2009 International Conference on Parallel Processing Workshops

The Owner Share scheduler for a distributed system

José Nelson Falavinha Junior*, Aleardo Manacero Júnior†, Miron Livny‡ and Daniel Bradley‡
*Department of Electrical Engineering
São Paulo State University - UNESP, Ilha Solteira, São Paulo - Brazil
Email: junior.falavinha@gmail.com
†Department of Computer Science
São Paulo State University - UNESP, São José do Rio Preto, São Paulo - Brazil
Email: aleardo@ibilce.unesp.br
‡Department of Computer Science
University of Wisconsin, Madison, Wisconsin - USA
Email: miron@cs.wisc.edu / dan@hep.wisc.edu

***Abstract*—In large distributed systems, where shared resources are owned by distinct entities, there is a need to reflect resource ownership in resource allocation. An appropriate resource management system should guarantee that resource's owners have access to a share of resources proportional to the share they provide. In order to achieve that some policies can be used for revoking access to resources currently used by other users. In this paper, a scheduling policy based in the concept of distributed ownership is introduced called Owner Share Enforcement Policy (OSEP). OSEP goal is to guarantee that owner do not have their jobs postponed for longer periods of time. We evaluate the results achieved with the application of this policy using metrics that describe policy violation, loss of capacity, policy cost and user satisfaction in environments with and without job checkpointing. We also evaluate and compare the OSEP policy with the Fair-Share policy, and from these results it is possible to capture the trade-offs from different ways to achieve fairness based on the user satisfaction.**

***Keywords*-Scheduling Algorithms; Fair Scheduling; Distribute systems; Ownership concept;**

I. Introduction

The cost/performance ratio of computational systems leads to distributed ownership [1] since groups of users who own hardware are willing to share it if they can utilize hardware owned by others. Once an infrastructure where the resources are owned by distinct entities is created, a common goal is to share resources to make the best use of the infrastructure [2].

Resource sharing represents a good method to make use of the processing capacity and data storage of a computational system. An important requirement in this situation is that resource owners must be able to access their share of the resources whenever they want to. There are two methods to define an owner's share of the resources. The first method is the precise set of resources owned by the user. The second method does not require an exact set, but an equivalent set of resources. The choice between the two definitions is a matter of policy. The second model is aimed at providing a configurable scheduling infrastructure that uses equivalent resources to achieve the owner share.

A scheduling policy, called the Owner Share Enforcement Policy (OSEP), is defined and applied in a preemptive space-sharing approach [3], [4], using job preemption to achieve the owner's proportion of resources. We implemented this in Condor, which supports job preemption [5], [6]. OSEP dynamically adjusts the amount of allocated resources for users according to the amount of resources they provide. OSEP also enables the use of checkpointing, that is the completion of a preempted job from the state in which it was interrupted.

Unlike the Fair-Share scheduler [7] which takes into account the user's historical usage, OSEP uses instantaneous information about the owners' share of contributed resources as the main factor to achieve fairness.

In the remainder of the paper we describe OSEP in detail. We present several performance metrics, then evaluate our implementation of OSEP through several runs over an actual system. We also present results of comparisons between the OSEP and the Fair-Share policies. Our analysis compares scenarios both with and without checkpointing.

II. The Owner-Share Enforcement Policy

The OSEP policy is based on the resource ownership of a distributed system. In order to fully understand its operation, it is necessary to define a few terms, which is done in the following paragraphs.

Distributed ownership is the term used to express the multiple ownership claims in aggregated infrastructures composed of several parts with independent administrative domains, with independent policies, but with some common purpose, such as a grid [8].

The resource ownership is an important factor to be considered by a scheduler and by a resource allocation policy. The scheduler can allocate resources to all users in the system, but the resource owners must be able to access their share of the resources when needed. Based on this fact, we created and evaluated a scheduling enforcement policy called Owner-Share Enforcement Policy (OSEP) as a method to guarantee that each user/owner can have his/her appropriate share of resources when he needs it.

Owner Share is a parameter that measures how much of the system resources each user can use and how much he/she provided.

Consider now a cluster with n machines used by m users, $U = \{u_1,...,u_m\}$ with different amounts of jobs. Then, the owner share is defined by:

Definition 1: The Owner Share of resources, denoted by a_i, is the number of nodes n_i that the user u_i provided to the infrastructure, where if n is the total number of nodes in the system, then $\sum_i a_i = n$.

Feasible share of resources (fes_i), similarly to the concept presented in [4] for the fair-share policy, determines the exact number of resources that should be allocated to each user, considering his/her demand and owner share, and the unused nodes by users that have $d_i < a_i$. Its value is determined by:

Definition 2: The parameter fes_i[1] can be determined by the equation $fes_i \equiv min\{(a_i + \hat{a}_i), d_i\}$, where:
- a_i is the Owner-Share of nodes that u_i is entitled to;
- $\hat{a}_i$ is the excess share of nodes allocated to u_i;
- d_i is the total demand for resources of u_i;
- $\sum_i fes_i = n$.

A. OSEP algorithm

The algorithm responsible for enforcing the OSEP works in a preemptive space sharing approach for a distributed system. It is aimed to adjust the number of running jobs or allocated machines per user/owner in order to reach the owner's share of resources.

In the Owner-Share enforcement policy users must have a number of allocated resources equivalent to the minimum value between their demand and their owner share, or as many machines as possible if there are extra available machines. In order to implement OSEP, the scheduling algorithm is divided in two main parts: one responsible for its dynamic behavior and another that is responsible for analyzing the user information about allocated resources and making preemption decisions according to OSEP.

1) Dynamic Algorithm: The Owner-Share enforcement system needs updated data about resource usage in order to make the appropriate decisions. This data is provided by different mechanisms in different environments, such as by ClassAds in Condor. With this mechanism jobs can effectively state their requirements and preferences and, similarly, machines can specify requirements and preferences about the jobs they are willing to run. We use Condor's ClassAds to provide the data needed by OSEP.

The *Dynamic Algorithm* considers the variables p and t, where p is the maximum number of jobs to be preempted in each iteration, and t is a pre-defined iteration interval of the algorithm.

Dynamic Algorithm

```
1. Every $t$ seconds, the updater system does:
2.   For each job in the queue of the Scheduler {
3.     Load the job resource usage data (ClassAd);
4.     Read the owner and job status information;
5.     Update the owner/user ClassAd; }
6.   Update $p$;
7.   Call Decision Algorithm($p$);
```

In this algorithm the main work occurs in line 5, where the information about the needs of each owner/user is updated. This means calculating how many resources are needed and how many must be reallocated. The actual reorganization of job allocation is performed by the *Decision Algorithm*, describe below.

2) Decision Algorithm: The *Decision Algorithm* is called from the previous one. It decides whether a job is preempted or not based on information stored in the user ClassAds. From this information a job can be considered **Active** (in execution), **Idle** (ready to be allocated) or on **Hold** (blocked by the user or the scheduler). In every iteration, the algorithm looks for jobs in the ***Idle*** state and negotiates the job allocation by the preemption of a currently ***Active*** job.

The choice of which job is preempted is made by comparing wall clock times of all jobs of a given user. The decision rule selects the job with the lowest wall clock (minimizing the loss of processing cycles). The preempted job must be owned by the user with the largest excess share.

The execution of the *Decision Algorithm* depends on parameters of a given user such as the number of idle jobs (i_{user}) and the difference between his/her share and the number of active jobs, $f_{user} = a_{user} - n\_ajobs_{user}$, where n_ajobs_{user} is the current numbers of active jobs of the user. It also needs a few parameters from every job of a given user, such as the wall clock time (w_{job}), and its status (s_{job}). The following algorithm operates on a cluster of m users. The variable p holds the maximum number of jobs to be preempted at each iteration.

[1]It is important to explain that when the demand d_i of the user u_i can not be attended, there is no excess share of nodes and then $\hat{a}_i$ is zero.

Decision Algorithm (int p)

```
1. If $(p > 0 \wedge (\exists\ u_{\max} \mid f_{\max} = max(f_1, ..., f_m)$
                         $\wedge\ f_{\max} > 0 \wedge i_{\max} > 0))$
2.   { If $(\exists\ u_{\min} \mid f_{\min} = min(f_1, ..., f_m) \wedge f_{\min} < 0)$
3.     { If $(\exists$ job $j \in u_{\min} \mid w_j = min(w_1, ..., w_l)$
                         $\wedge\ (s_j =$ **Active**$)$
4.       { ***vacate_ job***$(j)$;
5.         Reserve resource for the idle job $k$ from $u_{\max}$;
6.         Decrease $p$;
7.         Update the ClassAds of users $u_{\max}$, $u_{\min}$,
            and jobs $j$ and $k$;
8.       }
9.       Call ***Decision Algorithm***$(p)$;
10.    }
11. }
```

In line 1, the Decision Algorithm selects the user with the largest deficit between his/her share and his/her current number of active jobs. This user is the one that will receive the preempted resource at the current iteration. In the second line it selects the user with the largest amount of executing jobs over the share. This user will have one of his/her jobs preempted. The job that will be preempted is determined as the one with the lowest wallclock time. After a job preemption, the preempted job is put back in the queue, being completely reexecuted if checkpointing is not used. The steps on lines 4 through 8 perform the actual job termination and resource reservation. This algorithm is called recursively until the OSEP is achieved. The resource alocation is done by a separate process that runs between iterations of the OSEP algorithm.

III. Performance metrics

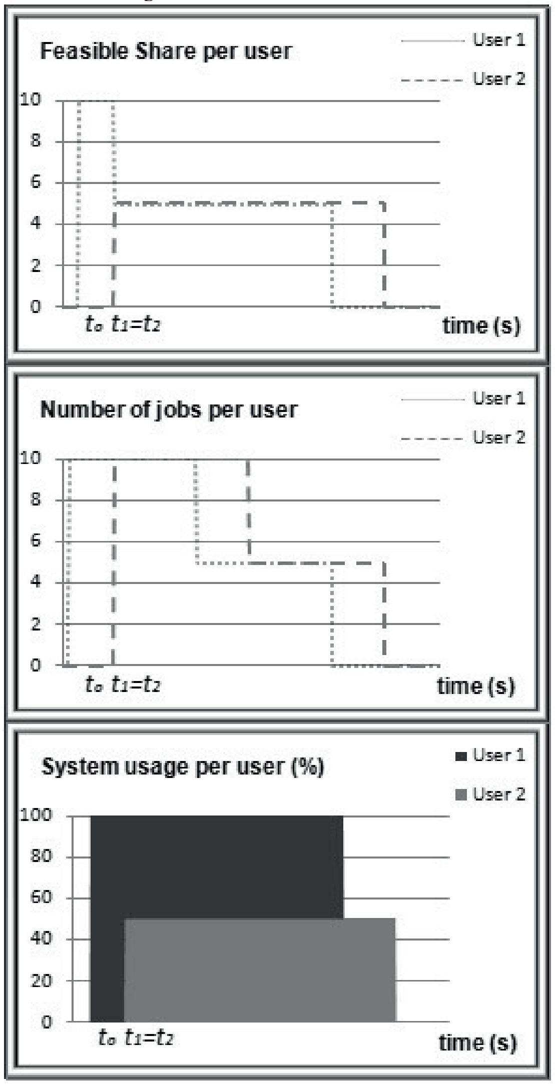
The performance metrics used to analyze the behavior of the enforcement policy are related to the utilization of the resources and the accomplishment of the Owner-Share enforcement policy based on its ideal case. It is important to remember that the infrastructure supports different configurations based on the owner's needs. Once the owner's needs and the ideal case are defined, the infrastructure can be configured in order to reach the ideal behavior as close as possible.

Therefore, in order to analyze, compare and measure the performance metrics, we defined the ideal case described in the following subsection. Some concepts and rules also need to be defined in order to describe the performance metrics.

A. Ideal case for OSEP

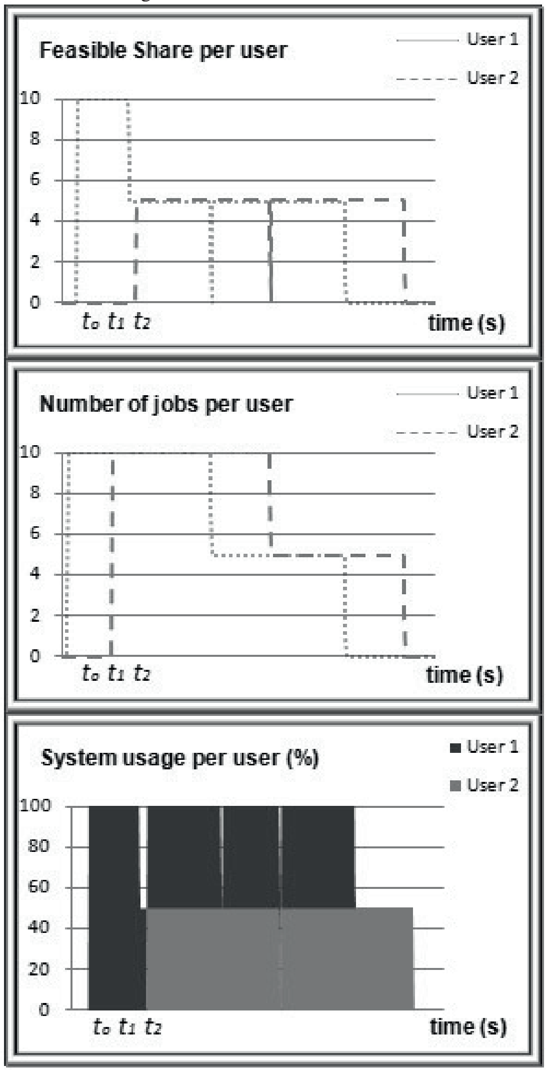
The ideal case for OSEP is a definition used to measure the effect of the policy actions over the system's performance. Different idealized goals can be established, according to the needs of the users. The general case adopts the notion that every user wants all of his/her resources as soon as the jobs are submitted. This is the level adopted throughout the tests.

Figure 1. Ideal situation for OSEP



As an example, consider an environment with 10 available nodes, and two users, u_1 and u_2, with the same owner share of resources, 5. The first user has 10 active jobs and no idle jobs at time t_0, and at t_1 the user u_2 submits 10 jobs. Before t_1 there was just one user in the system, u_1, and he had the right to use all resources. After t_1, both users must share the resources. In the ideal case they will have a feasible share of 5 (fes_1= fes_2=5) and no excess share ($\hat{a}_1$=$\hat{a}_2$=0). In an ideal case the system should satisfy the owner-share policy instantaneously. However, in real systems it would take $(t_2 - t_1)$ seconds to reach the Owner-Share policy where $t_2 > t_1$. Figures 1 and 2 show the plots of the system usage in an ideal situation and in a real system with the Owner-Share enforcement policy (OSEP).

Figure 2. Real situation for OSEP



B. Policy Violation

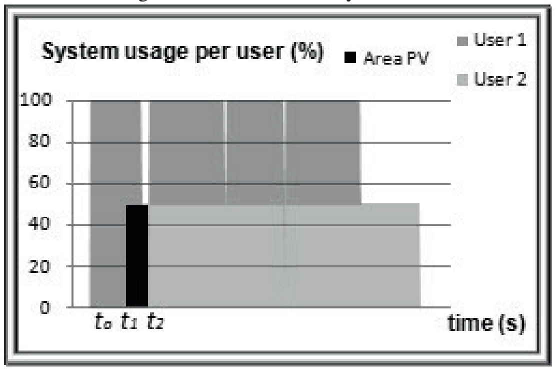
Policy Violation (PV) measures the latency introduced by the real system when compared to the ideal situation. Consider the previous example again, and the plots presented at figure 2. These plots results from the OSEP application in a real distributed system.

Formally the Policy Violation is defined as the area in the system usage plot where one user should have his/her share but does not have due to the policy latency, as shown at Figure 3. It measures the time spent by OSEP to reach the appropriate system utilization, being calculated by:

- $PV = \int_{t_1}^{t_2} SuI_i - SuR_i$ where:
  - SuI_i is the curve of the system utilization for the user u_i in the ideal case and;
  - SuR_i is the curve of the system utilization for the user u_i in the real case and.

The Policy Violation is calculated only for users who have demand but do not have their share of resources yet.

Figure 3. Area for Policy Violation

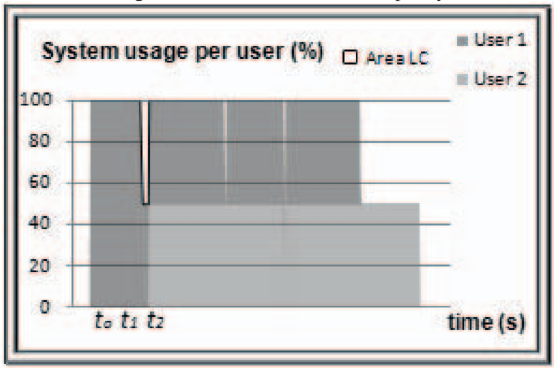


C. Loss of Capacity

The second metric is the Loss of Capacity (LC), also called Capacity Loss in [9]. It defines the amount of the system utilization that was lost due to the actions of the enforcement algorithm. During the interval between job preemption and allocation of other jobs the resources remain idle. This is defined as a lack of utilization and measures computing cycles wasted by the policy. In practical terms it can be calculated by the area where system utilization is under 100%. Figure 4 shows the Loss of Capacity between t_1 and t_2. It is calculated only for users who actually have excess resources by:

- $LC = \int_{t_1}^{t_2} SuI_i - SuR_i$ where:
  - SuI_i is the curve of the system utilization for the user u_i in the ideal case and;
  - SuR_i is the curve of the system utilization for the user u_i in the real case and.

Figure 4. Area for Loss of Capacity



D. Policy cost

The policy cost (PC) measures the overhead caused by OSEP in the system or the waste of CPU cycles by job reprocessing. It is defined by the difference between the areas of system usage in the real system and in the ideal case, as in:

- $PC = \int_{t_{0_r}}^{t_{f_r}} SuR - \int_{t_{0_i}}^{t_{f_i}} SuI$

The PC value may be affected by the non-distributed load of individual resources, that is, loads that are not allocated by OSEP could delay the execution of OSEP jobs, increasing PC value. Besides that it is an important metric to evaluate the impact of checkpointing in OSEP application.

E. User satisfaction

The user satisfaction (US) is a performance metric that shows how satisfied the user is with the system. Satisfaction is something complex to define since a lot of factors can be used. Quality, fast response, and delivery time are the most common factors to measure the user/customer satisfaction in any kind of system.

The main point to be analyzed in order to achieve 100% satisfaction is the user/owner need. In some cases, it is better for the user to wait and get more resources later than to get all the share of resources at the moment, depending on the system workload. Therefore, the user satisfaction metric must be based on the ideal behaviour of the system that is expected by the user, independent of the system workload. Based on this fact we define the ideal case for the user satisfaction.

Definition 3: The ideal case of satisfaction for a user occurs when gets all resources needed to execute the jobs and they finish at the expected moment, no matter what. The parameter that measures the user satisfaction is the delivery time of the job, so the user satisfaction US_{i_j} is calculated for each job j and the average of these values represented by A_US_i determine how satisfied the user i is with the system, where:

- $US_{i_j} = ((t_{j_{ideal}} - t_{j_0})/(t_{j_{real}} - t_{j_0}))x100$;
- $A\_US_i = \sum_j US_{i_j}/n$ where n is the number of jobs of the user i.

IV. Tests and results

Several tests were conducted to evaluate the influence and performance of OSEP, when applied to different configurations of the infrastructure and workload. The tests provided benchmarks to evaluate the policy against the defined performance metrics, that is, the Policy Violation, Loss of Capacity, Policy cost, and User Satisfaction.

We benchmarked a pool of machines distributed along several departments of the University of Wisconsin in Madison using the Condor System. Therefore, the results reflect a real testbed for the OSEP analysis, where resource owners from different departments may want their Owner-Share of resources. The machines are distributed in multiple clusters and they run distributions of UNIX operating systems.

The evaluation environment created was a system with several resources from different departments and users, where the job submission is controlled by a centralized Condor. We nominated a share of resources for each user and then, they start to submit jobs to the system.

The main situation to be analysed occurs when a user/owner u_1 submits, at certain time t_0, k jobs to the system, and a second user/owner u_2 submits l jobs to the same system at time t_1, where $t_1 > t_0$. If k and l are bigger than the number of resources equivalent to the corresponding user share, it creates a dispute over the resources among the users.

The system starts to enforce the OSEP policy when the number of resources alocated to some user is smaller than his/her share and if there is at least one user with extra resources. In order to reduce the loss of CPU cycles due to job reprocessing, we minimized the iteration interval between two OSEP activations. The definition of this interval is dependent on the OSEP policy implementation. As we implemented it through the Condor system, we define the iteration interval according to Condor's negotiation interval.

A. Evaluating the impact of environment variations

The tests were conducted by several cases involving the main system variables, such as the number of resources in the system, the number of jobs per user, the size of the jobs and the number of users and their shares.

The results for all tests showed that the best and less expensive way to achieve the OSEP policy is to give to the user all his/her share in just one iteration, that is, allocating all viable tasks in a single invocation of the Decision Algorithm. This means that the variable p used at the OSEP algorithm must receive the value of the highest share among the users.

One important result achieved with OSEP is user satisfaction of the owners. A major problem with the well known FCFS (First Come, First Served) policy, as well as with the Fair-Share policy, is that they give a higher priority to the first user that submits jobs. On the other hand, the OSEP policy enforces the reallocation of resources according to the user share, making the user satisfaction proportional to their share.

Consider the case represented by figure 5 and table I, where $user$ 1 and $user$ 2 have the same share of resources (50% for each one). Also $user$ 1 submits jobs that execute for 3600 seconds each at $t_1 = 0s$, and $user$ 2 submits jobs that execute only for 600 seconds each at moment $t_2 = 300s$. The graphs at figure 5 show the system utilization. OSEP policy applied with the checkpointing capability is represented by the symbol Θ in table I.

Figure 5. Variation of the job size

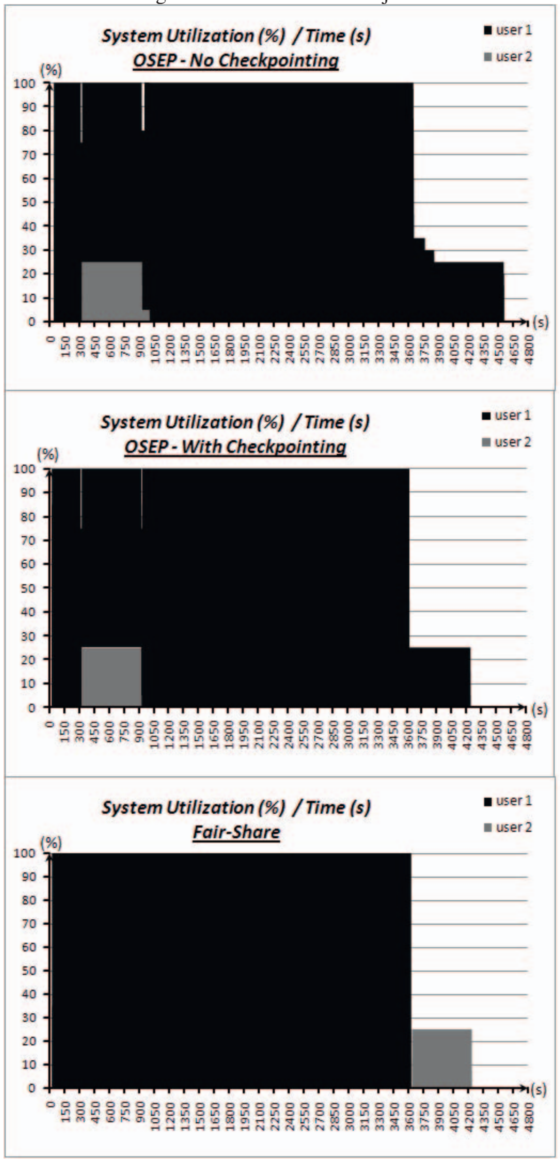


Table I
VARIATION OF THE JOB SIZE

	OSEP	OSEP Θ	Fair-Share
PV	600	550	-
LC	275	195	-
PC	9845	625	-
A_US_1	93.21%	95.66%	99.26%
A_US_2	95.63%	97.28%	21.24%

The difference among the results for the OSEP with and without the checkpointing capability is clear. All the metrics, Policy Violation (PV), Loss of Capacity (LC) and Policy Cost (PC) are reduced with the checkpointing capability since all the work done before the preemption of jobs is not lost. The Fair-Share policy causes a lower satisfaction level for the $user$ 2 while the OSEP gets good satisfaction for both users (more than 90%).

B. Evaluating the impact of several users and different shares

This section presents a more general and realistic case. It defines 4 users with different shares and jobs of several sizes submitted at different moments. Size restrictions limited us to a single example of the tested environments. This case is described as:

- System with 20 resources;
- Different Shares: user 1 with 30%, user 2 with 40% and user 3 and 4 with 15% each;
- 15 jobs per user, job size ranging from 600 to 3600 seconds each;
- t_1 =0s, t_2 =100s, t_3 =200s e t_4 =210s, where t_x is the moment when $user\ x$ submits his/her jobs;

Figure 6 shows the graphs of system utilization resulting from this test while table II presents the results for the performance metrics and user satisfaction. The results show that the OSEP policy guarantees a user satisfaction proportional to the user share, meaning that the bigger the user share, the bigger will be the user satisfaction. This explains the large difference for the $user$ 2 between the OSEP and Fair-Share policies (about 20% of satisfaction). As said before, the Fair-Share policy favours the users who submitted their jobs first, so the $user$ 1 is the most satisfied one.

Table II
GENERAL CASE

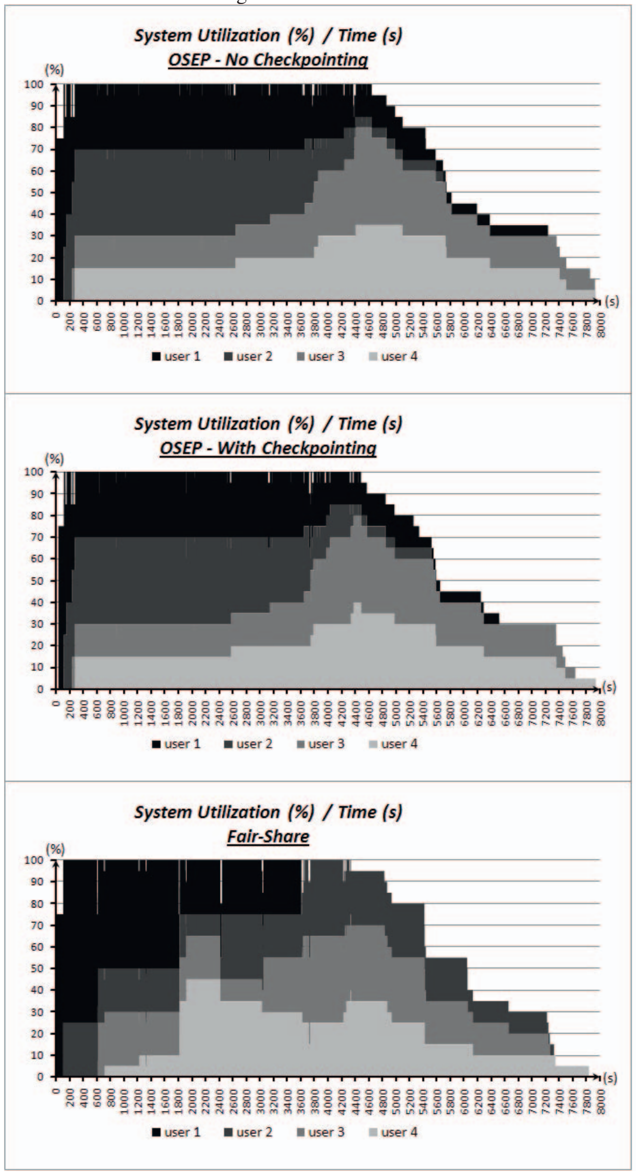
	OSEP	OSEP Θ	Fair-Share
PV	2395	2115	-
LC	915	800	-
PC	18380	790	-
A_US_1	67.18%	70.01%	99.40%
A_US_2	77.02%	78.20%	56.7%
A_US_3	47.98%	50.61%	50.59%
A_US_4	50.02%	50.60%	50.58%

The only situation where OSEP does not guarantee the best user satisfaction levels occurs when one user has a very small share and submits a reasonable amount of jobs. In this case it will get only a small amount of resources if there is dispute with other users. The Fair-Share policy, in this situation, could compensate this user according to the historical usage of the system, providing a better user satisfaction.

V. RELATED WORK

The concept of ownership is introduced from an economic point of view, by Grossman and Hart [10], showing the benefits of ownership and also the costs associated with the wrong allocation of residual rights or shares. Alstyne and others [11] apply ownership to a distributed database design in order to describe its impact on information sharing and system performance.

Figure 6. General Case



Besides these seminal works, most of the scheduling policies and distributed share of resources use the Fair-Share approach. Fair-Share policies try to distribute resources proportionally among users based on historical usage. From its definition, jobs already running are not preempted and users that are postponed are compensated with heavier use at a later moment. Based on this approach Arpaci-Dusseau and Culler [12] introduced a policy where the share is based on a time-sharing technique. In other approach, Kleban and Clearwater [13] use a non-preemptive space-sharing algorithm in order to provide the fair-share policy.

Amar and others [4] use the concept of fair-share to build an algorithm for node allocation in a cluster, where they used preemption in a space-sharing approach. Their algorithm is concerned with addition or subtraction of nodes into the system. Andrade and others [14] introduced the Network of Favours that is an incentive mechanism for p2p systems sharing multiple resources, it uses an accounting information between peers (sites) in order to give or take resources. Also, Wang and Merchant [15] use the idea of fair-share into a distributed algorithm to enforce proportional sharing of storage resources among streams of requests, but this is a context that is not directly concerned with cpu usage.

None of these works use the concepts of owner share and distributed ownership. These concepts are relatively new, since they were better defined with the appearance of computational grids and large resource sharing systems.

VI. Conclusion and Future Work

This paper presented a scheduling infrastructure for the Owner Share scheduler based on the distributed ownership concept. It was implemented through a dynamic enforcement algorithm in a distributed system with a preemptive space-sharing approach, and tested on different scenarios, where job checkpointing is or is not possible.

The infrastructure provides a configurable and flexible framework to analyze and evaluate ownership as the main factor of a scheduling policy. Based on the owner's policies, the system variables can be analyzed and evaluated to determine the best configuration to reach the owner's share of resources according to his/her need.

The results showed that the OSEP policy aims to provide a user satisfaction proportional to the user's share of resources, since it tries to achieve fairness based on the distributed ownership concept. The comparisons with the Fair-Share policy were useful to determine the problems and advantages of each policy. The OSEP policy presents a smaller standard deviation for user satisfaction when compared to the results of the Fair-Share policy.

The results also showed that checkpointing represents a performance improvement for the owner share scheduler, since the waste of CPU cycles related to job preemption is minimized. Moreover, the overhead due to preemption is even lower for long-running jobs, for which preemption is a smaller percentage of the runtime.

In this work, the ownership concept is represented by the number of machines since the infrastructure was considered to be homogeneous machines. For systems with heterogeneous resources the ownership concept can be represented by the resource's processing capacity instead of the number of resources.

Some things can still be done for future work, as the treatment for non owner users, to allocate available resources among non-owner users and owners when every owner already has the owner share. We would like to study and verify how the OSEP and Fair-Share policies could be mixed to create a new policy, more efficient and fair. Also, we aim to include the support of virtual machines and resources to the OSEP system.

Acknowledgment

The authors wish to thank all members of the Condor team for their help. This research was supported in parts by funds from the Coordination for the Improvement of Higher Education - (CAPES - Brazilian government agency) and FAPESP.

References

[1] M. Livny and R. Raman, "High-throughput resource management," in *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds. Morgan Kaufmann, 1998.

[2] M. Lathia, "Advantages of grid computing," *IEEE Distributed Systems Online*, vol. 6, no. 2, p. 5, 2005.

[3] S. W. Turner, L. M. Ni, and B. H. C. Cheng, "Time and/or space sharing in a workstation cluster environment," in *Supercomputing '94: Proceedings of the 1994 ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM, 1994, pp. 630–639.

[4] L. Amar, A. Barak, E. Levy, and M. Okun, "An on-line algorithm for fair-share node allocations in a cluster," in *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 83–91.

[5] A. Roy and M. Livny, "Condor and preemptive resume scheduling," in *Grid resource management: state of the art and future trends*. Norwell, MA, USA: Kluwer Academic Publishers, 2004, pp. 135–144.

[6] M. Litzkow, T. Tannenbaum, J. Basney, and M. Livny, "Checkpoint and migration of UNIX processes in the Condor distributed processing system," University of Wisconsin - Madison Computer Sciences Department, Tech. Rep. UW-CS-TR-1346, April 1997.

[7] J. Kay and P. Lauder, "A fair share scheduler," *Commun. ACM*, vol. 31, no. 1, pp. 44–55, 1988.

[8] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200–222, 2001.

[9] J. F. C. M. de Jongh, "Share scheduling in distributed systems," *PhD Thesis*, 2002. [Online]. Available: citeseer.ist.psu.edu/jongh02share.html

[10] G. S. and O. Hart, "The cost and benefits of ownership: A theory of vertical and lateral integration," in *CEPR Discussion Paper no. 70*, London, Centre for Economic Policy Research, 1985. [Online]. Available: http://www.cepr.org/pubs/dps/DP70.asp

[11] M. V. Alstyne, E. Brynjolfsson, and S. Madnick, "Ownership principles for distributed database design," MIT Center for Coordination Science, Working Paper Series 142, May 1993. [Online]. Available: http://ideas.repec.org/p/wop/mitccs/142.html

[12] A. Arpaci-Dusseau and D. Culler, "Extending proportional-share scheduling to a network of workstations," in *PDPTA'97: International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, USA, 1997. [Online]. Available: http://now.cs.berkeley.edu/Papers2/Postscript/pdpta.ps

[13] S. D. Kleban and S. H. Clearwater, "Fair share on high performance computing systems: What does fair really mean?" in *CCGRID '03: Proceedings of the 3st International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2003, p. 146.

[14] N. Andrade, F. Brasileiro, and M. Mowbray, "Discouraging free riding in a peer-to-peer cpu-sharing grid," in *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC 04), IEEE Computer Society*, 2004, pp. 129–137.

[15] Y. Wang and A. Merchant, "Proportional share scheduling for distributed storage systems," *Proc. 5th USENIX Conference on File and Storage Technologies*, San Jose, CA, February 2007.