FUD – Balancing Scheduling Parameters in Shared Computing Environments

Art Sedighi, Dr. Milton Smith
Department of Industrial, Manufacturing, & Systems Engineering
Texas Tech University
Lubbock, TX
{art.sedighi, milton.smith}@ttu.edu

Dr. Yuefen Deng
Department of Applied Mathematics & Statistics
Stony Brook University
Stony Brook, NY
yuefan.deng@stonybrook.edu

***Abstract*— Shared computing environments such as Cloud, HPC and Grid Computing present a challenge for scheduling systems as they seek to balance incoming requests with available resources, maintain high utilization, be fair among users, and cope with environmental dynamicity. In this paper, we will introduce the FUD theorem. The FUD theorem is based on the premise that a scheduler's desire to optimize the three system parameters: Fairness, Utilization and Dynamicity (FUD) comes at a cost. These parameters adversely affect one another, and thus, in a shared computing environment, a scheduler is unable to be fair while fully utilizing the available resources and still deal with the dynamicity of the environment. The presented FUD model argues for relaxing one of the three parameters to optimize scheduling decisions based on the remaining two.**

Keywords—Scheduling, resource management, fairness, utilization, dynamicity, shared computing environment

I. INTRODUCTION

Cloud [2-4], and high performance computing systems [5] represent a shared computing environment where many users compete for resources to complete their tasks. Regardless of the specifics, sharing resources with others in any environment entices users to compete for resources and at times engage in unfair practices [6, 7]. To compensate, many system providers take measures by using a scheduling system[1] to assign resources efficiently to incoming requests. With the prevalence of Cloud infrastructure with very large user bases, the issue of sharing and resource management is even more important. There have been several efforts [8-11] to tackle the problem of scheduling for a Cloud environment, with [10] targeting high utilization, and [11] focusing in on dynamicity vis-à-vis job grouping. [8] is using SLA for cloud to enforce fairness across the users of the cloud.

In High Performance Computing (HPC), the scheduler has been the topic of much research over the decades, and continues to be so [12]. With the prevalence of big data, HPC environments are turning into high throughput computing environments [13], and schedulers have taken on the function of a data transport layer.

Scheduling is known to be a NP-Complete problem [14], where the proposed algorithms and solutions aim at best effort scheduling based on a single criteria such as the due-date of the job, known as Earliest Due Date scheduling [15], or multi-criteria scheduling such as the models presented in [16]. We stipulate that regardless of the criteria by which decisions are made, it is the job of the scheduler to optimize three system conditions (FUD) (See Figure III-1):

- Fairness (F): where it is the job of all schedulers to "seem fair" [17] in the way which resources are distributed
- Utilization (U): where it is the job of all schedulers to achieve and maintain high utilization of resources under management
- Dynamicity (D): where it is the goal of all schedulers to be able to deal with dynamic aspects of the system

[1] Terms "resource management" and "scheduling" are used interchangeably throughout this paper, as it pertains to the context in question. For example, in HPC environments, we will use the term scheduler, but in scenarios where we are concerned with fairness of a common resource, we will use the term resource management.

such as new users, new requests, new resources, failures, etc.

There are other system parameters [15] such as time in system, flow time, start time, speed up time, and many others, but all the parameters are affected by or will affect one or more of the FUD parameters. For example, time in system affects fairness; flow time is affected by utilization and dynamicity of the environment; and speed up time is affected by utilization and fairness. We argue that fairness, utilization and dynamicity represent the set of primary parameters in a scheduling system. We further argue that optimizing for all three parameters is not possible, and that at least one of these parameters must be relaxed to better optimize the other two parameters and the system.

II. Background on Shared Computing Systems

Shared computing environments represent any type of environment where the desired resource is limited and shared amongst several users, with no single user having full access to that environment. There are three classes of shared computing environments a) cluster computing, b) grid computing and c) cloud computing, where each type is mainly differentiated by the scale and the locality of deployment infrastructure. TABLE II-1 shows the difference between the various classes of HPC [18]. Cluster computing environments tend to be smaller in size, orders of 100's of nodes or servers. Grid and Cloud computing environments, however, tend to be much larger in size and span multiple physical locations.

TABLE II-1: COMPARISON OF CLUSTER, GRID AND CLOUD COMPUTING

Feature	Cluster	Grid	Cloud
Size (# of nodes)	Small to medium (100's)	Large (1000's)	Small to large (100's – 1000's)
Network type	Private LAN	Private LAN or WAN	Public WAN
System Management method	Centralized	Centralized or Decentralized	Centralized or Decentralized
Deployment location	One location Very close proximity	Multiple locations Close proximity within one location	Multiple locations

The need for a shared environment has risen from the fact that 1) the need for computational power has increased, while 2) the average utilization of a datacenter is consistently about 5-20% [2]. Meanwhile, the cost of initial acquisition of a computer is about 1/3 of the total cost of management and maintenance for the life duration of that computer [3]. As such, it makes sense to share the unused resources amongst many users, if there is a way to ensure that fairness is intact, and that all users are getting their fair share of available resources.

III. Scheduling in Shared Computing Environments

Tasks that run in a shared computing environment (i.e. CPU core[s]) can be of various sizes and may require a myriad of system configurations. No single user owns the resources in full, and thus shares access rights with other users. This creates a contention that each user is competing with other users of the system. This affect is universal in that if more than one user is requesting access to a shared resource, there will be competition, and a scheduler is the intermediator that controls access. The primary goal of a scheduler is to maintain a high system utilization, and to reduce the time in system, the time it takes for a request to be processed [15]. All schedulers need to be axiomatically fair [17], and all schedulers need to deal with dynamicity of the shared computing environment. These three pillars are shown in Figure III-1, where optimization of one aspect comes at the cost of at least one of the others.

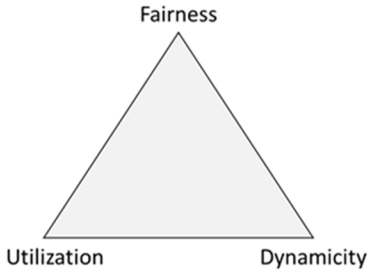

Figure III-1: Fairness, Utilization and Dynamicity of a Scheduling System

A. *Utilization*

One function of a scheduler is to keep the CPU cores busy and utilized. Tasks enter the scheduling queue to be scheduled, and a core or several cores is assigned to the tasks by the scheduler. At a granular level, a scheduler allows sharing of CPU time by the users. The performance of a scheduler is determined primarily by the inverse of CPU idle time.

B. *Dynamicity*

Large computing systems are dynamic in that environmental failures take place, processes fail, new users or new requests enter the system, etc. [19, 20]. A scheduler's effectiveness in managing a dynamic environment is critical in reactive scheduling schemes [21], and ignored in predictive scheduling schemes [22].

C. *Fairness*

Fairness is not well defined in the literature for scheduling systems. It is stipulated that schedulers must "seem fair", and that has translated to long-term fairness across all the users and all the tasks [17]. This creates a temporal starvation for some users as we showed in [23], which may not be desirable. In essence, short-term fairness at times is sacrificed to achieve long-term equality [24]. For scheduling systems, this was demonstrated for fair-share scheduling mechanisms used often in High Performance Computing systems [23].

IV. FUD TRADEOFFS

In this section, we will delve into the specifics of the proposed FUD theorem. We will cover the various combinations of the three parameters in question, and delve into three major cases of optimization. First, however, we will cover the basic cases of optimization shown in TABLE IV-1.

TABLE IV-1: BASIC CASES OF OPTIMIZATION

Optimization of Utilization	Optimizing for Dynamicity	Optimizing for Fairness	Use Cases and Implementation
Maximized	Ignored	Ignored	Predictive scheduling systems. Resource reservation model. Tight control over system parameters required
Ignored	**Maximized**	Ignored	Primary model for reactive scheduling models. Makes no assumption about system parameters or any future state
Ignored	Ignored	**Maximized**	None present in HPC, and more applicable in social justice models as outlined in [25]

In the case where the only concern is utilization, an effective scheduler is predictive [22] with no regards to dynamic aspects of the resources. The number of tasks along with their duration is known in advance, and scheduling is more resource management as typically seen in manufacturing.

In the case where dealing with dynamicity of the overall system is critical and must be optimized, a scheduler is concerned with proper collection of system events such that an accurate state-of-the-system is known, and decisions are made informatively. This type of scheduler is typically seen in reactive scheduling models such as First-Come-First-Served scheduling, network and web load-balancers, and generally scenarios where the current state of the system is the most critical in deciding on the subsequent scheduling decision.

Fairness as a single-parameter optimization is often seen in social justice situations where a common resource (such as drinking water) must be accessible and used by the public. The resource is available, but not necessarily consumed by everyone at any given point, and its proper usage (i.e. utilization) is not monitored. The primary concern here is that drinking water is available and accessible. The water supplier acts as the resource manager to ensure that water is available to all people in the area.

V. FAIRNESS IN RESOURCE MANAGEMENT

In social justice and welfare economics the concept of fairness has been studied from Aristotle's equity principle [26] to John Nash [27] and Rawls [25], and the classic utilitarian definition of resource allocation (see [24]):

- Utilitarian: allocation based on the highest utility or satisfaction from the system
- Aristotle's equity principle: allocation based on some pre-existing claims
- Rawlsian justice: allocation to the users that are the least well off as it guarantees the highest-level increase in satisfaction and utility
- Nash's bargaining theory: resource re-allocation is justified if the gainer increases its utility by a higher percentage than the decrease in loser's utility [28]

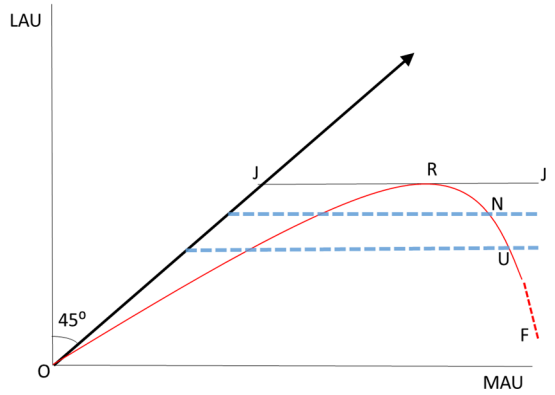

Figure IV-1: Fairness Optimization: (R)awlsian, (N)ash, (U)tilitarian and (F)eudal point models (Recreated from [1])

The utilitarian is criticized to be unethical [28] in that the goal is to maximize system utility, and as a result, the

utility of some of the participants suffer [24]. Utilitarian method is however the method followed by many scheduling systems aiming to increase the overall system utilization. Aristotle's principle is not relevant in our case as system resources are not pre-assigned. Pre-assigning resources, however, has been used in many scheduling systems as demonstrated in [29, 30].

The two remaining views on fairness, Rawlsian and Nash, represent a model that can be used to model dynamic systems [31]. In an HPC environment, at any point in time, 'a' user is the least advantaged user (LAU), and 'a' user is a most advantaged user (MAU) as per Rawlsian model of justice [1] (Figure IV-1).

In a two-user scenario where one is LAU and the other is MAU, the equitable distribution of resources is the 45□ line that separates the two, and the red line is how resources are distributed. The red OF line favors MAU, but the degree to which that takes place is how fairness is determined. Parallel JJ lines to the MAU represent the highest point for equitable distribution for LAU and MAU. ON point in the graph represent the Nash point, where the product of resource allocation is maximized. OU is the utilitarian point, where the sum of individual resource allocation is maximized. OF is the unfair point in the graph where the LAU will continue to starved for resources [1, 25].

VI. Relaxing on Optimizing for Fairness

The four options of fairness mentioned prior need to be applied in context to the problem at hand. Abstractly speaking, if a user is consuming the resources for several hours, while another, smaller user, waits, the fairness models can be applied as such:

- Utilitarian: we could wait for the long running user to finish. In this case, we are interested in maximizing the satisfaction of the individual, and a long running user is preferred over a smaller user.
- Nash: We could pre-empt the long running user. In this case, we are interested in maximizing the overall satisfaction, and want to ensure that both users are satisfied. We care about satisfaction of the long running user more, however, as we will resume the long running user at a later point.
- Rawlsian: We could stop the long running user in favor of the smaller user, as we believe that the smaller is less advantaged, and is being unfairly treated.
- Aristotle: The long running user owns the resources, and has the right to use as needed.

These scenarios are all valid and can be argued to be fair. Within the context of the problem it would make sense to use one of the options over another. Each option has consequences over the two other pillars of scheduling and resource management: utilization and dynamicity. We will ignore the model where the user has pre-existing claims, as that does not fall into the shared computing environment. The other three fairness models effects utilization and dynamicity as depicted in TABLE VI-1.

In the Utilitarian fairness model, the MAU takes much of the resources, and fairness objective is met. Resources are utilized with very little incentive to change resource allocation. The Nash model represents the in-between case as shown in Figure IV-1. Nash's model requires bargaining to take place between the two users (LAU and MAU). Depending on system parameters, if utilization is high, it is possible that less bargaining is taking place, and thus dynamicity is low, and vice versa. In the Rawlsian fairness model, resources are split between MAU and LAU user[s]. Scheduler needs to track the latest LAU, and make decisions frequently. Utilization suffers as continuous change would potentially mean restarting an already running job.

TABLE VI-1: EFFECTS OF VARIOUS FAIRNESS MODELS ON UTILIZATION AND DYNAMICITY

Fairness Model	Utilization	Dynamicity
Utilitarian	High	Low
Nash	High-to-Low	Low-to-High
Rawlsian	Low	High

VII. Relaxing on Optimizing for Utilization

We are interested in utilization in the absolute sense in that utilization must lead to completion of a task. If a task needs to be restarted from the beginning, the processing time wasted is counted towards under-utilization of the environment. If a task is 1-hour long, and it was killed after 55-minutes of processing time to give priority to another task, the wasted 55-minutes counts towards under-utilization of the environment. Furthermore, utilization can be cardinal or ordinal in nature: 52% utilized, 93% utilized; or high utilization, medium utilization, etc. In this paper, we are interested in ordinal nature of utilization.

Targeting a desired utilization can affect other system parameters as shown in TABLE VII-1. It is important to note that fairness is not absolute in that we can consider a utilitarian model of fairness to be fair. The question here is then determining the desired utilization vis-à-vis fairness model chosen. To that end, we can keep fairness a relative parameter, and focus on utilization in the absolute sense.

TABLE VII-1: EFFECTS OF DESIRED UTILIZATION ON FAIRNESS AND DYNAMICITY

Utilization	Fairness	Dynamicity
Low (I)	High	High
Medium (II)	Medium	Medium-High
High (III)	Low	High
High (IV)	High	Low

If the fairness model is Rawlsian, then for us to achieve a high fairness target, we need to either compromise on utilization or dynamicity (I and IV). This is because Rawlsian fairness aims to optimize for the LAU, and finding the current LAU requires us to either be highly dynamic or inefficient. However, if we aim for Rawlsian fairness, but are interested in a higher utilization (II), we may be able to compromise on how we meet our fairness objectives by relaxing how often we check for a new LAU, which may be acceptable in some environments.

If we are willing to compromise on fairness related to LAU, in order to have high marks on the other two (Dynamicity and Utilization), we can employ a proportional fairness scheme [32], which can achieve high utilization in dynamic environments. If we are looking at a utilitarian fairness model, where a proportional fair scheduling can achieve our fairness criteria, then it is possible for us to achieve high utilization marks as shown in [32]. Nash's fairness model requires bargaining between two users, and thus requires a high degree of dynamicity. This will in turn lower our utilization as it will require us to change the order in which tasks are being executed.

VIII. Relaxing on Optimizing for Dynamicity

Dynamicity speaks to dealing with changes that occur while the system is in operation. A scheduler's ability to deal with a dynamic environment can span from its ability to deal with varying numbers of users, tasks, runtime parameters to many others. Whereas our current fairness model is composed of four different types, and utilization is limited to a set of ordinal parameters, dynamicity can be an endless set of parameters and system configurations.

The accuracy of an outcome depends on the parameters used to influence that outcome. For a scheduler to make informed decisions, it can depend on an undetermined number of parameters. The caveat is that the longer the list of parameters, the lesser the ability of the scheduler to deal with changes in said parameters, and thus the lower its level of dynamicity. This is purely because keeping track of a parameter costs resources, which can quickly become scarce as the number of parameters increase.

Abstractly speaking, however, one can assume an ordinal value for dynamicity as well, with the intention that it is relative to the number of parameters being watched for changes.

Table VIII-1: EFFECTS OF DESIRED DYNAMICITY ON FAIRNESS AND UTILIZATION

Dynamicity	Fairness	Utilization
Low	High	High
High	Low	High
High	High	Low

With a low dynamicity, we either have few parameters that change, or we are considering fewer parameters as part of our decision making. With either case, we can increase our utilization and fairness considering that system parameters under consideration do not change often (Table VIII-1).

If we are interested in being able to deal with a high degree of dynamicity, we should lower our expectation for utilization. This is because a dynamic environment requires validation and re-evaluation of LAU and MAU, which would in turn affect utilization. If we are interested in a high degree of utilization, we cannot be shuffling LAU and MAU as that affects utilization, which in turn causes a lower degree of fairness.

IX. Conclusion and Future Research

The FUD theorem introduced in this paper outlined the optimization model for scheduling systems in shared computing environments. We argued that scheduling systems need to optimize for fairness of access for all users, increase and maintain high utilization and deal with the dynamicity of the environment. These three parameters, we argued, cannot all be optimized simultaneously. We further argued that the optimization of one of the parameters has a direct effect on at least one of other parameters. We took a special look at fairness in shared systems as the number of users increases concurrently with the prevalence of shared systems such as Cloud and High Performance Computing. Further study is needed to elaborate on the correlation of these parameters vis-à-vis desired optimization scheme for a given scheduler.

References

[1] J. Rawls, *Justice as fairness: A restatement*: Harvard University Press, 2001.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A view of cloud computing," *Communications of the ACM,* vol. 53, pp. 50-58, 2010.

[3] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing—The business perspective," *Decision support systems,* vol. 51, pp. 176-189, 2011.

[4] D. M. Smith, "Hype cycle for cloud computing, 2011," *Gartner Inc., Stamford,* vol. 71, 2011.

[5] C. Evangelinos and C. Hill, "Cloud computing for parallel scientific HPC applications: Feasibility of running coupled atmosphere-ocean climate models on Amazon's EC2," *ratio,* vol. 2, pp. 2-34, 2008.

[6] E. Angel, E. Bampis, and F. Pascual, "Truthful algorithms for scheduling selfish tasks on parallel machines," *Theoretical Computer Science,* vol. 369, pp. 157-168, 2006.

[7] G. Christodoulou, L. Gourves, and F. Pascual, "Scheduling selfish tasks: about the performance of truthful algorithms," in *Computing and Combinatorics*, ed: Springer, 2007, pp. 187-197.

[8] H. N. Van, F. D. Tran, and J.-M. Menaud, "SLA-aware virtual resource management for cloud infrastructures," in *Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on*, 2009, pp. 357-362.

[9] A. G. García, I. B. Espert, and V. H. García, "SLA-driven dynamic cloud resource management," *Future Generation Computer Systems,* vol. 31, pp. 1-11, 2014.

[10] L. Tripathy and R. R. Patra, "Scheduling in cloud computing," *International Journal on Cloud Computing: Services and Architecture (IJCCSA),* vol. 4, pp. 21-7, 2014.

[11] S. Selvarani and G. S. Sadhasivam, "Improved cost-based algorithm for task scheduling in cloud computing," in *Computational intelligence and computing research (iccic), 2010 ieee international conference on*, 2010, pp. 1-5.

[12] Y. Deng, P. Zhang, C. Marques, R. Powell, and L. Zhang, "Analysis of Linpack and power efficiencies of the world's TOP500 supercomputers," *Parallel Computing,* vol. 39, pp. 271-279, 2013.

[13] G. P. Rodrigo, P. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan, "Towards understanding HPC users and systems: a NERSC case study," 2017.

[14] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of scheduling*: Courier Corporation, 2012.

[15] K. R. Baker and D. Trietsch, *Principles of sequencing and scheduling*: John Wiley & Sons, 2013.

[16] V. T'kindt and J.-C. Billaut, *Multicriteria scheduling: theory, models and algorithms*: Springer Science & Business Media, 2006.

[17] J. Kay and P. Lauder, "A fair share scheduler," *Communications of the ACM,* vol. 31, pp. 44-55, 1988.

[18] H. Hussain, S. U. R. Malik, A. Hameed, S. U. Khan, G. Bickler, N. Min-Allah, M. B. Qureshi, L. Zhang, W. Yongji, and N. Ghani, "A survey on resource allocation in high performance distributed computing systems," *Parallel Computing,* vol. 39, pp. 709-736, 2013.

[19] C. Boneti, R. Gioiosa, F. J. Cazorla, and M. Valero, "A dynamic scheduler for balancing HPC applications," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008, p. 41.

[20] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of scheduling,* vol. 12, pp. 417-431, 2009.

[21] V. Suresh and D. Chaudhuri, "Dynamic scheduling—a survey of research," *International journal of production economics,* vol. 32, pp. 53-63, 1993.

[22] P. Burke and P. Prosser, "A distributed asynchronous system for predictive and reactive scheduling," *Artificial Intelligence in Engineering,* vol. 6, pp. 106-124, 1991.

[23] A. Sedighi, Y. Deng, and P. Zhang, "Fariness of task scheduling in high performance computing environments," *Scalable Computing: Practice and Experience,* vol. 15, pp. 273-285, 2014.

[24] D. Bertsimas, V. F. Farias, and N. Trichakis, "The price of fairness," *Operations Research,* vol. 59, pp. 17-31, 2011.

[25] J. Rawls, *A theory of justice*: Harvard university press, 2009.

[26] R. A. Shiner, "Aristotle's Theory of Equity," *Loy. LAL Rev.,* vol. 27, p. 1245, 1993.

[27] J. F. Nash Jr, "The bargaining problem," *Econometrica: Journal of the Econometric Society,* pp. 155-162, 1950.

[28] H. P. Young, *Equity: in theory and practice*: Princeton University Press, 1995.

[29] M. J. Bach, *The design of the UNIX operating system* vol. 5: Prentice-Hall Englewood Cliffs, NJ, 1986.

[30] G. J. Henry, "The unix system: the fair share scheduler," *AT&T Bell Laboratories Technical Journal,* vol. 63, pp. 1845-1857, 1984.

[31] L. v. Bertalanffy, *General system theory; foundations, development, applications*. New York,: G. Braziller, 1969.

[32] H. V. Bui, *Fairshare scheduling-a case study*: University of Arkansas, 2008.