness

- Combine heuristics that ***improve packing efficiency*** with those that ***lower*** average ***job completion time***

- Achieving desired amounts of ***fairness can coexist*** with improving cluster performance

- Imp
  sho

We are working towards a Yarn check-in

http://research.microsoft.com/en-us/UM/redmond/projects/tetris/