Student Name: Nikhil Doifode
Student ID#: 112714121

Review of “Dominant Resource Fairness: Fair Allocation of Multiple Resource Types”

Cloud computing involves a lot of resource sharing at any data center. For this, the popular method used is “max-min fairness” sharing algorithm.
“Max-Min fairness” sharing algorithm has 2 important properties:
- **Sharing incentive**: Each user should get at least (1/nth) of the total resource available.
- **Strategy-proofness:** Users should not be able to benefit from lying about their resource demands.
- **Envy Freeness:** Each user’s allocation should be fair so that, any other allocation is well suited for current user.
- **Pareto Efficiency:** Increase of allocation of resource to one user should not happen without decreasing the allocation of other user

In addition to this, “max-min fairness” also provides flexibility based on the needs for priority, weight, reservation, etc.
However, “max-min fairness” considers resource to be of single type like CPU, etc. But as we know the demand for resource is not just for CPU. Resource can be anything like Memory, Disk, I/O and CPU. Moreover, for each user the demand for the each resource can be different.

E.g. In the cluster with total 9 CPU’s and 18GB of RAM represented as <9(CPU), 18(RAM)>
User 1 needs 1 CPU and 4GB RAM for each task represented as <1(CPU), 4(RAM)>
User 2 needs 3 CPU and 1GB RAM for each task represented as <3(CPU), 1(RAM)>

Now, in this case “max-min fairness will not give better output” since it will consider only one resource for dividing the total resource and it will end up making the cluster either underperform or overperform.

Dominant Resource Fairness (DRF): DRF algorithm is based on the idea of allocation of resource based on the “dominant resource”.
E.g.: In the above example for User 1, RAM is dominant resource since it’s share (4/18 = 2/9) is greater that User 2’s share of (1/18).
Similarly, for User 2, CPU is the dominant resource since it’s share (3/9 = 1/3) is greater than User 1’s share (1/9).

Let x and y be the number of tasks allocated by DRF to users A and B, respectively. So User 1 receives <x (CPU), 4x (RAM)>, while User 2 gets <3y (CPU), y (RAM)>.
Now to explain our allocation we have following restrictions.
max(x, y)
x + 3y <= 9 (CPU constraint)
4x + y <= 18 (RAM constraint)

Also, the dominant shares of users 1 and 2 are 2x = 9 and y=3, respectively (their corresponding shares of memory and CPU). Hence to maximize the allocation we can divide in the ratio of:
2x/9 = y/3
Solving this we get, x = 3 and y = 2. Thus User 1 gets <3(CPU), 12(RAM)> and user 2 gets <6(CPU), 2(RAM)>.
This type of resource sharing still satisfies all the 4 properties, we noted above.

There are other multiple-resource sharing algorithm:

- Asset fairness
- Competitive Equilibrium from Equal Incomes (CEEI)

But both these algorithms have some drawback and violate the two properties which we stated earlier.

Asset Fairness: Asset fairness tries to maximize allocation of dominant resources by giving equal number of total allocation to each user.
For the above example, it can be given as:
max(x, y)
x + 3y <= 9 (CPU constraint)
4x + y <= 18 (RAM constraint)
But now, we are treating 1 unit of RAM equal to 1 unit of CPU hence it gives the allocation as, 6x = 7y.
And we get x = 2.52 and y = 2.16.
Hence, User 1 gets <2.5 (CPU), 10.1 (RAM)> and user 2 gets <6.5 (CPU), 2.2 (RAM)>. This way of sharing violates the first property we stated above “Sharing incentive”. Since it ends up getting the user less that ($1/n^{th}$) of total resources available.

Competitive Equilibrium from Equal Incomes (CEEI): CEEI tries to maximize the product of the dominant shares by resource exchange.
For the above example, it can be given as:
max(x * y)
x + 3y <= 9 (CPU constraint)
4x + y <= 18 (RAM constraint)
And we get x = 45/11 and y = 18/11.
Hence, User 1 gets <4.1 (CPU), 16.4 (RAM)> and user 2 gets <4.9 (CPU), 1.6 (RAM)>. This way of sharing violates the second property we stated above “Strategy-proofness”. Since, user can increase its allocation by lying about its need of resource.

Performance: In performance, when deployed in Mesos, DRF brings fairness in the resource allocation while considering the different demands of each user for each type of resource. It also provides better cloud utilization to maximize the output. Hence, its output for large and small tasks will be better than CPU based sharing used in “max-min fairness” and slot based sharing used nowadays.

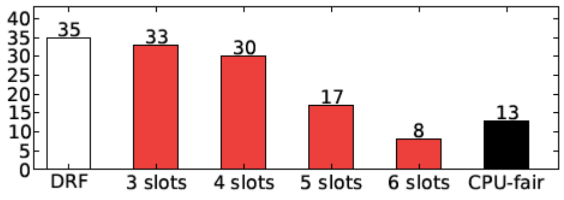

Figure 9: Number of large jobs completed for each allocation scheme in our comparison of DRF against slot-based fair sharing and CPU-only fair sharing.

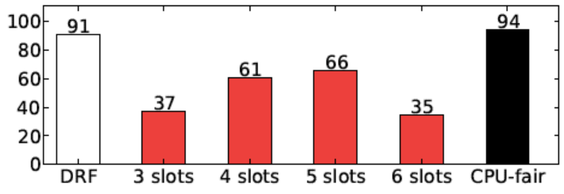

Figure 10: Number of small jobs completed for each allocation scheme in our comparison of DRF against slot-based fair sharing and CPU-only fair sharing.

As we can see in the above image DRF performs far better that slot based and CPU bases sharing.

Strengths:

1. Main strength of DRF sharing algorithm is that in case multiple users where each user will have different requirement of different resources, it's allocation address all these needs to maximize the utilization and output.
2. DRF satisfies Strategy-proofness property so that users are incentivized to report their demands accurately. DRF also incentivizes users to share resources by ensuring that users perform at least as well in a shared cluster as they would in smaller, separate clusters.
3. DRF also satisfies other properties demonstrated by "max-min fairness" like Sharing incentive, Envy Freeness, Pareto Efficiency, Single resource fairness, Bottleneck fairness and Population monotonicity

Weakness:

1. DRF doesn't consider fairness when tasks have placement constraints, such as machine preferences, etc.

Questions:

1. Difference between "max-min fairness" and DRF?

Ans: Main difference between DRF sharing algorithm and "max-min fairness" sharing is that in case multiple users where each user will have different requirement of different resources DRF will take all that into consideration, while "max-min fairness" will only consider CPU requirements for resources sharing for all type of resources

2. How does DRF work?

Ans: DRF is based on the idea of allocation of resource based on the "dominant resource". E.g.: In the cluster with total 9 CPU's and 18GB of RAM represented as <9(CPU), 18(RAM)>
User 1 needs 1 CPU and 4GB RAM for each task represented as <1(CPU), 4(RAM)>

User 2 needs 3 CPU and 1GB RAM for each task represented as <3(CPU), 1(RAM)>
In the above example for User 1, RAM is dominant resource since it's share (4/18 = 2/9) is greater that User 2's share of (1/18).
Similarly, for User 2, CPU is the dominant resource since it's share (3/9 = 1/3) is greater than User 1's share (1/9).

Let x and y be the number of tasks allocated by DRF to users A and B, respectively. So User 1 receives <x (CPU), 4x (RAM)>, while User 2 gets <3y (CPU), y (RAM)>. Now to explain our allocation we have following restrictions.
max(x, y)
x + 3y <= 9 (CPU constraint)
4x + y <= 18 (RAM constraint)

Also, the dominant shares of users 1 and 2 are 2x = 9 and y=3, respectively (their corresponding shares of memory and CPU). Hence to maximize the allocation we can divide in the ratio of:
2x/9 = y/3
Solving this we get, x = 3 and y = 2. Thus User 1 gets <3(CPU), 12(RAM)> and user 2 gets <6(CPU), 2(RAM)>.

In case it users are coming in the system it will take the user with lowest dominant share than the current one by storing them in the min heap.

3. Properties provided by DRF and meaning of those properties?

Ans: Properties provides by DRF are Strategy-proofness, Sharing incentive, Envy Freeness, Pareto Efficiency, Single resource fairness, Bottleneck fairness and Population monotonicity.

Strategy-proofness: If user lies about its requirements, it cannot increase its allocation. It will still remain the same.
Sharing incentive: Provides incentives to users to share resources also makes sure that no user is better off on its separate cluster and hence can get at least ($1/n^{th}$) of the total resources.
Envy Freeness: Provides best allocation possible to each user so no user would want to trade its allocation with that of another user
Pareto Efficiency: It allocates all available resources hence is not possible to improve the allocation of a user without decreasing the allocation of another user.
Single resource fairness: And as we can see if there is only one resource to share then DRF becomes "max-min fairness" algorithm. Since none of the properties are violated.
Bottleneck fairness: DRF considers fairness and proper allocation of resources even when one resource is demanded the most by multiple users which may create bottleneck.
Population monotonicity: DRF makes sure that when one user leaves the system and relinquishes its resources, none of the allocations of the remaining users should decrease

Reference:
Dominant Resource Fairness: Fair Allocation of Multiple Resource Types - *Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, Ion Stoica*