Dominant resource fairness: Fair allocation of heterogeneous resources in datacenters

Ali Ghodsi, Matei Zaharia, Benjamin Hindman,
Andy Konwinski, Scott Shenker, Ion Stoica

University of California, Berkeley

Presenter: Da Wang
6.897 Cloud Computing Seminar, EECS, MIT

April 1, 2011

Overview

Motivation

- Multiple resources
- Heterogeneous task demands

Fairness

- Sharing incentive
- Strategy-proofness
- Envy-freeness
- Pareto-efficiency
- More . . .

Dominant Resource Fairness

- Algorithm
- Properties

Alternatives

- Asset
- CEEI

Motivation

Heterogeneity in data centers:

- Resources
- User demand

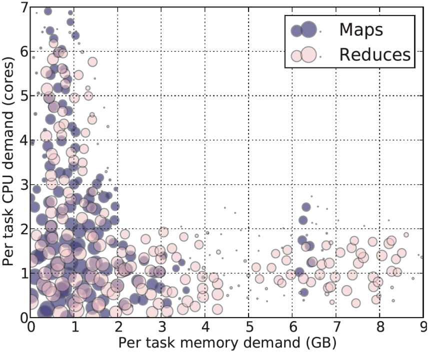
Existing schedulers:

- Quincy
- Hadoop Fair Scheduler

Ignores the user demand heterogeneity, causing

- Mismatch
- Inefficiency

CPU and Memory demands in the Facebook data center

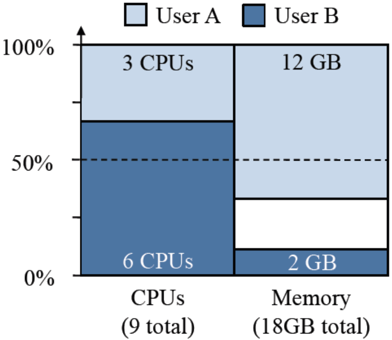


Resource Scheduling Problem

System model

- n users $u_1, u_2, \cdots, u_n$
- m resources $R_1, R_2, \cdots, R_m$
- demands matrix $[D_{ij}]_{n \times m}$
  - Each user has demand vector D_i
- allocation $A = [a_1, a_2, \cdots, a_n]$

- u_A, u_B
- R_1 = 9 CPUs
  R_2 = 18 GB Memory
- $D_A = [1, 4]$, $D_B = [3, 1]$
- $A = [3, 2]$



Simplifications

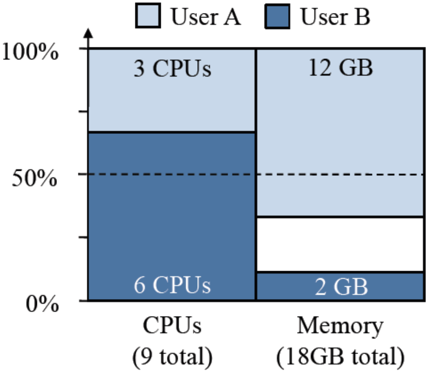
- infinite task demand
- divisible resources
- homogeneous tasks for each user

Allocation Properties

Sharing incentive

Each user should be better off sharing the cluster,
than exclusively using her own partition of the cluster.

- $D_A = [1, 4]$
  $D_B = [3, 1]$



Allocation Properties

Strategy-proofness

Users should not be able to benefit
by lying about their resource demands.

- benefit: run more tasks
- lying: fake the demand vector

Allocation Properties

Envy-freeness

A user should not prefer
the allocation of another user.

- The notion of "fairness" in economics
- prefer: runs more task
  - ⇒ Strictly more resource for each type

Allocation Properties

Pareto efficiency

It should not be possible to increase the allocation of a user without decreasing the allocation of at least another user.

- “Maximal” system utilization
- Not difficult to achieve

Allocation Properties

Additional ones

Single resource fairness

For a single resource, the solution should reduce to max-min fairness.

- Maximize the minimum share of resources.
- Infinite task demand ⇒ equal division among users.

Allocation Properties

Additional ones

Single resource fairness

Bottleneck fairness

For bottleneck resource, the solution should reduce to max-min fairness for that resource.

- Bottleneck: everyone wants the resource the most!

Allocation Properties

Additional ones

Single resource fairness

Bottleneck fairness

Population monotonicity

When a user leaves the system, none of the allocation of the remaining users should decrease.

Allocation Properties

Additional ones

Single resource fairness

Bottleneck fairness

Population monotonicity

Resource monotonicity

When more resources are added to the system, none of the allocation of existing users should decrease.

Allocation Properties

Additional ones

Single resource fairness

Bottleneck fairness

Population monotonicity ✓

Resource monotonicity ✓

Dominant Resource Fairness (DRF)

Dominant Resource

- Each user receives a share of the system resources
- The maximum among all shares of a user: dominant share
- Resource corresponding to the dominant share: dominant resource

Example

- System: 9 CPUs, 12GB RAM
- Task demand: [3 CPUs, 1GB]
- Dominant resource: CPU

The number of tasks one can run is limited by the dominant resource.

Dominant Resource Fairness (DRF)

Dominant Resource

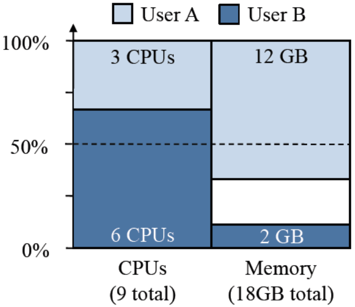
- Each user receives a share of the system resources
- The maximum among all shares of a user: dominant share
- Resource corresponding to the dominant share: dominant resource

Dominant Resource Fairness

- Maximize the smallest dominant share in the system
- Achieve all four main fairness properties

Example

- System: 9 CPUs, 12GB RAM
- Task demand: [3 CPUs, 1GB]
- Dominant resource: CPU



The number of tasks one can run is limited by the dominant resource.

Dominant Resource Fairness

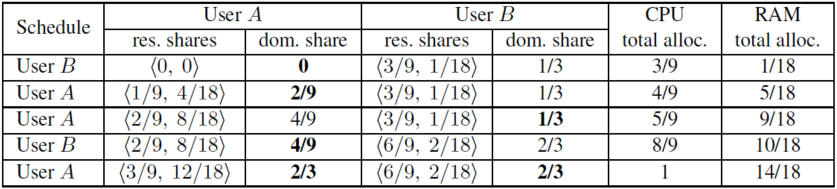
Algorithm

Greedy algorithm

Repeatedly allocation one task to the user

- with the minimum dominant share
- and when there are enough resources to allocate another task

- System: 9 CPUs, 18GB RAM
- $D_A = [1, 4]$, $D_B = [3, 1]$
- $A = [3, 2]$



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

Dominant Resource Fairness

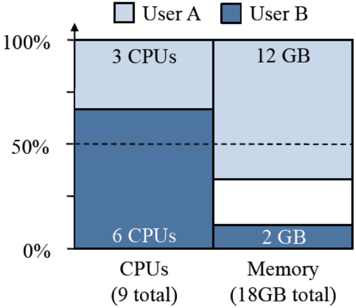
Algorithm

Greedy algorithm

Repeatedly allocation one task to the user

- with the minimum dominant share
- and when there are enough resources to allocate another task

- System: 9 CPUs, 18GB RAM
- $D_A = [1, 4]$, $D_B = [3, 1]$
- $A = [3, 2]$



Dominant Resource Fairness

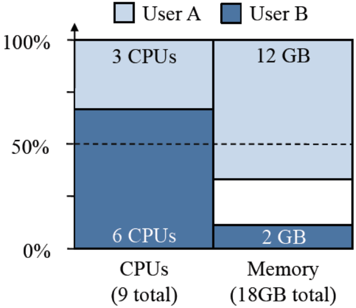
Algorithm

Progressive filling

When resources can be allocated in arbitrary small amounts,

- increase all users' dominant shares at the same rate
- increase other resources shares proportionally
- until at least one resource is saturated

- System: 9 CPUs, 18GB RAM
- $D_A = [1, 4]$, $D_B = [3, 1]$
- $A = [3, 2]$



Dominant Resource Fairness

Properties

- Sharing incentives

Can be proven based on its allocation algorithm.
Each user gets at least $1/n$ dominant resources.

Dominant Resource Fairness

Properties

- ✓ Sharing incentives
- Strategy-proofness

Lying about
dominant resource demand
v.s. other demands.

Dominant Resource Fairness

Properties

- ✓ Sharing incentives
- ✓ Strategy-proofness
- Envy-freeness

No user can get more dominant resource than other users.

Dominant Resource Fairness

Properties

- ✓ Sharing incentives
- ✓ Strategy-proofness
- ✓ Envy-freeness
- ■ Pareto efficiency

Resources utilization cannot be increased.

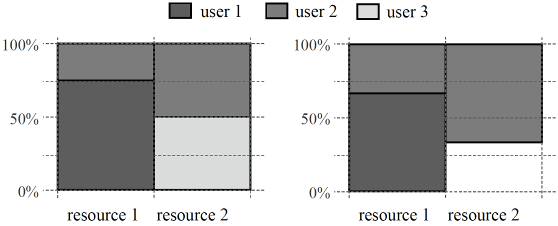
Dominant Resource Fairness

Properties

- ✓ Sharing incentives
- ✓ Strategy-proofness
- ✓ Envy-freeness
- ✓ Pareto efficiency
- Population Monotonicity

Given strictly positive demand vectors, population monotonicity is satisfied. Otherwise, it may be violated.

- $D_1 = [2, 0]$, $D_2 = [1, 2]$, $D_3 = [0, 2]$
- 24 units of each resource
- Then 3rd user leaves.
- Allocation: $[9, 6, 6] \Rightarrow [8, 8]$



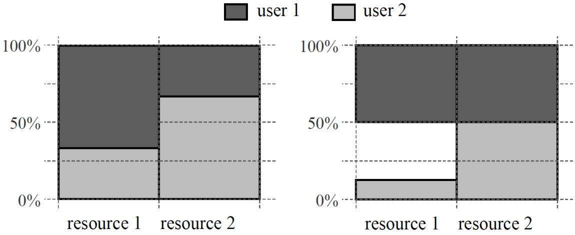
Dominant Resource Fairness

Properties

- ✓ Sharing incentives
- ✓ Strategy-proofness
- ✓ Envy-freeness
- ✓ Pareto efficiency
- ✓ Population Monotonicity
- Resource Monotonicity

DRF does not satisfies resource monotonicity.

- $D_1 = [2, 1]$, $D_2 = [1, 2]$
- Initially: 12 units of each resource
- Then first resource increase to 24
- Allocation: $[4, 4] \Rightarrow [6, 3]$



Dominant Resource Fairness

Properties

- ✓ Sharing incentives
- ✓ Strategy-proofness
- ✓ Envy-freeness
- ✓ Pareto efficiency
- ✓ Population Monotonicity
- ✗ Resource Monotonicity

Dominant Resource Fairness

Weighted case

Weights

- W_{ij}: the weight of user i for resource j
- Weighted dominant share:

$$\max_j \frac{\text{user } i's \text{ share of resource } j}{W_{ij}}$$

- Algorithm essentially the same.

Models

- User priority over resources
- User tasks with different demand vectors

Alternative Allocation Policies

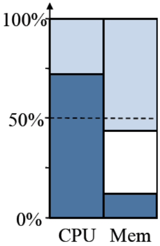
Asset Fairness

Idea Equal shares of differenct resources are worth the same.

Aim To equalize the aggregate share allocated to each user.

Example

- R_1 = 9 CPUs
  R_2 = 18 GB Memory
- $D_1 = [1, 4]$,
  $D_2 = [3, 1]$
- $A = [2.52, 2.16]$



Alternative Allocation Policies

Asset Fairness

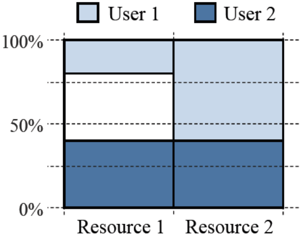
Idea Equal shares of differenct resources are worth the same.

Aim To equalize the aggregate share allocated to each user.

Weakness Fail to meet the sharing incentive property.

Example

- R_1 = 30 CPUs
  R_2 = 30 GB Memory
- $D_1 = [1, 3]$,
  $D_2 = [1, 1]$
- $A = [6, 12]$



Alternative Allocation Policies

Competitive Equilibrium from Equal Incomes

Idea Allocate resources in a perfectly competitive market.

Aim To maximize user utility (e.g., dominant share)

Weakness Fail to meet the strategy-proofness property
Fail to meet the population monotonicity property

Allocation Policies

Comparisons

Property	Asset	CEEI	DRF
Sharing incentives		✓	✓
Strategy-proofness	✓		✓
Envy-freeness	✓	✓	✓
Pareto efficiency	✓	✓	✓
Single Resource Fairness	✓	✓	✓
Bottleneck Fairness		✓	✓
Population Monotonicity	✓		✓
Resource Monotonicity			

Allocation Policies

Tradeoffs

Inevitable trade-off between

- resource monotonicity
- sharing incentive
- Pareto efficiency

No allocation policy can satisfy all three properties at the same time!

Allocation Policies

Tradeoffs

Inevitable trade-off between

- resource monotonicity
- sharing incentive
- Pareto efficiency

No allocation policy can satisfy all three properties at the same time!

Proof by example:

- Two users A and B
- Two resources with equal amount
- A demand: [2,1]
  B demand: [1,2]

Allocation Policies

Tradeoffs

Inevitable trade-off between

- resource monotonicity
- sharing incentive
- Pareto efficiency

Proof by example:

- Two users A and B
- Two resources with equal amount
- A demand: [2,1]
  B demand: [1,2]

Sharing incentive

- A gets at least $1/2$ of resource 1
- B gets at least $1/2$ of resource 2

Pareto efficiency

- $\Rightarrow$ at least one of two users hold more than half of a resource
- WLOG, assume it is A holding more than half of resource 1.

Allocation Policies

Tradeoffs

Inevitable trade-off between

- resource monotonicity
- sharing incentive
- Pareto efficiency

Proof by example:

- Two users A and B
- Two resources with equal amount
- A demand: [2,1]
  B demand: [1,2]

Sharing incentive

- A gets at least $1/2$ of resource 1
- B gets at least $1/2$ of resource 2

Pareto efficiency

- $\Rightarrow$ at least one of two users hold more than half of a resource
- WLOG, assume it is A holding more than half of resource 1.

Quadruple resource 2

- $\Rightarrow$ User B now has less than $1/2$ of resource 2

Allocation Policies

Tradeoffs

Inevitable trade-off between

- resource monotonicity
- sharing incentive
- Pareto efficiency

Proof by example:

- Two users A and B
- Two resources with equal amount
- A demand: [2,1]
  B demand: [1,2]

Sharing incentive

- A gets at least $1/2$ of resource 1
- B gets at least $1/2$ of resource 2

Pareto efficiency

- $\Rightarrow$ at least one of two users hold more than half of a resource
- WLOG, assume it is A holding more than half of resource 1.

Quadruple resource 2

- $\Rightarrow$ User B now has less than $1/2$ of resource 2

Sharing incentive

- $\Rightarrow$ Give both user $1/2$ of resource 1
- $\Rightarrow$ A lost some resource 1!

Practical Issues

Indivisible Tasks & Resources

Continuous scenario

- Resources can be allocated in arbitrarily small amounts.
- Often not the case in practice

Discrete scenario

- Resources are allocated to tasks in discrete amounts
- Cluster: often consists of many small machines.

Relationships

In the discrete scenario, it is possible to allocate resources such that

- the difference between the allocation for any two users is bounded

compared to the continuous scenario.

Performance

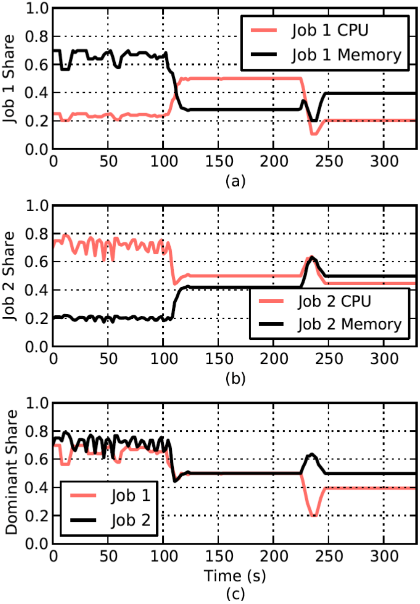
Resource share over time

Resource:
[CPU = 4, RAM = 15GB]

Time	Job 1	Job 2
1—120	[1, 10]	[1,1]
121—240	[2, 4]	[1,3]
241—360	[1,7]	[1,4]

Comments

- Adaptive to task demand change
- Sharing incentive
- Resource fragmentation in the 3rd period

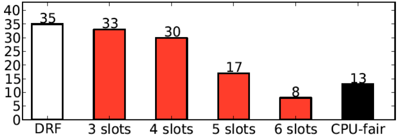


Performance

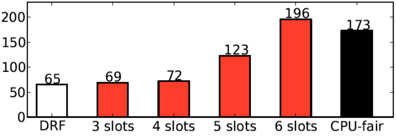
Large v.s. small jobs

DRF Comparison

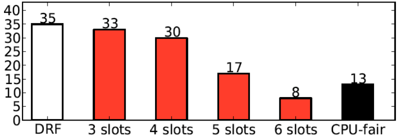
- Slot-level fair sharing: Hadoop Fair Scheduler & Quincy
- CPU-only fair sharing: single-resource scheduling



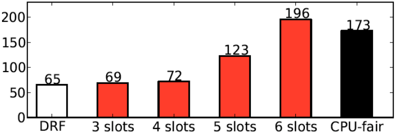
Large jobs: # completion



Large jobs: response time



Small jobs: # completion

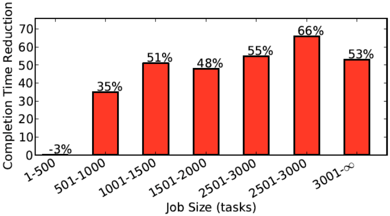


Small jobs: response time

Performance

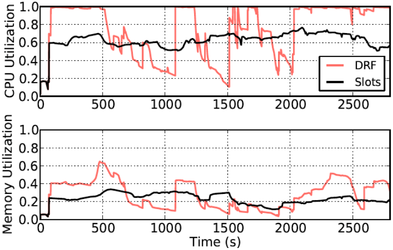
Data from Facebook cluster

Time reduction



- Small jobs are hard to improve (single phase execution)
- Large reduction on long jobs (multiple phase execution)

Utilization



- Higher utilization by adapting resource allocation with task demands

Comments

Insight

Number of tasks that can be run is determined by the dominant resource.

- Is this always the case?
- The definitions of both strategy-proofness and envyness are based on this

Thoughts

- Smoother trade-off between fairness properties?
  - Mixing different allocation strategies for better trade-offs (currently either have or do not have)
- Use the statistical properties the jobs to further optimize the efficiency
  - a probabilistic system model

Summary

Main contributions

- Use dominant share as a proxy for utility
  - The number of tasks is limited by the amount of dominant resource.
- Propose fairness properties and show that DRF satisfies most of them

Future work

- Minimize resource fragmentation (bin packing) under fairness constraints
- Allocation under placement constraints
- Use DRF as operating system scheduler for multicore systems