Md Majid Jahangir | SBU ID - 112077145
AMS 560 - Big Data Systems, Algorithms and Networks

Assignment 1 (Paper Reading)
Dominant Resource Fairness : Fair Allocation of Multiple Resource Types

Motivation of the Paper : Weighted max-min fairness policy to allocate homogeneous resources by maximizing the minimum allocation to a user had been quite famous up until the advent of cloud computing. In this era of cloud computing, the scenario has changed where we now deal with mostly heterogeneous resources having heterogeneous resource demands. It has been shown that applying a resource allocation policy for a single resource type like that of weighted max-min fairness policy, to such an environment of heterogeneous resources and demands leads to inefficient allocation as a slot quite often does not match the task demands. Such inefficient allocation has increased the need for fair resource allocation policies for an environment having heterogeneous resources and user demands and the paper proposes one such policy named as Dominant Resource Fairness (DRF).

Prior solutions : Max-min fairness resource allocation policy was a prior solution but only applied to homogeneous resources. The policy would allocate resources by maximising the minimum allocation received by a user in the system. Also, this policy had also been generalised to allocate share of a resource to a user proportional to its weight and it could support variety of other allocation policies such as priority or deadline based allocation.

Key ideas of the proposed solution : DRF is a generalization of max-min fairness of resource allocation for heterogeneous resources therefore for a single resource scenario it reduces to max-min fairness policy. The key idea is that in a multi-resource environment, the allocation of a user should be determined by the user's dominant share. A user's dominant share is the maximum share that has been allocated of any resource to that user. Thus, based on this DRF tries to maximize the minimum dominant share across all users in allocating any resources. The following is a summary on the key questions of the solution -

1. **How does DRF work?** - DRF computes and tracks the total resources allocated to each user along with theirs dominant share. Whenever a resource has to be allocated, DRF picks a user that has a task ready to be launched and also that has the minimum dominant share among all users. If DRF finds that the selected user's task demand can be met with available resources, the task will be launched and correspondingly the dominant share of that user along with the minimum dominant share among all users in the system is updated for next allocation. This cycle repeats till there are some task requests of some users that can be met with available resources. Thus, DRF maximizes the smallest dominant share in the system leading to equalizing of user's dominant shares at times hence ensuring fair allocation.
    i.
2. **Properties provided by DRF** - 4 key properties that helps in determining fair allocation policy-
    a. **Sharing Incentive** - In a heterogeneous request and resource environment it is always advantageous for a user to share the cluster than having its own fixed partition of cluster. It is because in a heterogeneous demand system, a fixed partition of a cluster for a user may fall short when the request demand increases thus over utilizing while there may be some users who are under utilizing their resources.
    b. **Strategy-proofness** - A user should not be able to benefit by asking for more resources than it actually needs. This complements the sharing incentive because if every user lies about actual demands then in a heterogeneous demand system, when demand increases for each user, all of them will over utilise resources as no resource is available to be shared since every users lied about their usage.
    c. **Envy-freeness** - A user should be satisfied with it allocation and not prefer the allocation of any other user. This embodies fairness.
    d. **Pareto efficiency** - It should not be possible to increase the allocation of a user without decreasing the allocation at least another user. Thus this property leads to maximum system utilization.

3. Some other nice to have properties are -
    a. Single resource fairness - It should reduce to max-min fairness for a single resource environment.
    b. Bottleneck fairness - The solution should reduce to max-min fairness for a resource that is being demanded the most by every user.
    c. Population monotonicity - When a user leaves system and relinquishes its resources, the allocations should not decrease for the remaining users.
    d. Resource monotonicity - Adding more resources to the system should not decrease the allocations of the existing users.
1.
4. **Difference between DRF and max-min fairness** - Max-min fairness applies to a single resource type environment whereas DRF applies to an environment having multiple/heterogeneous resources with heterogeneous demands. In max-min fairness, the goal is to maximise the minimum share to each user whose demand is not fully serviced. Whereas, DRF tries to maximise the minimum “dominant” share “across all users”, where a dominant share is the maximum share that a user has been allocated of any resources.

Strengths : The strengths of the paper or its solution DRF are -

1. It applies to an environment which is prevalent today i.e. in an era of cloud computing and huge data-centers. Unlike decades ago, the system’s environment has changed which now comprises heterogeneous resources and heterogeneous demands because of which a very simple yet powerful allocation policy like max-min fairness cannot be applied as it is effective and leads to efficient allocation only for single resource type.
2. The paper also demonstrates this with experimental statistics by performing experiments on real clustered systems employing fair schedulers such as Quincy and Hadoop Fair Scheduler that allocate resources at granularity of slots, ignoring heterogeneity of demands and hence leading to inefficient allocation. The experiments showed via CDF of demand to slot ratio for both CPU and memory that under such scheduling system the system would either under utilise or over utilise their resources.
3. DRF tries to maximise the allocations by equalising the dominant share of resources for each users through its allocation algorithm. In doing so, DRF ensures that for instance a user’s dominant resource is CPU while another user’s dominant resource is memory, so in its allocation, it tries to equalise the dominant shares for both users thus, user A’s task being CPU extensive and user’s B task being memory extensive, both the users gets equal share of memory and CPU to perform their tasks.
4. DRF also provides performance isolation since it satisfies the sharing incentive property. As a result of this, DRF guarantees a minimum allocation to each user i.e. a user among ‘n’ users cannot do worse than owning 1/n partition of the cluster irresspective of the demands of the other users.
5. By satisfying sharing incentive and strategy-proofness property, DRF maximizes the overall utilization of the system. In a heterogeneous request and resource environment, a user may see sudden increase or decrease in task demands, hence it is always advantageous for a user to share the cluster than having its own fixed partition of cluster. Having a fixed partition of a cluster, a user may fall short when the request demand increases thus over utilizing while when request decreases user may under utilize their resources. Hence, by satisfying sharing incentive and strategy proofness property, each user may share the cluster as it deems fit as per the demand, thus it increases throughput and system utilization and resembles a fair allocation policy.
6. Also, the fact that the DRF need not always equalize user’s dominant shares. When a user’s total demand is met, so the surplus resources for that user will be split among the other users. And, if a resource type gets exhausted, user’s task that do not need that resource will still continue to get shares of other resources to complete the task thus maintaining the progress of the system.

Weakness : Application of DRF works effectively and efficiently under the assumption that there is a one big resource pool from which resources can be allocated in small amounts. However, in practice

this is not the case quite often. Clusters may consist of multiple small machines where resources can be allocated to tasks in discrete amounts. In such a scenario, often resource fragmentation happens and fairness of resource allocation is affected in such discrete cases and resource shares are not equal. Hence the paper doesn't talk about minimizing of resource fragmentation without compromising fairness in cluster environments with discrete tasks.

Performance : In terms of dynamic resource sharing, experiments were performed using DRF on Mesos cluster on Amazon EC2. 2 jobs were launched having different resource demands at different times during a fixed interval. Job 1's dominant resource was RAM while other's dominant resource was CPU. DRF equalised the job's shares of their respective dominant resources. Thus, it showed that jobs benefited that from running in shared cluster than a fixed partition of cluster for each. Even on changing the size of the job after an interval, whenever both the job's dominant resource changed to either CPU or memory, DRF would dynamically equalise the share for the same for both the job.
On evaluating it against a slot based fair scheduling policy such as Hadoop Fair Scheduler and Quincy, a test was performed where a class of users would submit large jobs while another class of users would submit smaller jobs and it was found that both the throughput and the job response times were worse than that of DRF, regardless of varying number of slots for the jobs. Since DRF based scheduler was aware of both resources, it had the highest throughput and lowest response time
Similarly, running simulations using Facebook traces where data comprised Hadoop MapReduce jobs, DRF showed lesser average job completion time and higher utilization compared to Hadoop Fair Scheduler.

Follow Up Works :

1. **Hierarchical scheduling for diverse datacenter workloads[4]** - Arka A. Bhattacharya, David Culler, Eric Friedman, Ali Ghodsi, Scott Shenker, Ion Stoica : This paper produces an algorithm implemented in Hadoop, generalizing DRF to support hierarchies. The evaluation shows that the proposed algorithm, H-DRF, avoids starvation and resource inefficiencies of the existing open-source schedulers and outperforms slot scheduling.
2. **Fair Allocation of Heterogeneous and InterchangeableResources[3]** - Xiao Sun, Tan N. Le, Mosharaf Chowdhury, Zhenhua Liu : The paper shows that DRF fail to provide its properties for interchangeable resources. It formulates a novel multiinterchangeable resource allocation (MIRA) problem where some resources are interchangeable and the Budget-based (BUD) algorithm, which preserves Pareto efficiency, sharing incentive, and envyfreeness while providing better performance over currently used algorithms.
3. **Choosy: max-min fair sharing for datacenter jobs with constraints[2]** - Ali Ghodsi, Matei Zaharia, Scott Shenker, Ion Stoica : Nowadays, an increasing number of jobs have hard *placement constraints*, restricting the machines they can run on due to special hardware or software requirements and it is not clear as to how to define, and achieve, max-min fairness in the presence of such constraints. The paper proposes Constrained Max-Min Fairness (CMMF), which is an extension to max-min fairness that supports placement constraints, and shows that it is the only policy satisfying an important property that incentivizes users to pool resources.

References -

1. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types - Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, Ion Stoica
2. ACM Digital Library - https://dl.acm.org/citation.cfm?id=2465387
3. ACM Digital Library - https://dl.acm.org/citation.cfm?id=3305227
4. ACM Digital Library - https://dl.acm.org/citation.cfm?id=2523637