Multi-tenancy in Datacenters: to each according to his …

Lecture 15, cs262a
Ion Stoica & Ali Ghodsi
UC Berkeley, March 10, 2018

Cloud Computing

- IT revolution happening in-front of our eyes

Above the Clouds: A Berkeley View of Cloud Computing

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz,
Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia
(Comments should be addressed to abovetheclouds@cs.berkeley.edu)

UC Berkeley Reliable Adaptive Distributed Systems Laboratory *
http://radlab.cs.berkeley.edu/

February 10, 2009

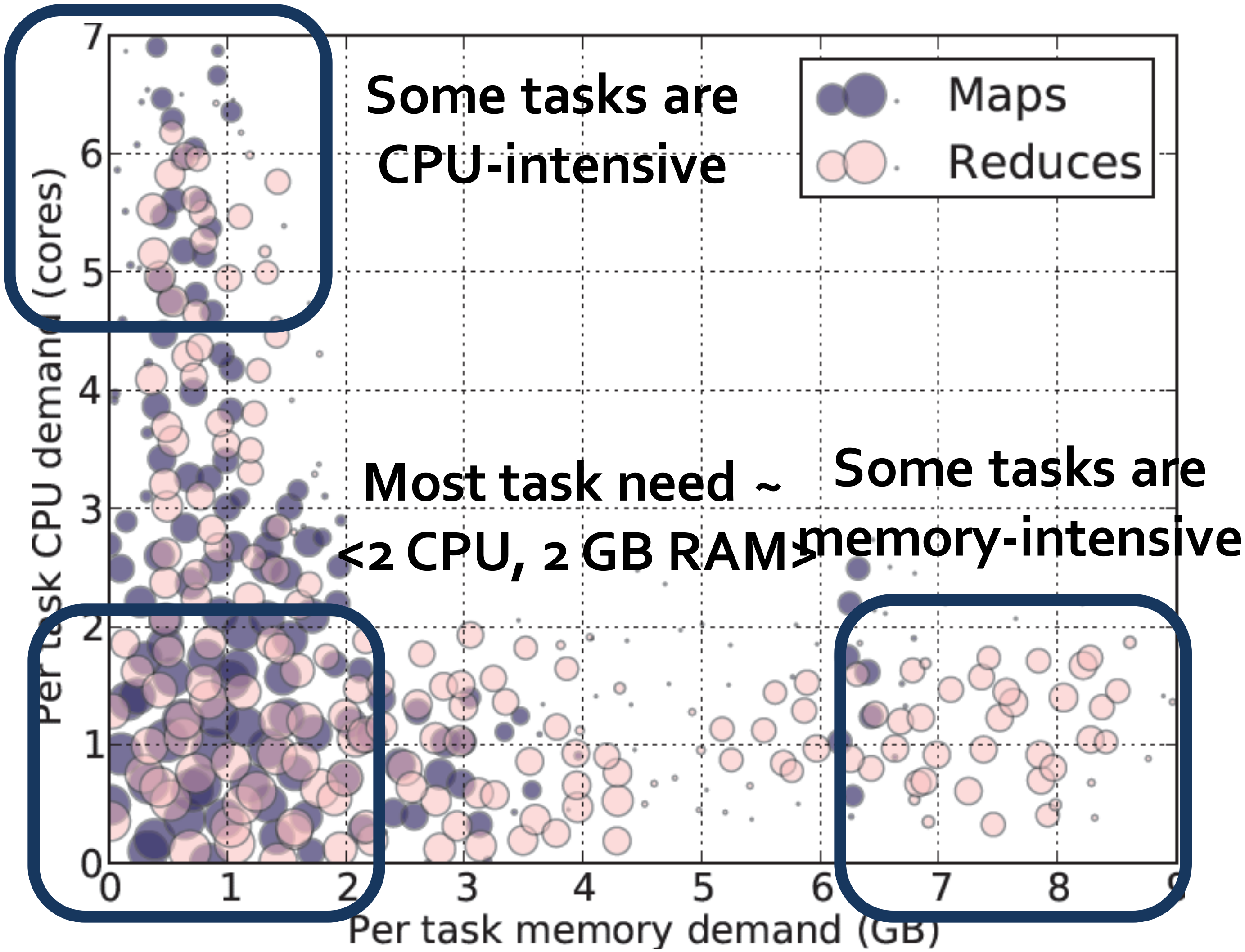
KEYWORDS: Cloud Computing, Utility Computing, Internet Datacenters, Distributed System Economics

Basic tenet of cloud computing

- Consolidate workloads into datacenters
  - Better resource utilization

- Goal: consolidate workloads onto one cluster
  - Now powering most of Twitter, Netflix, eBay, etc

Workload Study



Tasks have heterogeneous resource demands

alig@cs.berkeley.edu

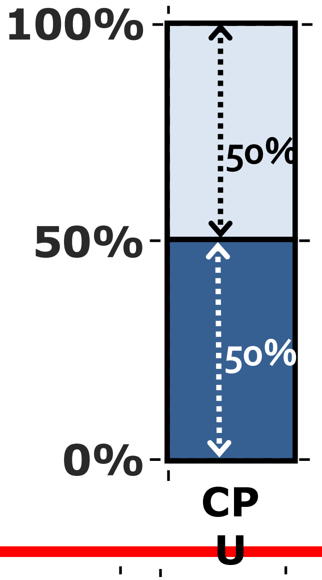
How to allocate resources to jobs with heterogeneous resource demands?

How to allocate resources to jobs with heterogeneous resource demands?

- Terminology
  - Synonymous: jobs, users, applications
  - A job can consist of many tasks
  - A task is a program running on one machine
  - Fine-grained scheduling (schedule one task at a time)

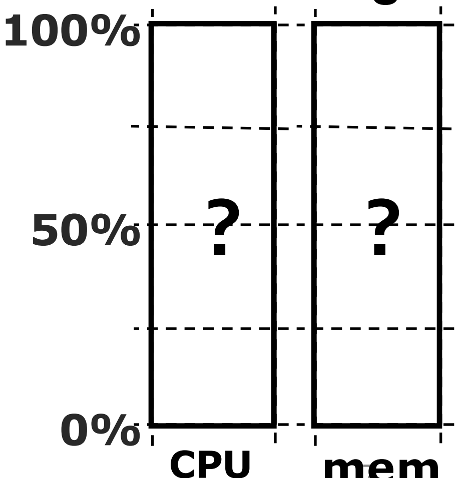
Example Problem

Assume two users with equal entitlement
- Infinite request of tasks



Single *resource example*
- 1 resource: 1,000,000 CPU
- User 1 wants **<1 CPU>** per task
- User 2 wants **<3 CPU>** per task

Multi-resource example
- 2 resources: CPUs & mem
- User 1 wants **<1 CPU, 4 GB>** per task
- User 2 wants **<3 CPU, 1 GB>** per task
- What's a fair allocation?



Why fairness? Equal entitlements?

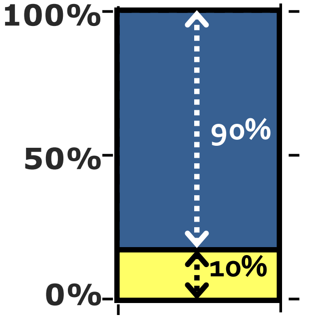
Fairness policy equivalent to isolation policy

Users cannot affect others beyond their fair share

Weighted fairness implements many policies

Not equal: user 1 weight 9, User 2 weight 1, …

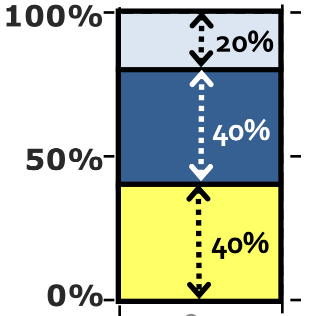
Priority: User 1 weight 10^{10}, User 2 weight 10, …



Fairness generalized by Max-Min Fairness

Handles if a user uses less than her fair share

e.g. user 1 only uses 20% of it's 33% entitlement



Talk from Bird's-eye View

	Allocation Policy
Single-Resource Fairness	**Max-Min Fairness**
Multi-Resource Fairness	↓ **?**

- Fair scheduling well studied in many contexts
  - Surprisingly little work on multi-resource fairness

Multi-resource scenario opens many new fundamental challenges

Talk Outline

- Multi-resource fairness – DRF
- DRF deployments in organizations
- Applying DRF to modern network routers
- Follow-up work on DRF
- Other research

alig@cs.berkeley.edu

Talk Outline

- Multi-resource fairness – DRF
  - What properties do we want?
  - Our proposed solution (DRF)
  - How would an economist solve this?
  - How well does this work in practice?

alig@cs.berkeley.edu

Properties of policies
Share guarantee
Strategy-proofness
Pareto efficiency
Envy-freeness
Single resource fairness
Bottleneck resource fairness
Population monotonicity
Resource monotonicity

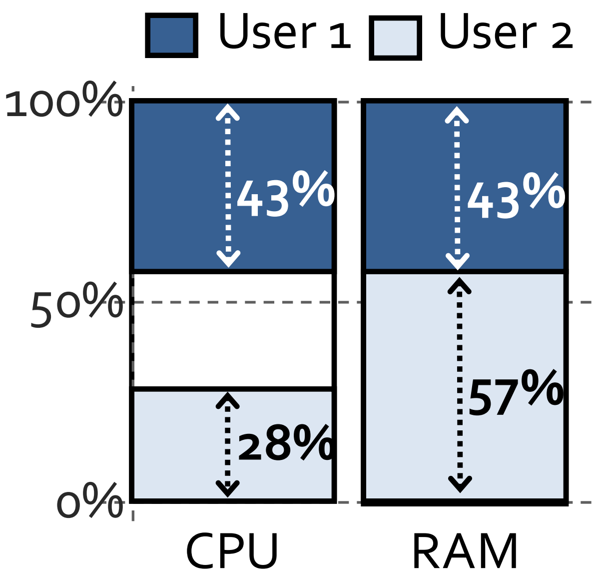
alig@cs.berkeley.edu

Properties of policies
Share guarantee
Strategy-proofness
Pareto efficiency
Envy-freeness
Single resource fairness
Bottleneck resource fairness
Population monotonicity
Resource monotonicity

alig@cs.berkeley.edu

A Natural Policy

- *Asset Fairness*
  - Equalize each user's *sum of resource shares*



Problem
User 1 has < 50% of both CPUs and RAM

Better off in a separate cluster with 50% of the resources

- Asset fairness yields
  - U_1: 15 tasks: 30 CPUs, 30 GB (∑=60)
  - U_2: 20 tasks: 20 CPUs, 40 GB (∑=60)

Share Guarantee

- Every user should get $1/n$ of at least one resource
- Intuition:
  - "You shouldn't be worse off than if you ran your own cluster with 1/n of the resources"

alig@cs.berkeley.edu

Cheating the Scheduler

- Users willing to *game* the system to get more resources
- Real-life examples
  - A familiar company provided dedicated machines to users that could ensure certain level of utilization (e.g. 80%)
  - **Users used busy-loops to inflate utllization**
  - A cloud provider had quotas on map and reduce slots
    Some users found out that the map-quota was low
  - **Users implemented map-reduce in the reduce phase!**

alig@cs.berkeley.edu

Strategy-proofness

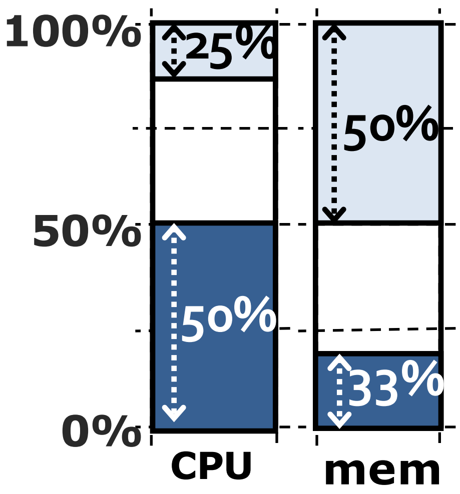
- A user should not be able to increase her allocation by lying about her demand
- Intuition:
  - Users are incentivized to make truthful resource requirements

alig@cs.berkeley.edu

Pareto efficiency

- There should not exist another allocation where at least one user is better off and no user is worse off.

- Avoid inefficient solutions
  User 1 wants **<1 CPU, 4 GB>** per task
  User 2 wants **<3 CPU, 1 GB>** per task



alig@cs.berkeley.edu

Challenge

- Max-min fairness for a single resource trivially satisfies all these properties

- Can we find a multi-resource fair sharing policy that provides:
  - Strategy-proofness
  - Share guarantee
  - Pareto efficiency

alig@cs.berkeley.edu

Talk Outline

- Multi-resource fairness – DRF
  - What properties do we want?
  - Our proposed solution (DRF)
  - How would an economist solve this?
  - How well does this work in practice?

alig@cs.berkeley.edu

Dominant Resource Fairness

- A user's *dominant resource* is the resource she has the biggest share of
  - Example:

    Total resources: **<10 CPU, 4 GB>**

    User 1's allocation: **<2 CPU, 1 GB>**

    Dominant resource is memory as 1/4 > 2/10 (1/5)

- A user's *dominant share* is the fraction of the dominant resource she is allocated
  - User 1's dominant share is **25%** (1/4)

alig@cs.berkeley.edu

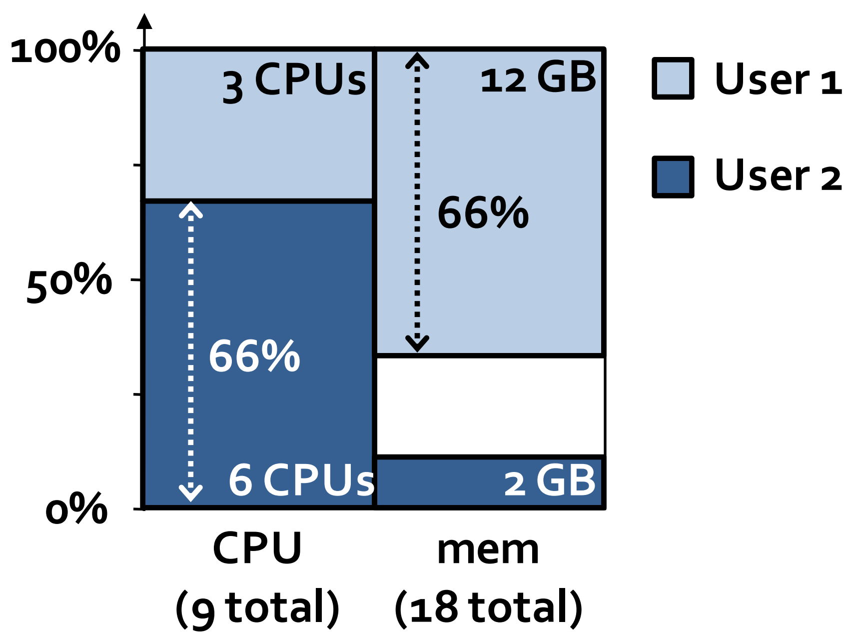
Dominant Resource Fairness (2)

- Apply max-min fairness to dominant shares
  - Equalize the dominant share of the users
  - Example:

    Total resources: **<9 CPU, 18 GB>**

    User 1 demand: **<1 CPU, 4 GB>** dom res: **mem**

    User 2 demand: **<3 CPU, 1 GB>** dom res: **CPU**



Online DRF Scheduler

Whenever there are available resources and tasks to run:

Schedule a task to the user with smallest dominant share

- O(log *n*) time per decision using binary heaps

alig@cs.berkeley.edu

Talk Outline

- Multi-resource fairness – DRF
  - What properties do we want?
  - Our proposed solution (DRF)
  - How would an economist solve this?
  - How well does this work in practice?

alig@cs.berkeley.edu

How would an economist solve it?

- Use pricing
  - Set prices for each good
  - Give each user the same budget
  - Let users buy what they want

- Problem
  - How do we determine the right prices for different goods?

alig@cs.berkeley.edu

The market approach

- Let the market determine the prices

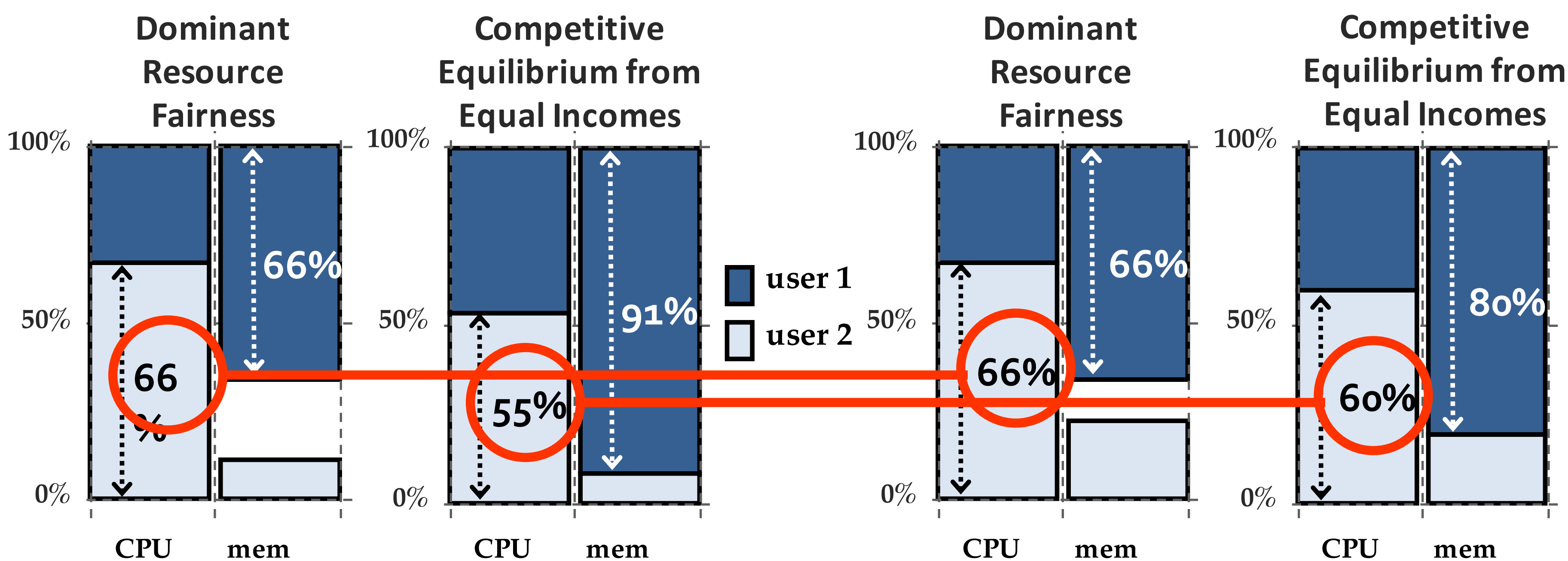
- Competitive Equilibrium from Equal Incomes (CEEI)
  - Give each user 1/n of every resource
  - Let users trade in a perfectly competitive market
  - Analytical solution: max of product of dominant shares

- **Violates strategy-proofness**

alig@cs.berkeley.edu

DRF vs CEEI

- User 1: <1 CPU, 4 GB> User 2: <3 CPU, 1 GB>
  - DRF more fair, CEEI better utilization



- User 1: <1 CPU, 4 GB> User 2: <3 CPU, 2 GB>
  - User 2 increased her share of both CPU and memory

alig@cs.berkeley.edu

Properties of DRF

- We proved DRF is strategy-proof
  - Assuming linear utilities
  - Others proved it's the only policy satisfying Strategy-proofness, sharing incentive, Pareto

Results carried over the economics literature

Properties of Policies

Property	Asset	CEEI	DRF
Share guarantee		✔	✔
Strategy-proofness	✔		✔
Pareto efficiency	✔	✔	✔
Envy-freeness	✔	✔	✔
Single resource fairness	✔	✔	✔
Bottleneck res. fairness		✔	✔
Population monotonicity	✔		✔
Resource monotonicity			

alig@cs.berkeley.edu

Talk Outline

- Multi-resource fairness – DRF
  - What properties do we want?
  - Our proposed solution (DRF)
  - How would an economist solve this?
  - DRF variants?

alig@cs.berkeley.edu

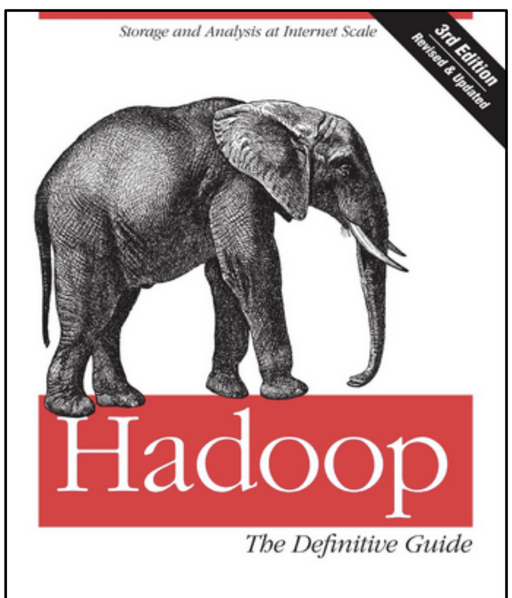
Follow-up papers

	Allocation in Space	Allocation in Time
Single-Resource Fairness	Max-Min Fairness	Fair Queueing
Multi-Resource Fairness	DRF	

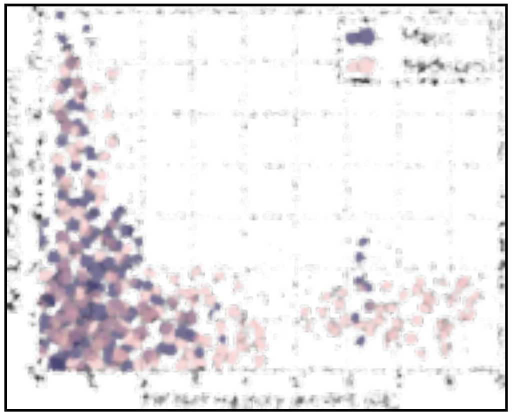
DRFQ broadly applicable: VMs, OSs

DRF in the wild

- DRF de-facto scheduler in Hadoop & Mesos
  - DRF capacity scheduler (HortonWorks)
  - DRF fair scheduler (Cloudera)
  - Mesos cluster of O(10k) nodes at Twitter

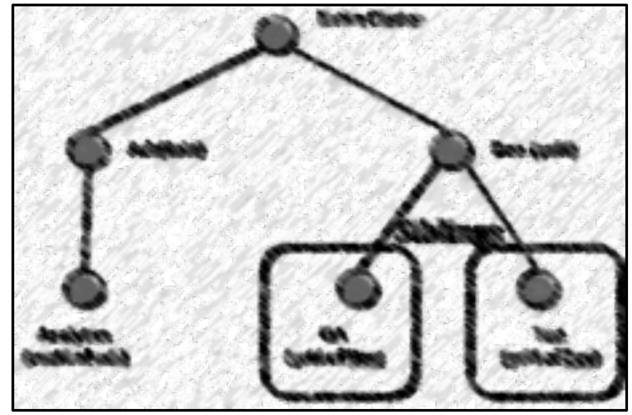


Multi-Resource Scheduling **Hierarchical Policies**



+

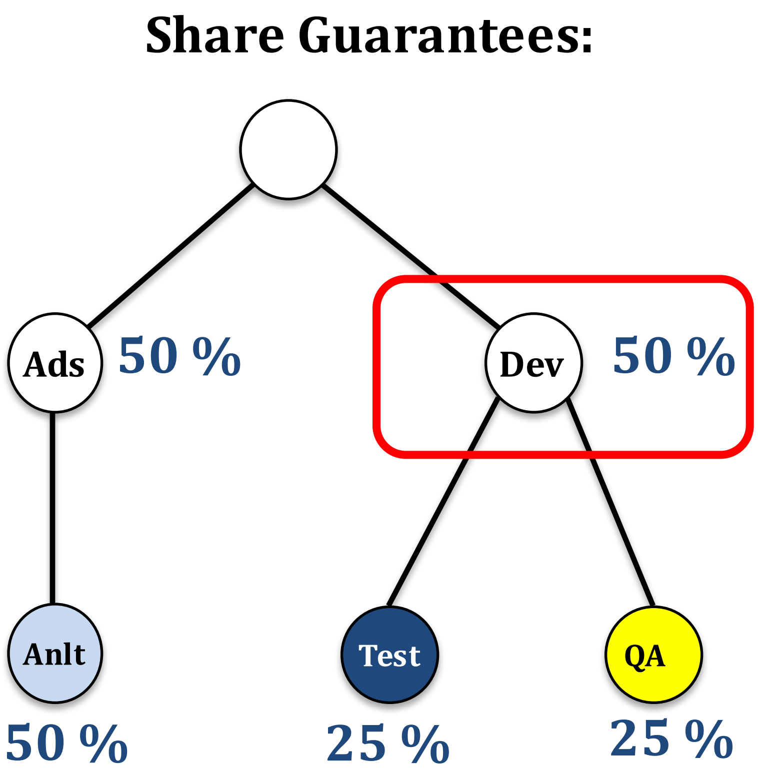
=

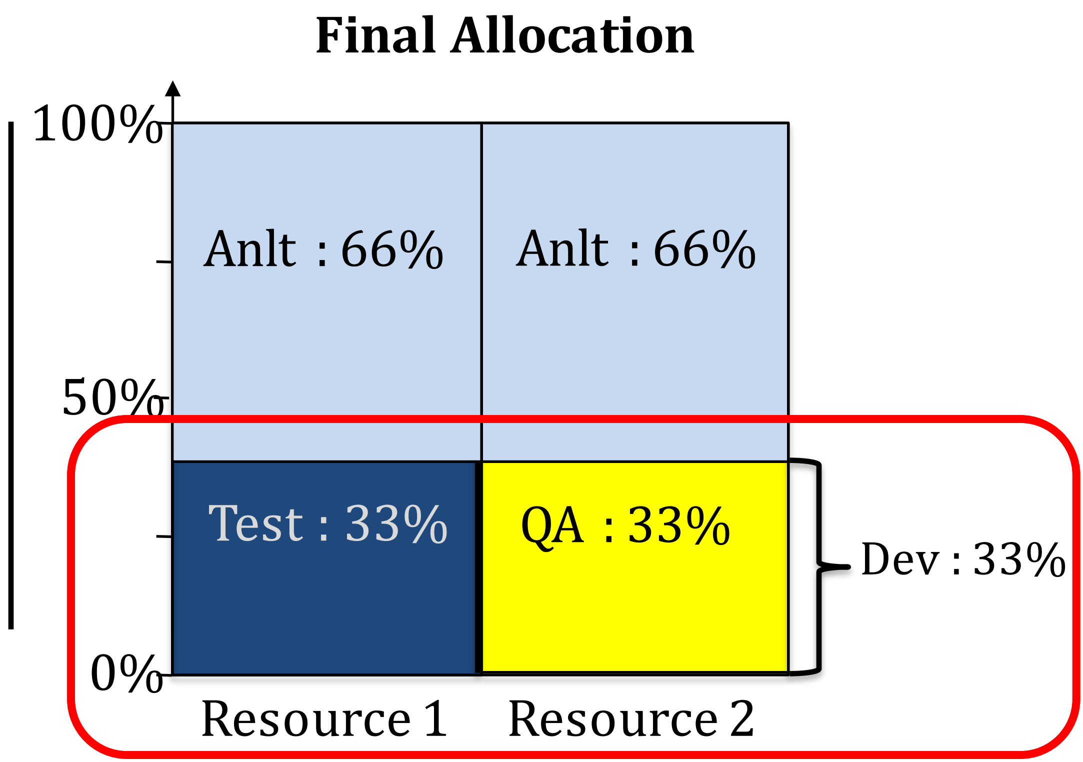


Challenging

- Hadoop DRF schedulers can break down
  - Leave resources unallocated (not Pareto) or
  - Starve users

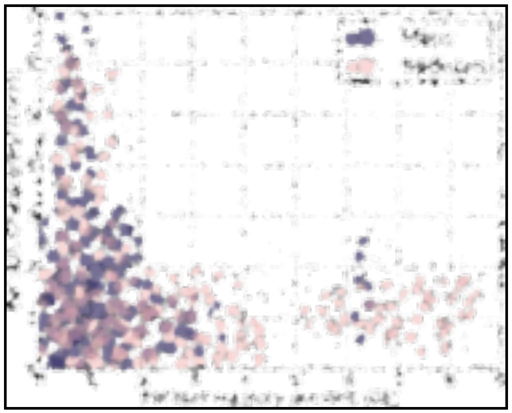
Hierarchical Share Guarantee Violated





Follow-up papers

Dominant Resource Fairness

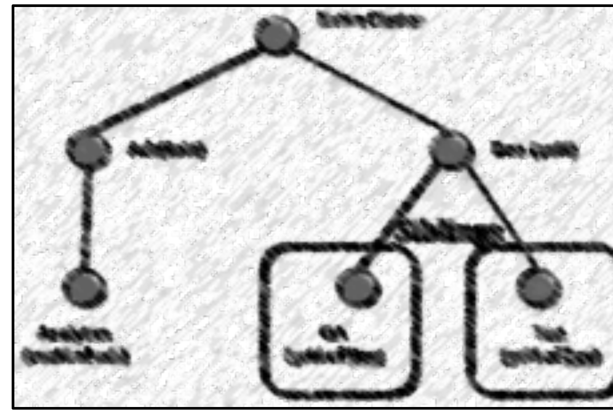


- **Share guarantee**
  1/n share to leafs
- **Pareto efficiency**
  Work-conservation

+

➔

H-DRF



- **Hierarchical share guarantee**
  1/n to every node
- **Pareto efficiency**
  Work-conservation

Talk Outline

- Multi-resource fairness – DRF
  - What properties do we want?
  - Our proposed solution (DRF)
  - How would an economist solve this?
  - DRF variants?
  - DRF evaluation

alig@cs.berkeley.edu

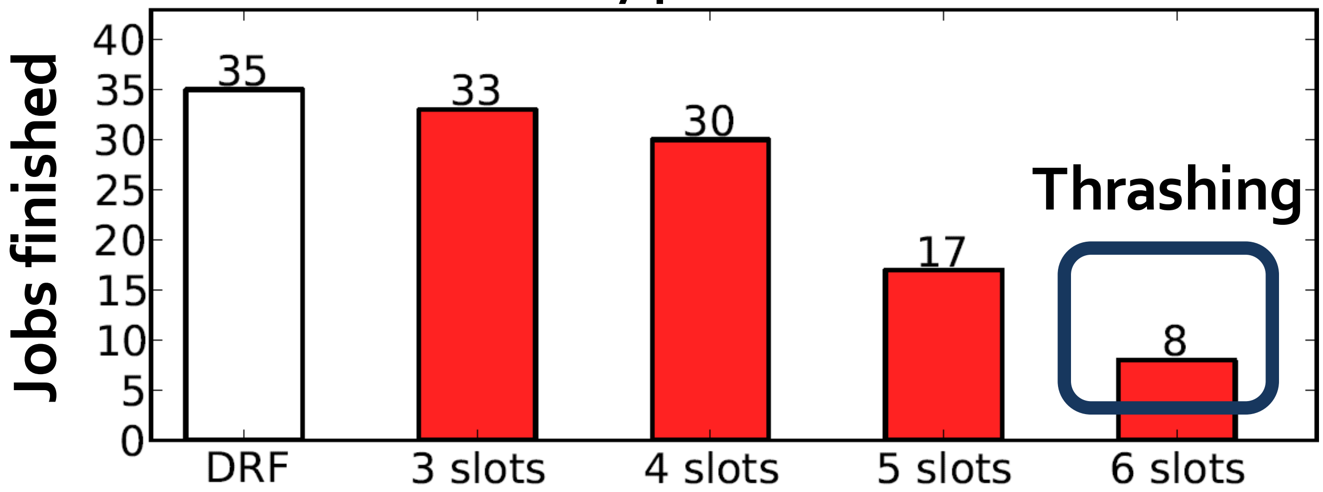
Previous approach: slot-based scheduling

- Hadoop Fair Scheduler
  - Each machine consists of k *slots* (e.g. k=14)
  - Run at most one task per slot
  - Give jobs "equal" number of slots,

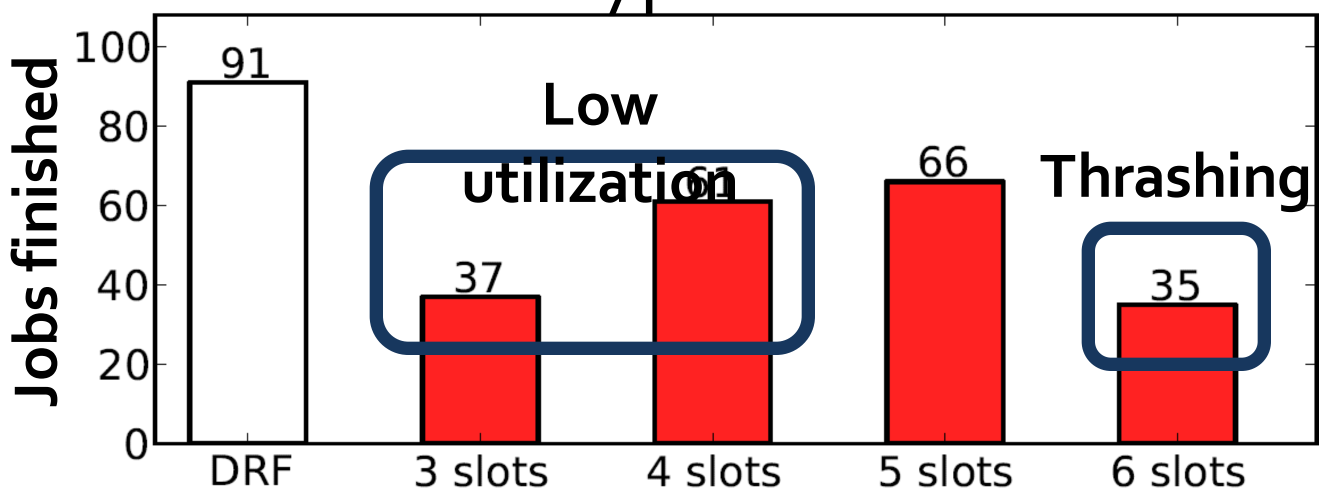
This is what we compare against

Experiment: DRF vs Slots

Number of Type 1 Jobs Finished



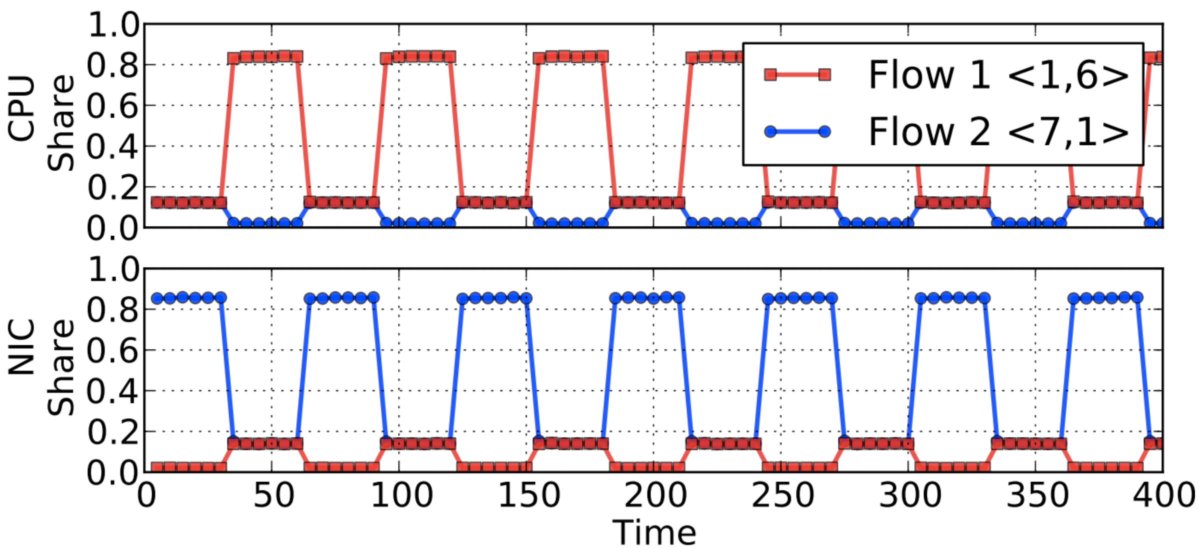
Number of Type 2 Jobs Finished



Type 1 jobs <2 CPU, 2 GB> **Type 2 jobs <1 CPU, 0.5GB>**

State-of-the-art: bottleneck fairness

- 2 flows and 2 res. <CPU μs, NIC μs>
  - Demands <1,6> and <7,1> → *bottleneck unclear*



- Especially bad for TCP and video/audio traffic

TCP and oscillations

- Implemented Bottleneck Fairness in Click
  - Bottleneck determined every 300 ms
  - 1 BW-bound flow and 1 CPU-bound flow

Scenario	Flow 1 (BW-bound)	Flow 2 (CPU-bound)
Running alone	191 Mbps	33 Mbps
Bottleneck	75 Mbps	32 Mbps
DRFQ	160 Mbps	28 Mbps

Oscillations in Bottleneck degrade performance of TCP

Thank you!

Questions?