Reading Summary 1

of Dominant Resource Fairness: Fair Allocation of Multiple Resource Types
Tianyang Liu 30193168

A computer cluster may contain different combinations of resources. Take AWS as an example: it primarily runs CPU, GPU, Mem, and storage instances. Users, however, have different tastes and requirements toward these four categories. Some users build neural networks on AWS and need great amounts of calculation; Others conducting image processing may prefer GPU boosting. How to resolve the conflicts is the topic of this paper. We may read through the abstract, and learn that three important judgements of this paper are: First, DRF encourages users to share resources with users till the win-win situation, upon the knowledge that equally allocated resources reduce overall performance. Second, DRF forbids getting more resources via boasting actual needs, and there are some judgement modules to ensure this. Third, it bans trade between DRF users. Last, increase of the allocation of one user accompanies decrease of the other, which means the total resource is constant and optimal. DRF resource manager can be deployed on any cloud to increase efficiency.

This paper has lots of advantages. First it is creative because it proposes the novel concept of DRF, which has the potential to be deployed on big clusters. Since the operation efficiency of clusters is a bottleneck, the use of DRF can resolve both the issue of electricity cost and time consumption. It is important also because it is bottom-up, resolving the conflicts from basis, bringing resonable allocation of resources and higher utilization rates than existing solutions. Second, this paper has done deep research with numerous experiments. It first builds up it own theorem satisfying all four properties of sharing incentive, strategy proofness, envy freeness, and pareto efficiency indicating the state that impossible to make one better off without making at least one worse off. Then has alternative solutions considering all possibilities. In the analysis and experiment section, it through various perspectives compares the pros and cons of DRF along with the theorem proof. Third, deep thinking and prospect. In clusters with mutiple tasks, it remains problems to reduce fragmentation of resource without compromising the fairness. Another issue is ensuring fairness when tasks have placement constraints. One more challenging way is to implement the DRF in PC as schedulers. All of these make this paper accepted and appreciated.

No paper is perfect. First, DRF cares so much on the fairness that it overlooks actual needs. In real pipeline, schedulers like Hadoop Capacity to segment resources into several queues run better. But Mesos DRF uses RO which brings wasteful resource fragment, therefore will ruin the overall ability. Second, the resource monotonicity feature in table two cannot be fulfilled by DRF, and the perfect scheduling protocol remains to be discovered. Last but not least, DRF concentrates mostly on the primary resource type, which is bad when all resource types are requested much. As an instance, I may need both CPU and GPU for 30 percent in a group of five. Then how the schedular will work is a question. In sum, anyone concept should be constrained to work in a certain range, which is also true for DRF.

At last I would like to brainstorm ideas upon DRF. A fact is that, DRF is more suitable than current allocation rules in heterogeneous clusters. I may adopt this principle in real life where all resources are heterogenerous and belong to various categories. As an instance, in communism future people live together but have different needs on resources. If trades between people happen, this is never communism and could be harmful to the whole society. DRF can eliminate this problem because it encourages sharing, forbids trading, and pursues the optimal result for all people. DRF can also be implemented in small computers thus pursue the optimal performance when tasks send queries to hardware. In both the situations above, there should be some targeting modifications.