Dominant Resource Fairness in Cloud Computing Systems with Heterogeneous Servers

Wei Wang, Baochun Li, Ben Liang
Department of Electrical and Computer Engineering
University of Toronto
April 30, 2014
Cloud computing system represents unprecedented heterogeneity

Server specification

Resource demand profiles of computing tasks
Heterogenous servers

Configurations of servers in one of Google’s clusters

CPU and memory units are normalized to the maximum server

<table>
<thead>
<tr>
<th>Number of servers</th>
<th>CPUs</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>6732</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>3863</td>
<td>0.50</td>
<td>0.25</td>
</tr>
<tr>
<td>1001</td>
<td>0.50</td>
<td>0.75</td>
</tr>
<tr>
<td>795</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>126</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>52</td>
<td>0.50</td>
<td>0.12</td>
</tr>
<tr>
<td>5</td>
<td>0.50</td>
<td>0.03</td>
</tr>
<tr>
<td>5</td>
<td>0.50</td>
<td>0.97</td>
</tr>
<tr>
<td>3</td>
<td>1.00</td>
<td>0.50</td>
</tr>
<tr>
<td>1</td>
<td>0.50</td>
<td>0.06</td>
</tr>
</tbody>
</table>
Heterogeneous resource demand

Ghodsi et al. NSDI11
How should resources be allocated *fairly and efficiently*?
State-of-the-Art Resource Allocation Mechanisms
Single-resource abstraction

Partition a server’s resources into slots

E.g., a slot = (1 CPU core, 2 GB RAM)

Allocate resources to users at the granularity of slots

Hadoop Fair Scheduler & Capacity Scheduler

Dryad Quincy scheduler

Ignores the heterogeneity of both server specifications and demand profiles
Dominant Resource Fairness (DRF)
Dominant Resource Fairness (DRF)

**Dominant resource**

The one that requires the most allocation share
Dominant Resource Fairness (DRF)

Dominant resource

The one that requires the most allocation share

For example

A cluster: (9 CPUs, 18 GB RAM)
Job of user 1: (1 CPU, 4 GB RAM)
Job of user 2: (3 CPUs, 1 GB RAM)
Dominant Resource Fairness (DRF)

**Dominant resource**

The one that requires the most allocation share

**For example**

A cluster: (9 CPUs, 18 GB RAM)

Job of user 1: (1 CPU, 4 GB RAM)

Job of user 2: (3 CPUs, 1 GB RAM)

**DRF allocation**

Equalize the *dominant share* each user receives

3 jobs for User 1: (3 CPUs, 12 GB)

2 jobs for User 2: (6 CPUs, 2 GB)

Equalized dominant share = 2/3
Why DRF?
Why DRF?

Addresses the demand heterogeneity
Why DRF?

Addresses the demand heterogeneity

Highly attractive allocation properties [Ghodsi11]

- Pareto optimality
- Envy freeness
- Truthfulness
- Sharing incentive
- and more…
However...

DRF assumes an *all-in-one* resource model

The entire resource pool is modeled as one super computer

Ignores the heterogeneity of servers

Allocation depends only on the **total amount** of resources

May lead to an infeasible allocation
An infeasible DRF allocation

The same example

A cluster: (9 CPUs, 18 GB)
Job of user 1: (1 CPU, 4 GB)
Job of user 2: (3 CPUs, 1 GB)

DRF allocation

3 jobs for User 1: (3 CPUs, 12 GB)
2 jobs for User 2: (6 CPUs, 2 GB)
An infeasible DRF allocation

The same example

A cluster: (9 CPUs, 18 GB)
Job of user 1: (1 CPU, 4 GB)
Job of user 2: (3 CPUs, 1 GB)

DRF allocation

3 jobs for User 1: (3 CPUs, 12 GB)
2 jobs for User 2: (6 CPUs, 2 GB)
An infeasible DRF allocation

The same example

A cluster: (9 CPUs, 18 GB)

Job of user 1: (1 CPU, 4 GB)

Job of user 2: (3 CPUs, 1 GB)

DRF allocation

3 jobs for User 1: (3 CPUs, 12 GB)

2 jobs for User 2: (6 CPUs, 2 GB)

User 1 can schedule at most 2 jobs!
A quick fix of DRF

Per-Server DRF

For each server, allocate its resources to all users, using DRF

However…

Per-server DRF may lead to an arbitrarily inefficient allocation

See the paper for details
Can the attractiveness of DRF extend to a heterogeneous environment?
The ambiguity of dominant resource

The same example

A cluster: (9 CPUs, 18 GB)
Job of user 1: (1 CPU, 4 GB)

- Server 1: (1 CPU, 14 GB)
- Server 2: (8 CPUs, 4 GB)
The ambiguity of dominant resource

The same example

A cluster: (9 CPUs, 18 GB)
Job of user 1: (1 CPU, 4 GB)

How to define dominant resource?

For server 1, the dominant resource is CPU
For server 2, the dominant resource is memory
For the entire resource pool, the dominant resource is memory
Our answer: DRFH

A generalization of DRF mechanism in Heterogeneous environments

Equalizes every user’s global dominant share

Retains almost all the attractive allocation properties of DRF

- Pareto optimality
- Envy-freeness
- Truthfulness
- Weak sharing incentive
- and more…

Easy to implement
DRFH Allocation
A global view of dominant resource

Global dominant resource

The one that requires the maximum allocation share of the entire resource pool

The same example

A cluster: (9 CPUs, 18 GB)
Job of user 1: (1 CPU, 4 GB)

Memory is the global dominant resource
Key intuition

Max-min fairness on the global dominant resources, subject to resource constraints per server

\[
\max_A \min_{i \in U} G_i(A_i) \\
\text{s.t. } \sum_{i \in U} A_{ilr} \leq c_{lr}, \forall l \in S, r \in R.
\]

Allocation share of resource \( r \) user \( i \) receives on server \( l \)

Global dominant share

Total availability of resource \( r \) on server \( l \)
DRFH Properties
Fairness property
Fairness property

**DRFH is envy-free**

No user can schedule more computing tasks by taking the other’s resource allocation

No one will envy the other’s allocation
Fairness property

DRFH is envy-free

No user can schedule more computing tasks by taking the other’s resource allocation

No one will envy the other’s allocation

DRFH is truthful

No user can schedule more computing tasks by misreporting its resource demand

Strategic behaviours are commonly seen in real system [Ghodsi11]
Fairness property

DRFH is envy-free

No user can schedule more computing tasks by taking the other’s resource allocation

No one will envy the other’s allocation

DRFH is truthful

No user can schedule more computing tasks by misreporting its resource demand

Strategic behaviours are commonly seen in real system [Ghodsi11]
Resource utilization

DRFH is Pareto optimal

No user can schedule more tasks without decreasing the number of tasks scheduled for the others

No resource that could be utilized to serve a user is left idle
Service isolation

Equal partition

Allocation $A$ is an equal partition if it divides every resource evenly among all $n$ users

$$\sum_{l \in S} A_{ilr} = 1/n, \quad \forall r \in R, \ i \in U$$

Allocation share of resource $r$ user $i$ receives on server $l$
Service isolation

Equal partition

Allocation $A$ is an equal partition if it divides every resource evenly among all $n$ users:

$$\sum_{l \in S} A_{ilr} = 1/n, \quad \forall r \in R, \ i \in U$$

Allocation share of resource $r$ user $i$ receives on server $l$

Weak sharing incentive

There exists an equal allocation $A'$ under which each user schedules fewer tasks than those under DRFH.

DRFH is unanimously preferred to an equal allocation by all users.
## Comparison

<table>
<thead>
<tr>
<th>DRFH</th>
<th>DRF (all-in-one model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pareto optimality</td>
<td>Pareto optimality</td>
</tr>
<tr>
<td>Envy freeness</td>
<td>Envy freeness</td>
</tr>
<tr>
<td>Truthfulness</td>
<td>Truthfulness</td>
</tr>
<tr>
<td><strong>Weak</strong> sharing incentive</td>
<td><strong>Strong</strong> sharing incentive</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th>DRFH</th>
<th>DRF (all-in-one model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pareto optimality</td>
<td>Pareto optimality</td>
</tr>
<tr>
<td>Envy freeness</td>
<td>Envy freeness</td>
</tr>
<tr>
<td>Truthfulness</td>
<td>Truthfulness</td>
</tr>
<tr>
<td><strong>Weak</strong> sharing incentive</td>
<td><strong>Strong</strong> sharing incentive</td>
</tr>
</tbody>
</table>

DRFH retains almost all the attractive properties of DRF
Trace-Driven Simulation
Resource utilization

We design a Best-Fit heuristic that implements DRFH as a single-resource abstraction such as a slot scheduler, DRFH is truthful. As for the DRFH implementations, we assume a homogeneous environment where all servers are of the same hardware configuration. The DRFH scenario is envy-free, Pareto optimal, and achieves significant improvements in resource utilization, leading to much shorter job completion times. DRFH shows no improvement over Slots scheduler in a real-world system. Our large-scale simulations driven by the Google Cluster-Usage Traces, http://code.google.com/p/googleclusterdata/., show that DRFH leads to uniformly higher resource utilization than the First-Fit alternative at all times.

Fig. 6. DRFH improvements on job completion times over Slots scheduler.

In a heterogeneous cloud computing system where the resource pool is composed of a large number of servers with different resource types, equalizing the global dominant resource allocation from a single server to multiple heterogeneous servers becomes challenging. We analyze DRFH and show that it retains almost all desirable properties that DRF provides in the single-server scenario. Notably, DRFH is envy-free, Pareto optimal, and for each user, we run its computing tasks on a dedicated server, known as a slot scheduler, and for each user submitted. We see that Best-Fit DRFH leads to higher task completion ratio — the number of tasks completed over the number of tasks submitted for every user using Best-Fit and Slots. As a complementary study, Fig. 7 computes the task completion ratio for almost all users. Around 20% users see fewer tasks finished in the shared environment. Even for these users, task completion ratio decreases only slightly, as can be seen from Fig. 8. DRFH does not guarantee 100% sharing incentive for all users, in particular, only 2% users see fewer tasks finished in the shared cloud (SC). We then compare the task completion ratio in DC (DCs) and the shared cloud (SC).

Fig. 8. Task completion ratio in SC.
Google cluster servers in Table I. We compare Best-Fit DRFH configurations are randomly drawn from the distribution of 2,000 servers so that fairness becomes relevant. The server and simulate it on a smaller cloud computing system of the 24-hour computing demand data from the Google traces. We take allocation at all times.

Users finish their tasks. Throughout the simulation, we see that global dominant resources. A similar process repeats until all its tasks and departs at 1080 s. After that, only users 2 and dominant share of 26% to all three users until user 1 finishes and 0.3 memory. The algorithm now allocates the same global to submit memory-intensive tasks, each requiring 0.1 CPU receive 44% global dominant share. At 500 s, user 3 starts resources, leading to a DRFH allocation in which both users CPU and 0.1 memory. Both users now compete for computing user 2 joins and submits CPU-heavy tasks, each requiring 0.5 share. This allocation continues until 200 s, at which time of its task. As shown in Fig. 4, since only user 1 is active at the beginning, requiring 0.2 CPU and 0.3 memory for each units and 51.32 memory units in total. User 1 joins the system Fig. 4. CPU, memory, and global dominant share for three users on a 100-

We next evaluate the resource utilization of Best-Fit DRFH naturally the larger the job is, the more improvement one may expect. We have observed for those containing more tasks. Generally, While DRFH shows no improvement over Slots scheduler based on the number of its computing tasks, and for each detailed breakdown, where jobs are classified into 5 categories DRFH and Slots scheduler are depicted. Fig. 6b offers a more translations significantly outperform the traditional Slots scheduler utilization than the First-Fit alternative at all times. We see that Best-Fit DRFH leads to uniformly higher resource configurations [6]. As for the DRFH implementations, we simultaneous environment where all servers are of the same hardware latter ignores the heterogeneity of both servers and workload. With much higher resource utilization, mainly because the with two other benchmarks, the traditional Slots schedulers Fig. 6. DRFH improvements on job completion times over Slots scheduler.
Conclusions

We have studied a multi-resource fair allocation problem in a heterogeneous cloud computing system.

We have generalized DRF to DRFH and shown that it possesses a set of highly attractive allocation properties.

We have designed an effective heuristic algorithm that implements DRFH in a real-world system.

http://iqua.ece.toronto.edu/~weiwang/