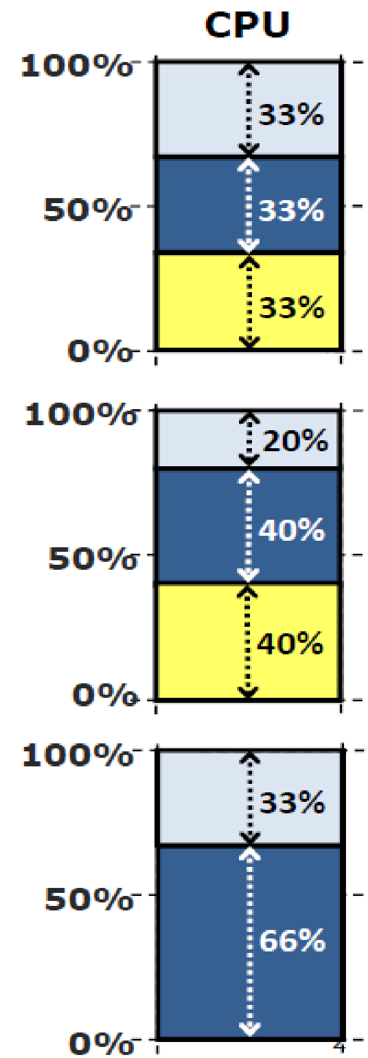
Dominant Resource Fairness: Fair Allocation of Multiple Resource Types

Ali Ghodsi, Matei Zaharia
Benjamin Hindman, Andy Konwinski,
Scott Shenker, Ion Stoica
University of California, Berkeley

NSDI'11

What is fair sharing?



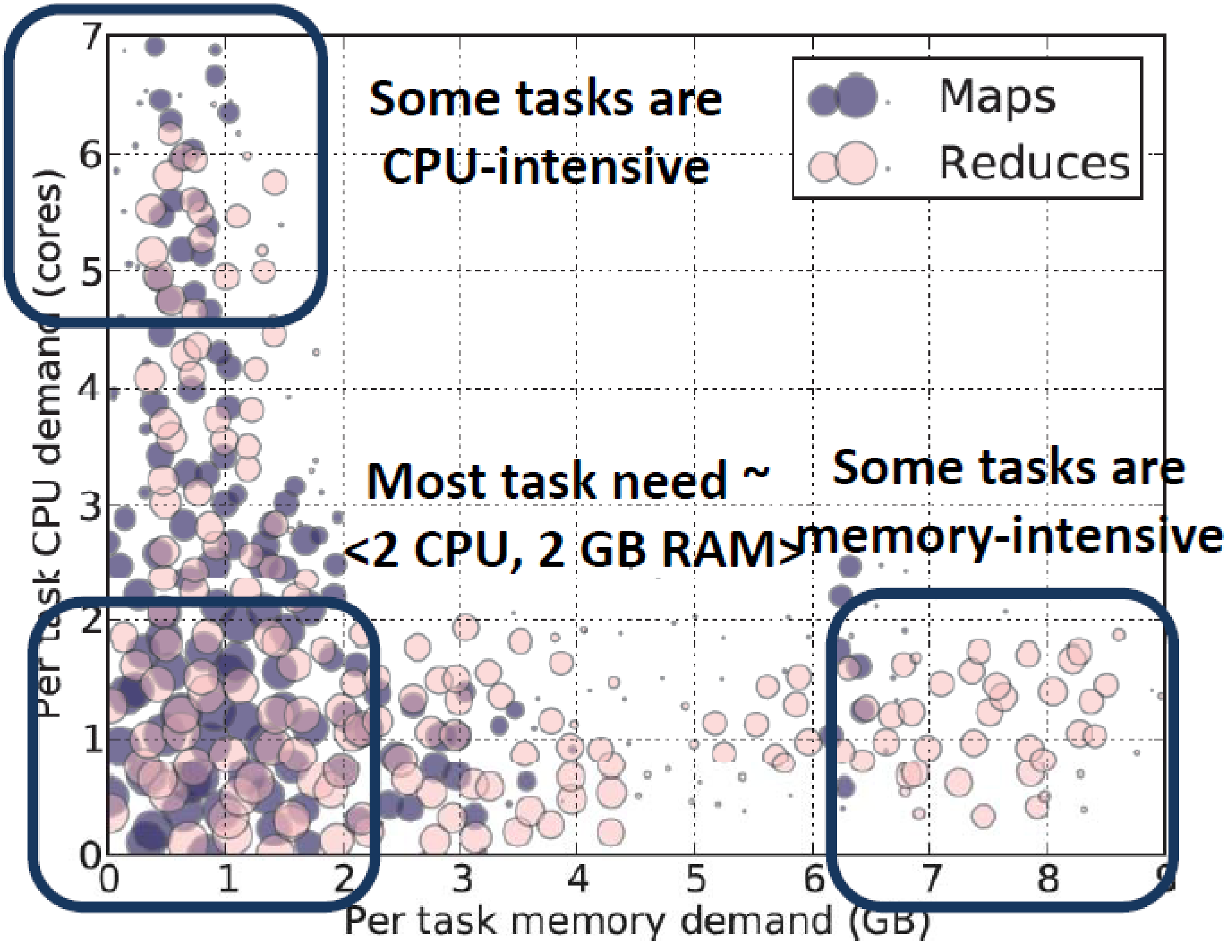
- n users want to share a resource
  - Solution:

    Allocate each 1/n of the shared resource
- Generalized by max-min fairness
  - Handles if a user wants less than its Fair share
- Generalized by weighted max-min fairness
  - Give weights to users according to importance

Why is max-min fairness not enough?

- Job scheduling in datacenters is not only about CPUs
  - Jobs consume CPU, memory, disk, and I/O
- Does this pose any challenge

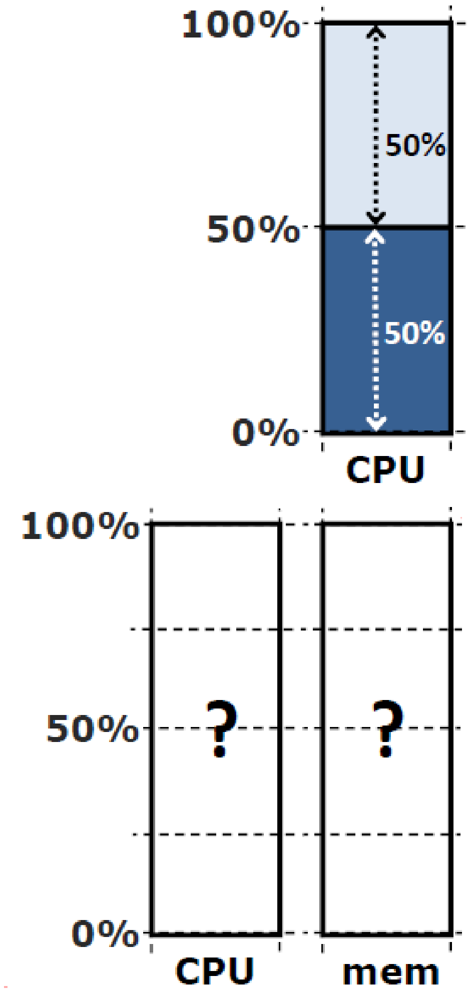
Heterogeneous Resource Demands



2000-node Hadoop Cluster at Facebook

Problem

- Single resource example
  - 1 resource: 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



Problem definition

- How to fairly share multiple resources when users have heterogenous demands on them?

Allocation Properties

- Share Guarantee
  - Every user should get 1/n of at least one resource
- Strategy proofness
  - A user should not be able to increase her allocation by lying about her demand vector
- Envy freeness
- Pareto efficiency

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
- A user's dominant share is the fraction of the dominant resource she is allocated
  - User 1's dominant share is 25%

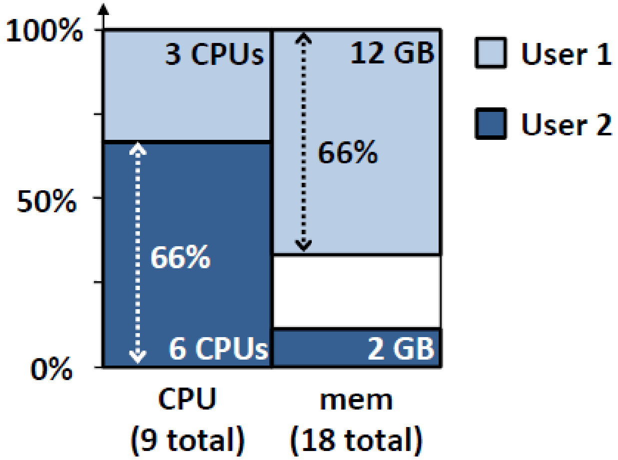
Dominant Resource Fairness(2)

- 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

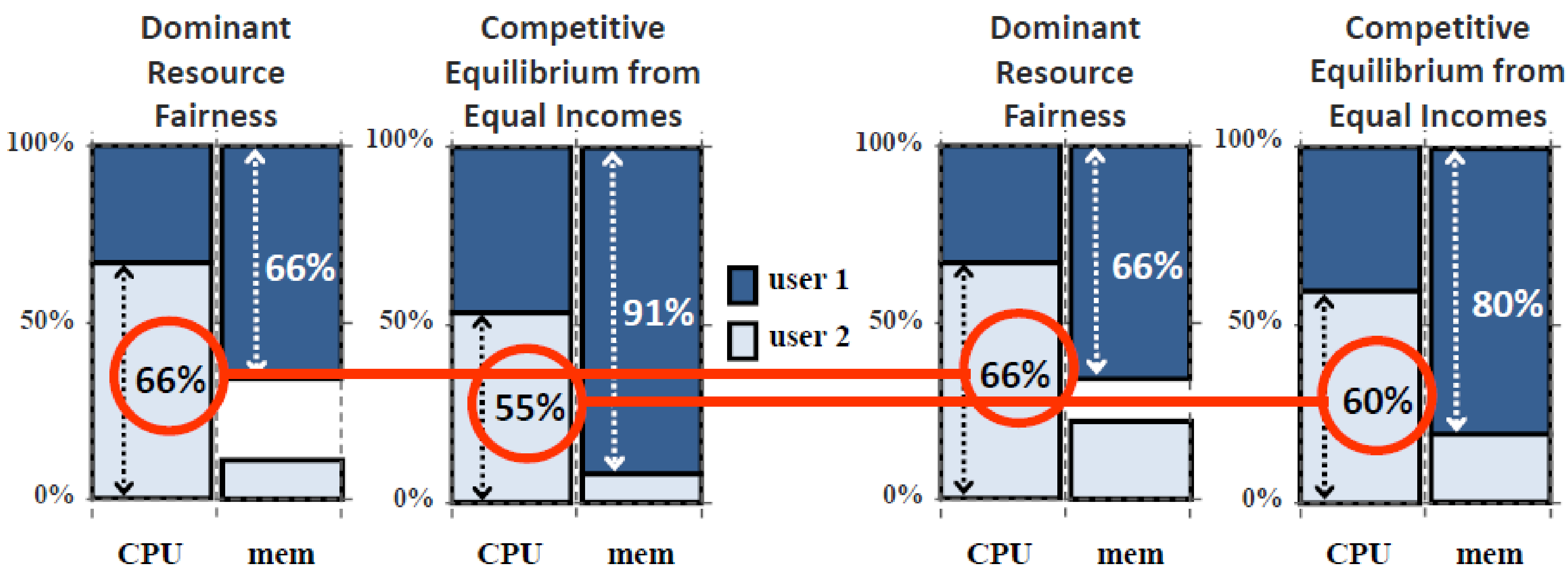
Schedule	User A		User B		CPU	RAM
	res. shares	dom. share	res. shares	dom. share	total alloc.	total alloc.
User B	$\langle 0, 0\rangle$	**0**	$\langle 3/9, 1/18\rangle$	1/3	3/9	1/18
User A	$\langle 1/9, 4/18\rangle$	**2/9**	$\langle 3/9, 1/18\rangle$	1/3	4/9	5/18
User A	$\langle 2/9, 8/18\rangle$	4/9	$\langle 3/9, 1/18\rangle$	**1/3**	5/9	9/18
User B	$\langle 2/9, 8/18\rangle$	**4/9**	$\langle 6/9, 2/18\rangle$	2/3	8/9	10/18
User A	$\langle 3/9, 12/18\rangle$	**2/3**	$\langle 6/9, 2/18\rangle$	**2/3**	1	14/18

Compare with Asset Fairness and CEEI

- Asset Fairness: Equalize each user's sum of resource shares
- CEEI: Competitive Equilibrium from Equal Incomes
  - Give each user 1/n of every resource
  - Let users trade in a perfectly competitive market

DRF vs CEEI

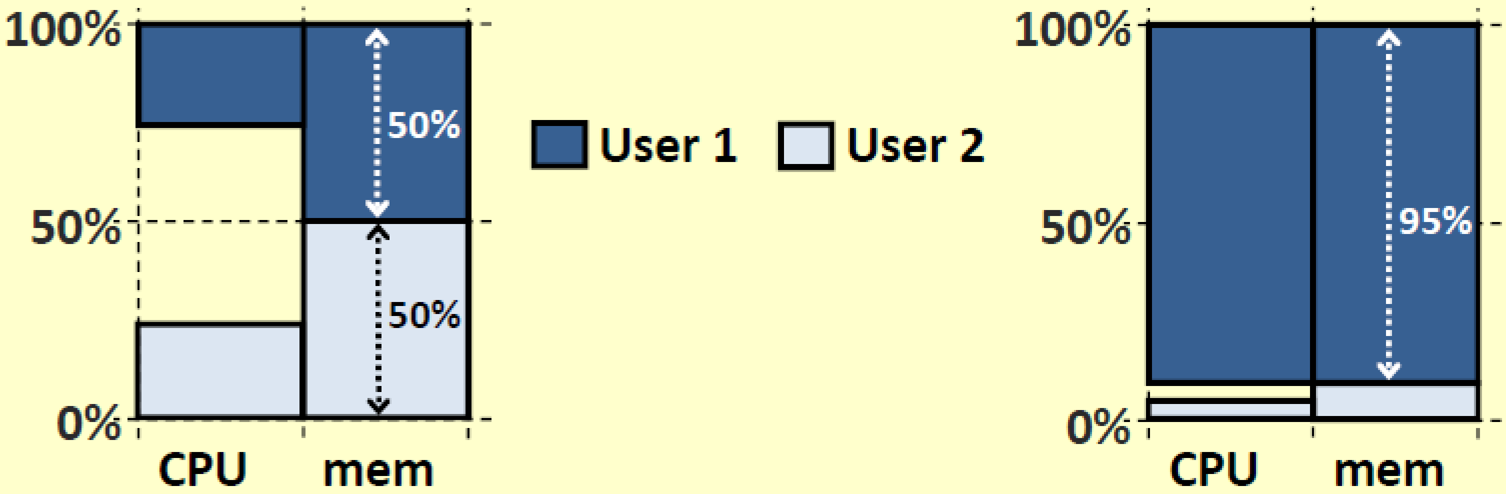
- User 1: <1 CPU, 4 GB> User 2: <3 CPU, 1 GB>
  - DRF more fair, CEEI better utilization



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

Gaming Utilization-Optimal Schedulers

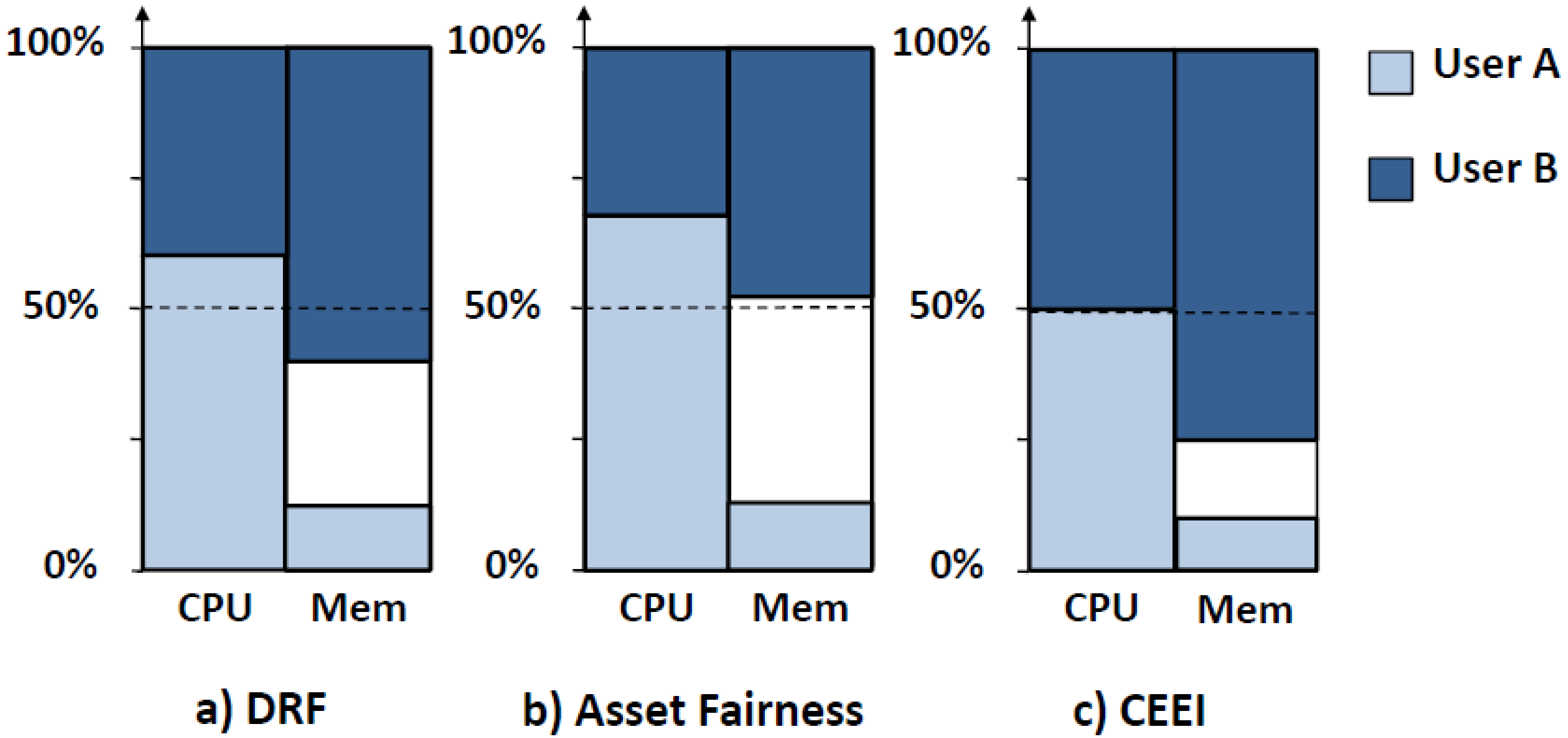
- Cluster with **<100 CPU, 100 GB>**
- 2 users, each demanding **<1 CPU, 2 GB>** per task



- User 1 lies and demands **<2 CPU, 2 GB>**
- Utilization-Optimal scheduler prefers user 1

Example of DRF vs Asset vs CEEI

- Resources <1000 CPUs, 1000 GB>
- 2 users A: <2 CPU, 3 GB> and B: <5 CPU, 1



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			

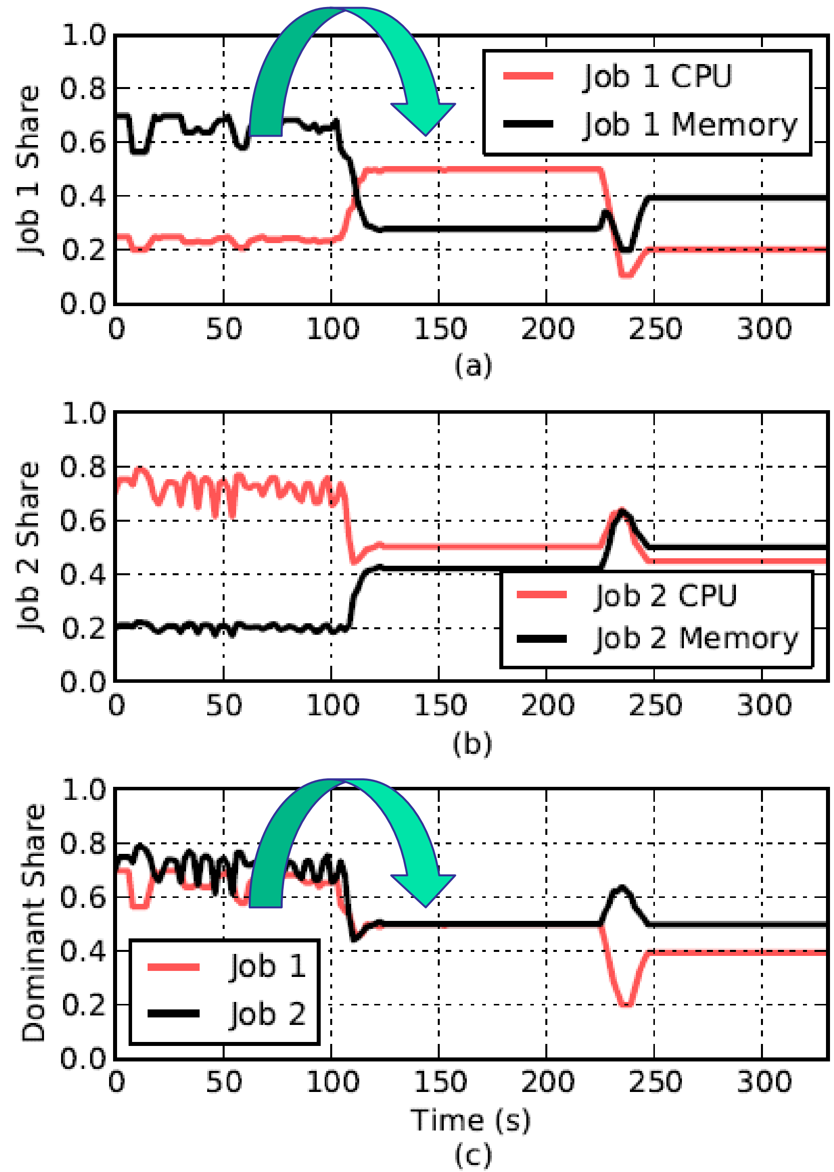
Evaluation Methodology

- Micro-experiments on EC2
  - Evaluate DRF's dynamic behavior when demands change
  - Compare DRF with current Hadoop scheduler
- Macro-benchmark through simulations
  - Simulate Facebook trace with DRF and current Hadoop scheduler

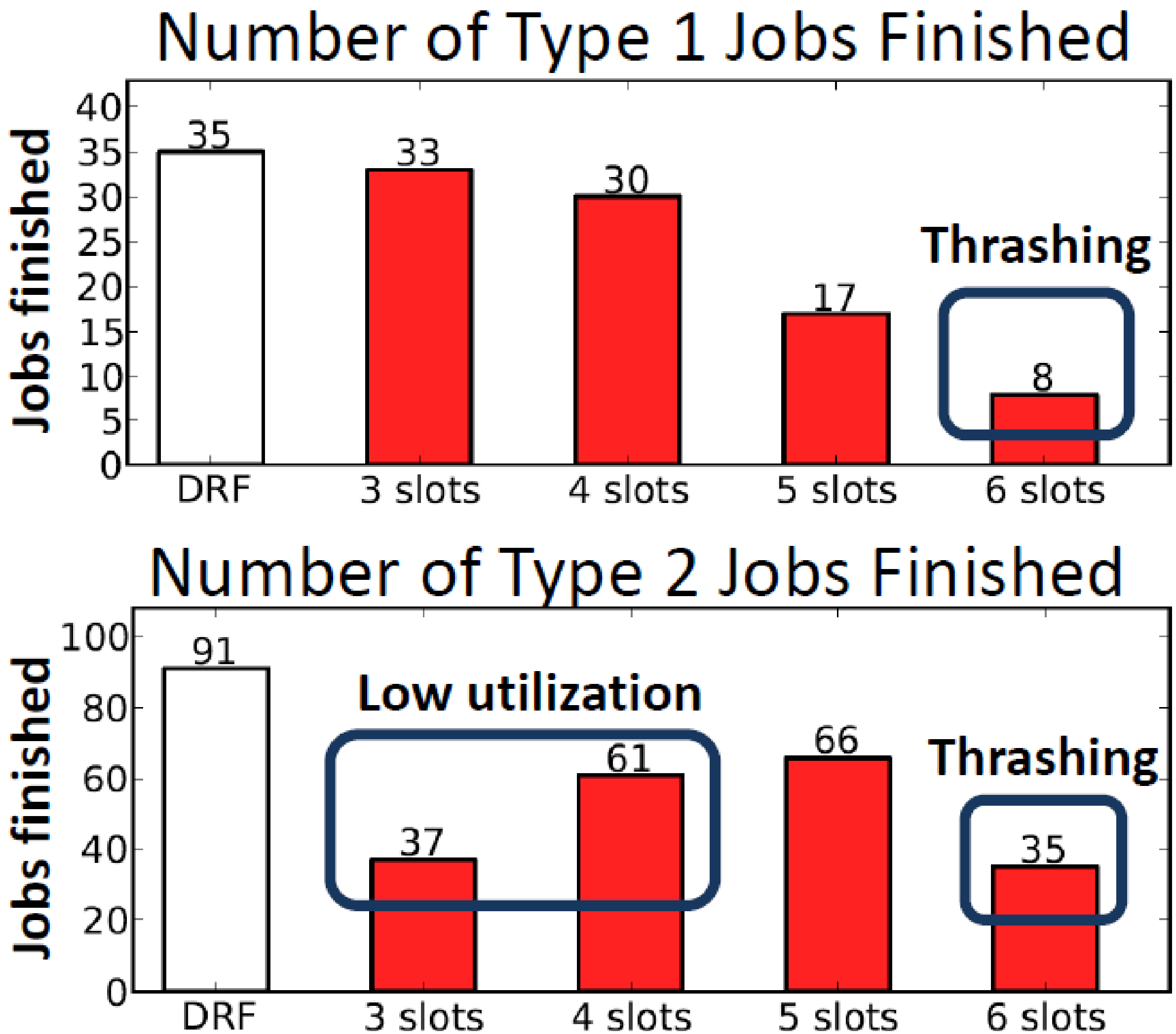
DRF inside Mesos on EC2

- Dominant shares are equalized
- Job1's dominant resource changes from Memory to CPU

Share guarantee changes from 70% to 50%

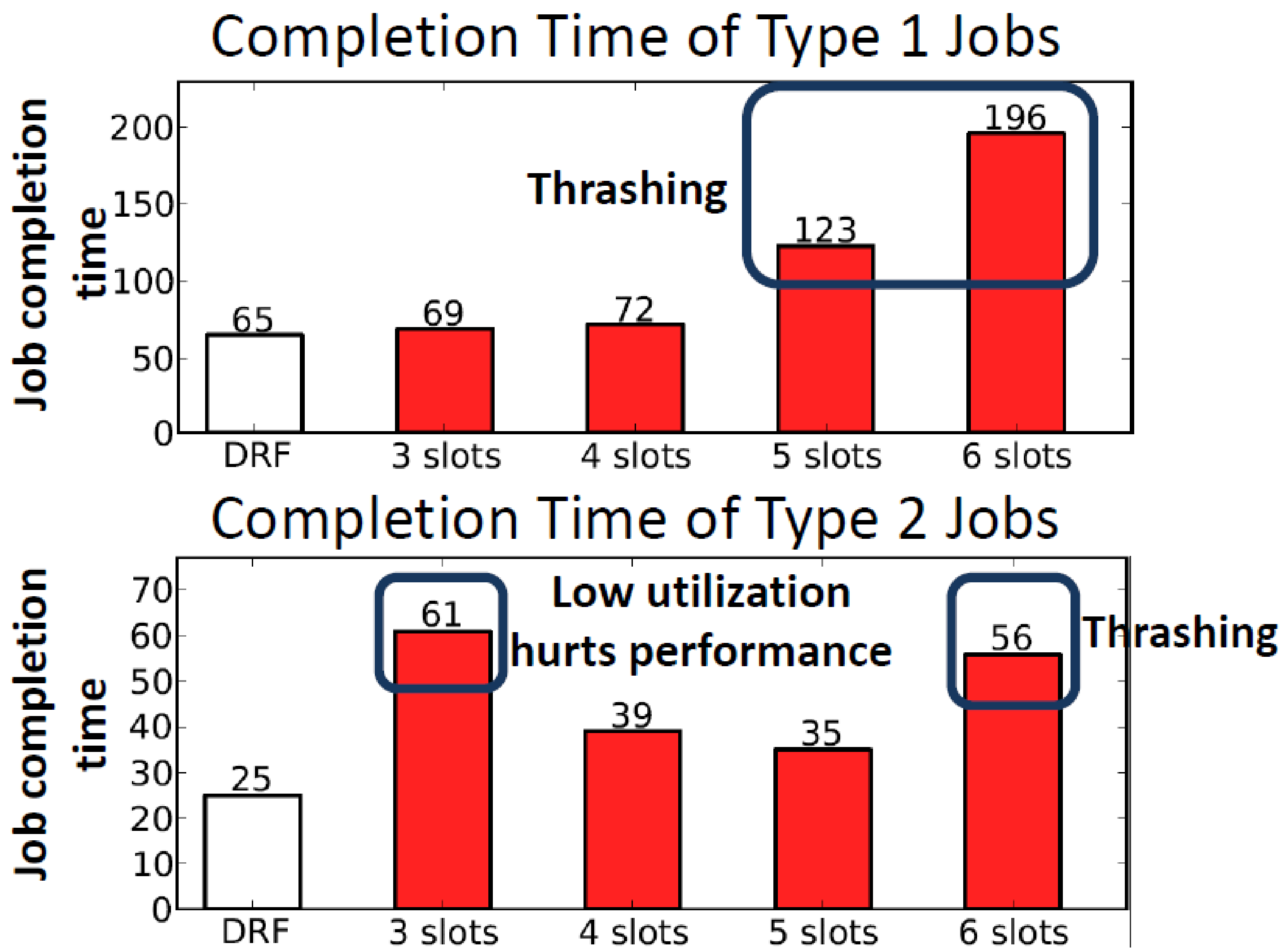


Experiment: DRF vs Slots



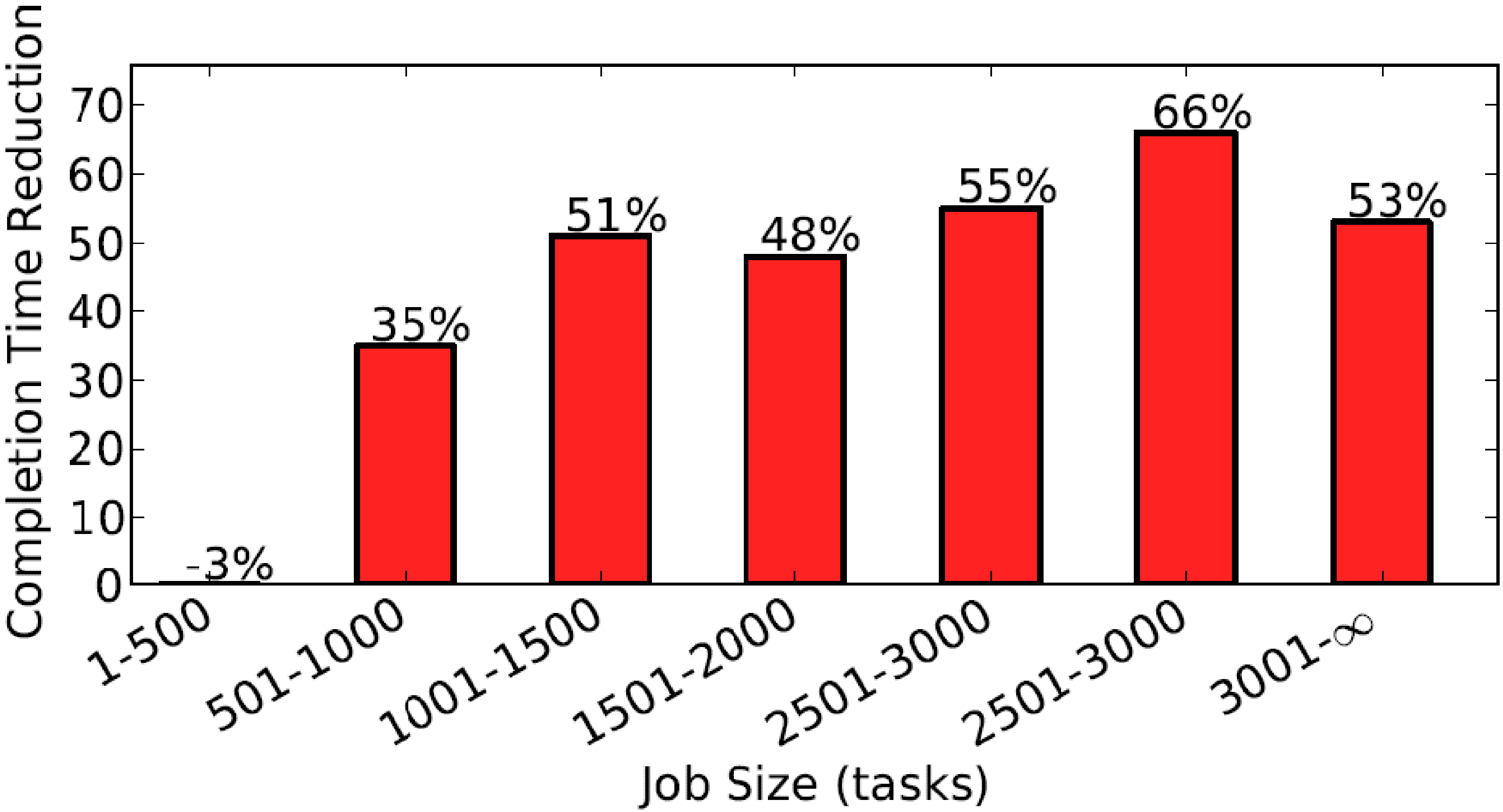
Type 1 jobs <2 CPU, 2 GB>   Type 2 jobs <1 CPU, 0.5GB>

Experiment: DRF vs Slots

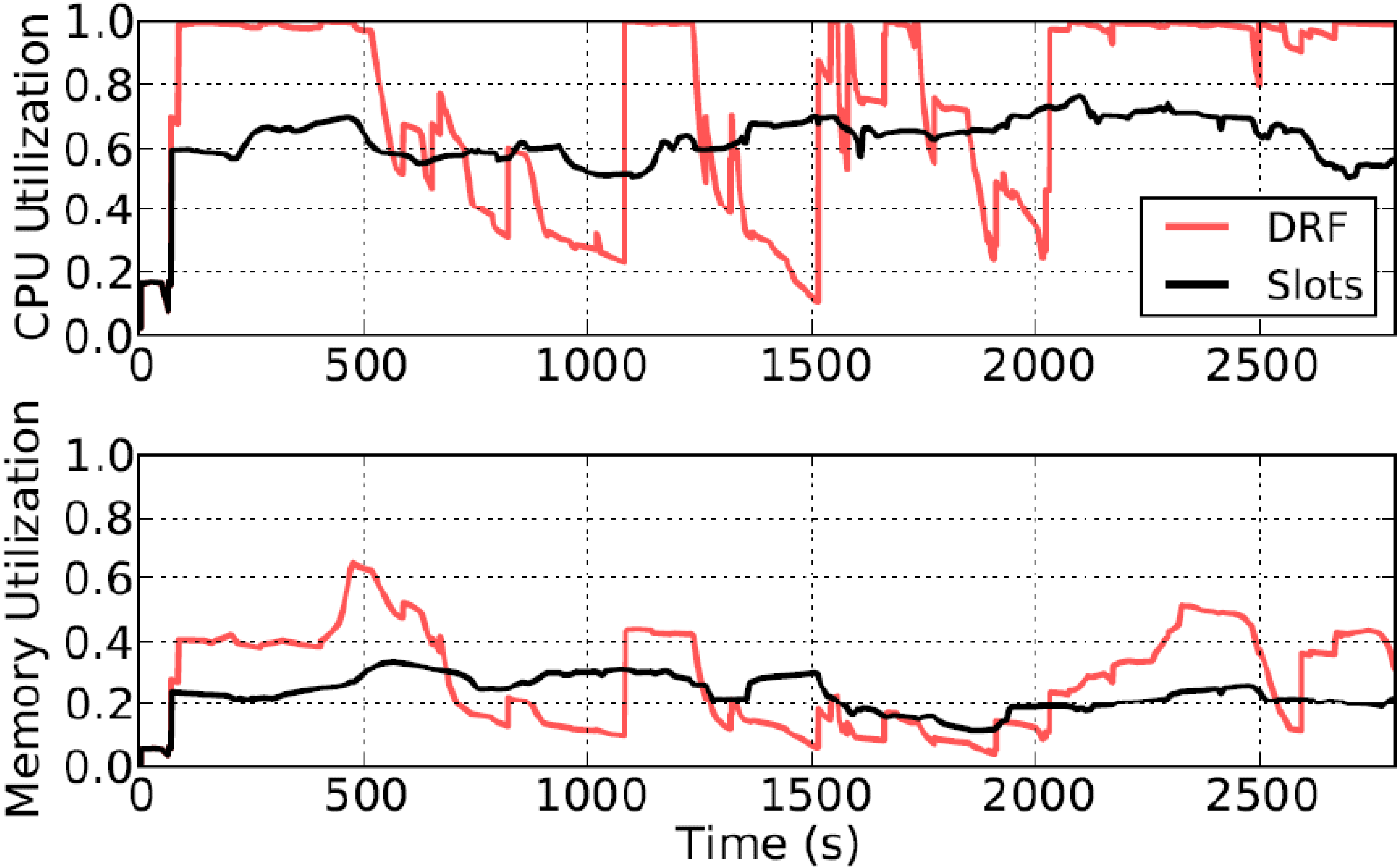


Type 1 job <2 CPU, 2 GB> Type 2 job <1 CPU, 0.5GB>

Simulations using Facebook Traces



Simulations using Facebook Traces



Conclusion

- DRF provides multiple-resource fairness in the presence of heterogenous demand
  - First generalization of max-min fairness to multiple-resources
- DRF's properties
  - Share guarantee
  - Strategy-proofness
  - Performs better than current approaches

Thanks