Dominant Resource Fairness: Fair Allocation of Multiple Resource Types

Fanchao Meng

University of Delaware

fcmeng@cis.udel.edu

October 28, 2015

Overview

What is the Problem?

What Hadoop and Dryad are doing ...

Allocate resources at the level of **fixed-size partitions** of the nodes, called *slots*, i.e. **single resource type**.

What is the Problem?

What Hadoop and Dryad are doing ...

Allocate resources at the level of **fixed-size partitions** of the nodes, called *slots*, i.e. **single resource type**.

It is bad, because ...

Different jobs in these clusters can have widely **different demands** for CPU, memory, and IO resources.

What is the Problem? (Cont.)

The problem facing in this study is ...

Fair allocation of **multiple types of resources** to users with heterogeneous demands.

What Is Called “Fair”?

The challenge is ...

How to **evaluate the fairness of allocation** in the multiple-type-of-resource-with-heterogeneous-user-demand environment?

What Is Called "Fair"? (Cont.)

Four Properties as "Must-have" Criteria

Sharing Incentive: Sharing should be better than monopolizing.

What Is Called “Fair”? (Cont.)

Four Properties as "Must-have" Criteria

Sharing Incentive: Sharing should be better than monopolizing.

Consider a cluster with identical nodes and n users. Then a user should not be able to allocate more tasks in a cluster partition consisting of $\frac{1}{n}$ of all resources.

What Is Called “Fair”? (Cont.)

Four Properties as "Must-have" Criteria

Sharing Incentive: Sharing should be better than monopolizing.

Consider a cluster with identical nodes and n users. Then a user should not be able to allocate more tasks in a cluster partition consisting of $\frac{1}{n}$ of all resources.

It also means the **performance isolation**, since it guarantees a minimum allocation to each user.

What Is Called "Fair"? (Cont.)

Four Properties as "Must-have" Criteria (Cont.)

Sharing Incentive: Sharing should be better than monopolizing.

Strategy-proofness: No one can get more resources by lying about demands.

What Is Called “Fair”? (Cont.)

Four Properties as ”Must-have” Criteria (Cont.)

Sharing Incentive: Sharing should be better than monopolizing.

Strategy-proofness: No one can get more resources by lying about demands.

Avoid cases like this ...

Provider: “I’ll give you dedicated machines as long as you could guarantee high utilization.”
User: “Deal! I’ll give you my infinite loop program.”

What Is Called “Fair”? (Cont.)

Four Properties as ”Must-have” Criteria (Cont.)

Sharing Incentive: Sharing should be better than monopolizing.

Strategy-proofness: No one can get more resources by lying about demands

Envy-freeness: No one would want to trade her allocation with that of others.

What Is Called “Fair”? (Cont.)

Four Properties as ”Must-have” Criteria (Cont.)

Sharing Incentive: Sharing should be better than monopolizing.

Strategy-proofness: No one can get more resources by lying about demands

Envy-freeness: No one would want to trade her allocation with that of others.

It means ...

Exactly what we call the fairness!

What Is Called “Fair”? (Cont.)

Four Properties as "Must-have" Criteria (Cont.)

Sharing Incentive: Sharing should be better than monopolizing.

Strategy-proofness: No one can get more resources by lying about demands

Envy-freeness: No one would want to trade her allocation with that of others.

Pareto Efficiency: Impossible to make one better off without making at least one worst off.

What Is Called "Fair"? (Cont.)

Four Properties as "Must-have" Criteria (Cont.)

Sharing Incentive: Sharing should be better than monopolizing.

Strategy-proofness: No one can get more resources by lying about demands

Envy-freeness: No one would want to trade her allocation with that of others.

Pareto Efficiency: Impossible to make one better off without making at least one worst off.

It means ...

To maximize system utilization subject to satisfying the other properties.

What Is Called "Fair"? (Cont.)

Four Properties as "Nice-to-have" Criteria

Single Resource Fairness: For a single resource, the solution should reduce to **max-min fairness**.

What Is Called "Fair"? (Cont.)

Four Properties as "Nice-to-have" Criteria

Single Resource Fairness: For a single resource, the solution should reduce to **max-min fairness**.

Bottleneck Fairness: If there is one resource that is percent-wise demanded most of by every user, then the solution should reduce to max-min fairness for that resource.

What Is Called "Fair"? (Cont.)

Four Properties as "Nice-to-have" Criteria

Single Resource Fairness: For a single resource, the solution should reduce to **max-min fairness**.

Bottleneck Fairness: If there is one resource that is percent-wise demanded most of by every user, then the solution should reduce to max-min fairness for that resource.

Population Monotonicity: When a user leaves the system and relinquishes her resources, none of the allocations of the remaining users should decrease.

What Is Called “Fair”? (Cont.)

Four Properties as ”Nice-to-have” Criteria

Single Resource Fairness: For a single resource, the solution should reduce to **max-min fairness**.

Bottleneck Fairness: If there is one resource that is percent-wise demanded most of by every user, then the solution should reduce to max-min fairness for that resource.

Population Monotonicity: When a user leaves the system and relinquishes her resources, none of the allocations of the remaining users should decrease.

Resource Monotonicity: If more resources are added to the system, none of the allocations of the existing users should decrease.

How To Address The Problem?

A new allocation policy: **Dominant Resource Fairness (DRF)**!

What Is Min-Max Fairness?

Min-Max Fairness

- It **maximizes the minimum allocation** received by a user in the system.
- Assuming each user has enough demand, it gives each user an **equal share** of the resources.

What Is Min-Max Fairness? (Cont.)

Weighted Min-Max Fairness

- It can take **other concerns (used to weight)** into account such as priority, reservation, and deadline.
- It **ensures isolation**, i.e. a user's share is guaranteed in a certain amount regardless others' demands.

DRF Is a Generalization of Min-Max Fairness

DRF is a generalized version min-max fairness to multiple resource types, including unweighted DRF and weighted DRF.

General Idea of DRF

General Idea

- For each user, compute the share (roughly means percentage) of each resource allocated to her. W.r.t. all her shares, the maximum one is called her **dominant share**, the corresponding resource is called the **dominant resource**.
- Apply max-min fairness to maximize the minimum dominant share across all users, i.e. maximize the smallest dominant share in the system, then the second-smallest, and so on.

Example

Given:

System: $\langle 9 CPUs, 18 GB\ RAM \rangle$.

User A: Demands per task $= \langle 1 CPU, 4 GB\ RAM \rangle$.

User B: Demands per task $= \langle 3 CPUs, 1 GB\ RAM \rangle$.

Seek:

Optimal resource allocation.

Example (Cont.)

Analysis:

User A: Consumption per task $= \langle \frac{1}{9} CPUs, \frac{2}{9} GB \rangle$.

User B: Consumption per task $= \langle \frac{1}{3} CPUs, \frac{1}{18} GB \rangle$.

Example (Cont.)

Analysis:

User A: Consumption per task $= \langle \frac{1}{9} CPUs, \frac{2}{9} GB \rangle$.

User B: Consumption per task $= \langle \frac{1}{3} CPUs, \frac{1}{18} GB \rangle$.

This step is not trivial ...

Because otherwise e.g. $1 CPU$ is even not in the same metric space as $4 GB\ RAM$ is.

Example (Cont.)

Analysis:

User A: Consumption per task $= \langle \frac{1}{9} CPUs, \frac{2}{9} GB \rangle$.
User B: Consumption per task $= \langle \frac{1}{3} CPUs, \frac{1}{18} GB \rangle$.

Find the dominant share for each user ...

User A: Memory.
User B: CPU.

Example (Cont.)

Analysis:

User A: Consumption per task $= \langle \frac{1}{9} CPUs, \frac{2}{9} GB \rangle$; Dominant share = Memory.

User B: Consumption per task $= \langle \frac{1}{3} CPUs, \frac{1}{18} GB \rangle$; Dominant share = CPU.

TODO

- Maximize the minimum dominant share across all users.
- Equalize dominant shares.

Example (Cont.)

Problem Formulation:

Let x and y be the number of tasks allocated by DRF to users A and B, respectively. Then user A receives $\langle x\mathit{CPU}, 4x\mathit{GB}\rangle$, and user B receives $\langle 3y\mathit{CPU}, y\mathit{GB}\rangle$.

The optimization problem:

max(x, y)	(Maximize allocations)
subject to	
$x + 3y \leq 9$	(CPU constraint)
$4x + y \leq 18$	(Memory constraint)
$\frac{2x}{9} = \frac{y}{3}$	(Equalize dominant shares)

Example (Cont.)

Solution:

$x = 3$ and $y = 2$, i.e. user A gets $\langle 3CPU, 12GB \rangle$, and user B gets $\langle 6CPU, 2GB \rangle$.

Example (Cont.)

Solution:

$x = 3$ and $y = 2$, i.e. user A gets $\langle 3CPU, 12GB \rangle$, and user B gets $\langle 6CPU, 2GB \rangle$.

NOTE 1

Equalization is not always mandatory, since when a user's demand is met, the user will not need more tasks, then the excess resources will be split among the other users.

Example (Cont.)

Solution:

$x = 3$ and $y = 2$, i.e. user A gets $\langle 3CPU, 12GB \rangle$, and user B gets $\langle 6CPU, 2GB \rangle$.

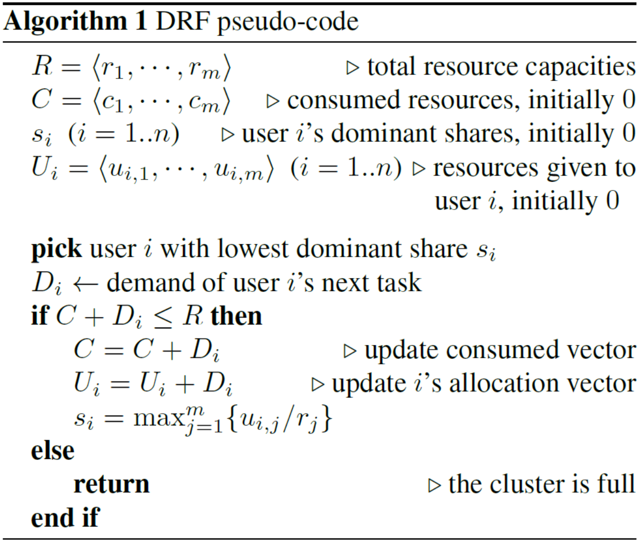
NOTE 1

Equalization is not always mandatory, since when a user's demand is met, the user will not need more tasks, then the excess resources will be split among the other users.

NOTE 2

If a resource has been exhausted, then the sharing will break down.

DRF Scheduling Algorithm



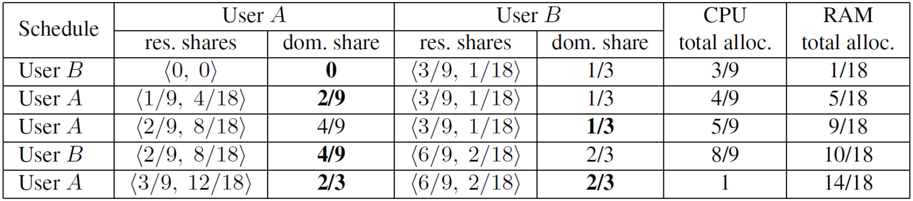
Algorithm 1 DRF pseudo-code

```
$R = \langle r_1, \cdots, r_m \rangle$    ▷ total resource capacities
$C = \langle c_1, \cdots, c_m \rangle$    ▷ consumed resources, initially 0
$s_i \ \ (i = 1..n)$    ▷ user $i$'s dominant shares, initially 0
$U_i = \langle u_{i,1}, \cdots, u_{i,m} \rangle \ (i = 1..n)$    ▷ resources given to user $i$, initially 0

pick user $i$ with lowest dominant share $s_i$
$D_i \leftarrow$ demand of user $i$'s next task
if $C + D_i \leq R$ then
    $C = C + D_i$    ▷ update consumed vector
    $U_i = U_i + D_i$    ▷ update $i$'s allocation vector
    $s_i = \max_{j=1}^{m} \{u_{i,j}/r_j\}$
else
    return    ▷ the cluster is full
end if
```

DRF Scheduling Algorithm (Cont.)

Explanation



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

Figure 1: DRF scheduling step by step.

DRF Scheduling Algorithm (Cont.)

Explanation

In a word, ...

At each step, the user with the lowest dominant share will be scheduled.

DRF Scheduling Algorithm (Cont.)

Explanation

In a word, ...

At each step, the user with the lowest dominant share will be scheduled.

Time Complexity

Binary heap: $O(logn)$
Any better way?

Weighted DRF

The reason we need it is ...

We may want to allocate more resources to users running more important jobs, or to users that have contributed more resources to the cluster.

Weighted DRF

The reason we need it is ...

We may want to allocate more resources to users running more important jobs, or to users that have contributed more resources to the cluster.

Add weights to the algorithm ...

Each user i is associated a weight vector $W_i = \langle w_{i,1}, \ldots, w_{i,m} \rangle$, where $w_{i,j}$ represents the weight of user i for resource j.
Modify the algorithm: $s_i = \max_j \{u_{i,j}/w_{i,j}\}$

Alternative Fair Allocation Policies

Asset Fairness Equalizes the aggregate resource value allocated to each user.

Alternative Fair Allocation Policies

Asset Fairness Equalizes the aggregate resource value allocated to each user.

It means ...

1. $x_i = \sum_j s_{i,j}$, where $s_{i,j}$ is the share of resource j to user i.
2. At each step, the user with the lowest **aggregate share** will be scheduled.

Alternative Fair Allocation Policies

Asset Fairness Equalizes the aggregate resource value allocated to each user.

Competitive Equilibrium from Equal Incomes (CEEI) The allocation is given by the Nash bargaining solution.

Alternative Fair Allocation Policies

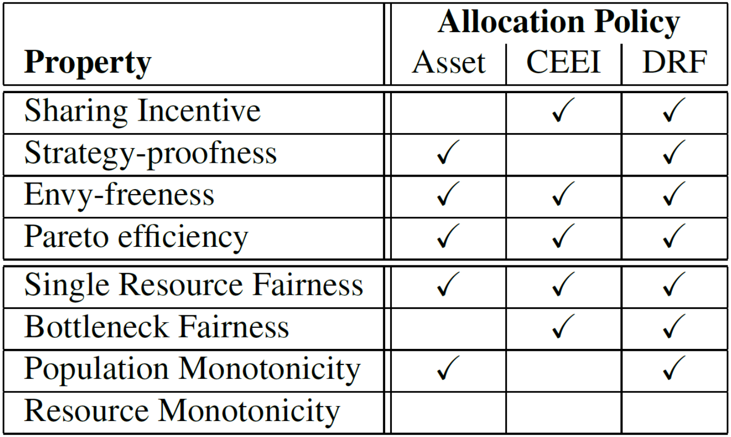
Asset Fairness Equalizes the aggregate resource value allocated to each user.

Competitive Equilibrium from Equal Incomes (CEEI) The allocation is given by the Nash bargaining solution.

It means ...

CEEI picks the feasible allocation that maximizes $\prod_i s_i$.

Comparison Between DRF and Others



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

Figure 2: Properties of Asset Fairness, CEEI and DRF.

Dynamic Resource Sharing

Experimental Settings

- 2 jobs on a 48-node Mesos cluster on Amazon EC2.
- "XL" instances with 4 CPU cores and 15 GB of RAM.
- Mesos is configured to allocate up to 4 CPUs and 14 GB (1 GB for OS) of RAM on each node.
- $0 \sim 2$ minutes: Job 1 $= \langle 1CPU, 10GB \rangle$ per task; Job 2 $= \langle 1CPU, 1GB \rangle$ per task.
- $2 \sim 4$ minutes: Job 1 $= \langle 2CPU, 4GB \rangle$ per task; Job 2 $= \langle 1CPU, 3GB \rangle$ per task.
- $4 \sim 6$ minutes: Job 1 $= \langle 2CPU, 7GB \rangle$ per task; Job 2 $= \langle 1CPU, 4GB \rangle$ per task.

Dynamic Resource Sharing (Cont.)

Experimental Results

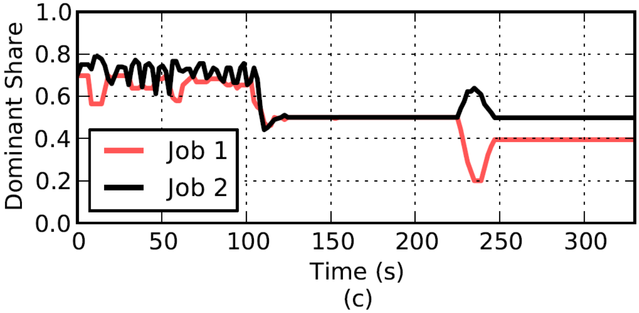


Figure 3: Dominant Shares

Dynamic Resource Sharing (Cont.)

Experimental Results

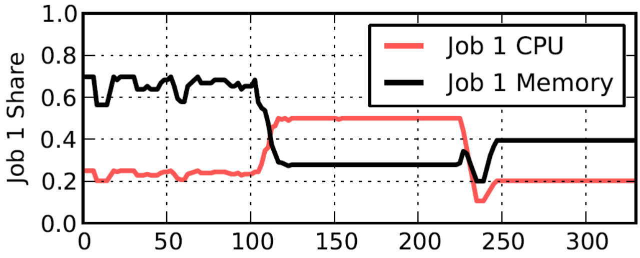


Figure 4: Job 1 Share

Dynamic Resource Sharing (Cont.)

Experimental Results

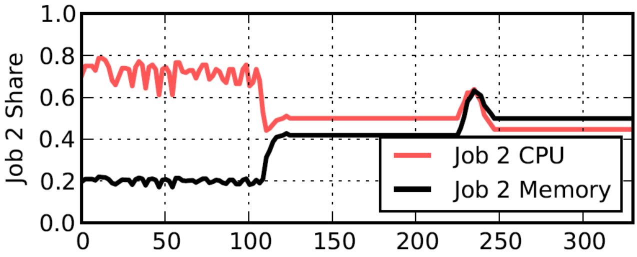


Figure 5: Job 2 Share

DRF v.s. Two Others

Rivals

- Slot-based fair scheduling.
- Max-min CPU-fair.

DRF v.s. Two Others

Rivals

- Slot-based fair scheduling.
- Max-min CPU-fair.

Experimental Settings

- 48-node Mesos cluster on EC2 instances.
- Instance $= \langle 8 CPUs, 7 GB \rangle$.
- Entity 1: 4 users, task $= \langle 1 CPUs, 0.5 GB \rangle$.
- Entity 2: 4 users, task $= \langle 2 CPUs, 2 GB \rangle$.
- 80 tasks for 1 job. As soon as a job finished, the user would launch another job with similar demands.
- Each experiment ran for 10 minutes.

DRF v.s. Two Others (Cont.)

Experimental Results

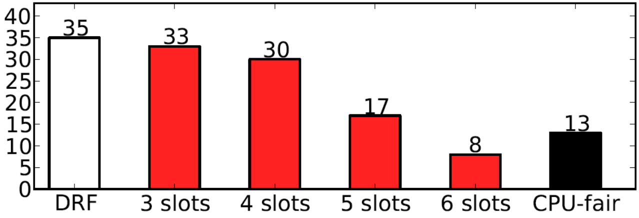


Figure 6: Number of large jobs completed for each allocation scheme in our comparison of DRF against slot-based fair sharing and CPU-only fair sharing.

DRF v.s. Two Others (Cont.)

Experimental Results

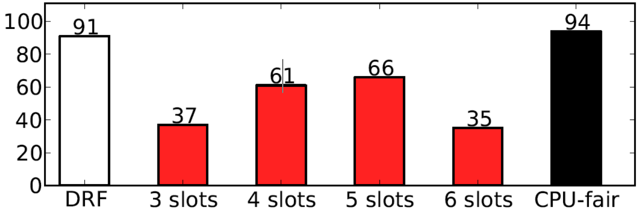


Figure 7: Number of small jobs completed for each allocation scheme in our comparison of DRF against slot-based fair sharing and CPU-only fair sharing.

DRF v.s. Two Others (Cont.)

Experimental Results

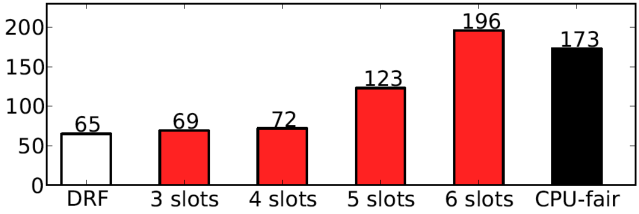


Figure 8: Average response time (in seconds) of large jobs for each allocation scheme in our comparison of DRF against slot-based fair sharing and CPU-only fair sharing.

DRF v.s. Two Others (Cont.)

Experimental Results

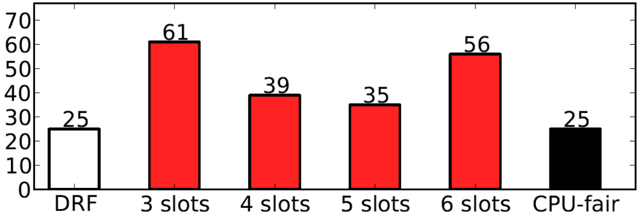


Figure 9: Average response time (in seconds) of small jobs for each allocation scheme in our comparison of DRF against slot-based fair sharing and CPU-only fair sharing.

Simulations using Facebook Traces

Rivals

- Hadoops fair scheduler.

Simulations using Facebook Traces

Rivals

- Hadoops fair scheduler.

Experimental Settings

- 2000-node cluster at Facebook.
- A one week period (October 2010).
- Hadoop MapReduce jobs.
- The traces are simulated on a smaller cluster of 400 nodes.
- Each node = 12 slots, 16 cores, and 32 GB memory.
- Each experiment ran for 10 minutes.

Simulations using Facebook Traces (Cont.)

Experimental Results

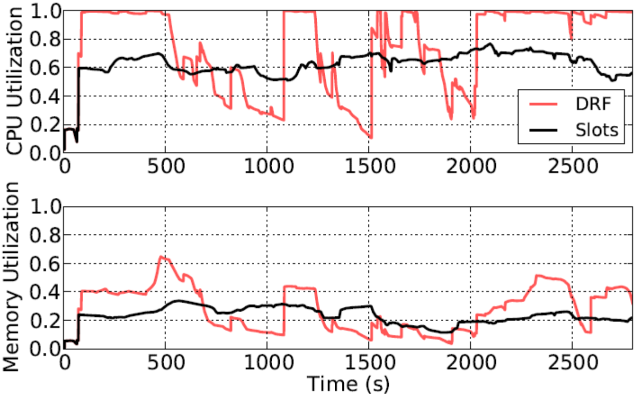


Figure 10: CPU and memory utilization for DRF and slot fairness for a trace from a Facebook Hadoop cluster.

Simulations using Facebook Traces (Cont.)

Experimental Results

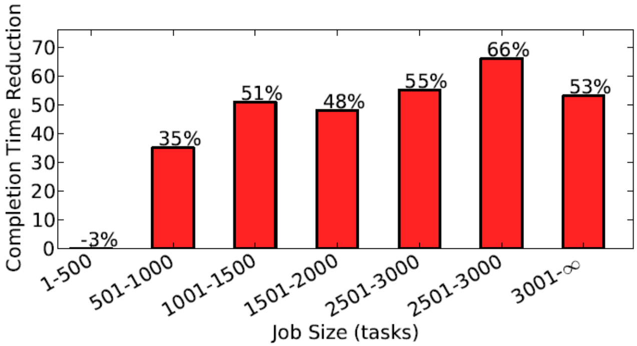


Figure 11: Average reduction of the completion times for different job sizes for a trace from a Facebook Hadoop cluster.

Conclusion

#1

DRF generalizes max-min fairness to multiple resource types.

Conclusion

#1

DRF generalizes max-min fairness to multiple resource types.

#2

DRF takes into account the heterogeneous demands, which leads to both fairer allocation of resources and higher utilization than existing solutions that allocate identical resource slices (slots) to all tasks.

Conclusion

#1

DRF generalizes max-min fairness to multiple resource types.

#2

DRF takes into account the heterogeneous demands, which leads to both fairer allocation of resources and higher utilization than existing solutions that allocate identical resource slices (slots) to all tasks.

#3

DRF has all the properties mentioned above.

Future Work

#1

In cluster environments with discrete tasks, minimize resource fragmentation without compromising fairness.

Future Work

#1

In cluster environments with discrete tasks, minimize resource fragmentation without compromising fairness.

#2

Take into account placement constraints, such as machine preferences.

Future Work (Cont.)

#3

DRF as an operating system scheduler.

Future Work (Cont.)

#3

DRF as an operating system scheduler.

#4

Whether DRF is the only possible strategy-proof policy for multi-resource fairness, given other desirable properties such Pareto efficiency.

Questions

#1

In the DRF, does it really make sense merely using the percentage of resources allocated for all types of resources as the metric of share?

Questions

#1

In the DRF, does it really make sense merely using the percentage of resources allocated for all types of resources as the metric of share?

#2

In the weighted DRF, does this really make sense?
$s_i = \mathsf{max}_j\{u_{i,j}/w_{i,j}\}$

Questions

#1

In the DRF, does it really make sense merely using the percentage of resources allocated for all types of resources as the metric of share?

#2

In the weighted DRF, does this really make sense?
$s_i = \max_j\{u_{i,j}/w_{i,j}\}$

#3

What if a task demand from a user is of much smaller "scale" than other users' tasks?

References I

[1] Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., & Stoica, I.
"Dominant Resource Fairness: Fair Allocation of Multiple Resource Types"
NSDI (Vol. 11, pp. 24-24), (2011, March).