Optimizing the Stretch for Virtual Screening Application on Pilot-agent Platforms on Grid/Cloud by using Multi-level Queue-based Scheduling

T. Q. Bui[1,2], E. Medernach[1] , V. Breton[1] and H. Q. Nguyen[2]
[1]*Laboratoire de Physique Corpusculaire, Universite Blaise Pascal, 63171 Aubiere, France*
[2]*Institut de la Francophonie pour l'Informatique, Vietnam National University, Hanoi, Vietnam*

Keywords: Virtual Screening, Grid Computing, Cloud Computing, Scheduling, Fairness, Stretch, Online-algorithm, Multilevel Queue Scheduling.

Abstract: Virtual screening has proven very effective on grid infrastructures. We focus on finding platform scheduling policy for pilot-agent platform shared by many virtual screening users. They need a suitable scheduling algorithm at platform level to ensure a certain fairness between users. Optimal criterion used in our research is the stretch, a measure for user experience on the platform. From our latest research (Quang et al., 2013), simulation result and experimentation on real pilot agent platform showed that SPT policy is the best policy in 4 different existing scheduling policies (FIFO, SPT, LPT and Round Robin) for optimizing the stretch. However, research on real grid workload (Medernach, 2005) showed that there are two types of grid user: normal users who submit frequently little jobs to grid and data challenge users who submit occasionally large number of jobs to grid. And SPT policy, in particularly, is not appropriate for data challenge user because they have to wait always normal user. In this paper, we proposed a new policy named SPT-SPT which uses multi-level queue scheduling technique for scheduling in a pilot agent platform. In SPT-SPT policy, the administrator creates two separate user groups in the platform: Normal group and Data Challenge group. Each group has their own task queue in the platform and SPT policy is applied on it. A parameter p ($p \epsilon [0,1]$), the probability that task queue is chosen to send pilot agent their task, is assigned to one task queue and 1-p for the other one. This policy improves user experience for Data Challenge group and do not impact very much for Normal group.

1 INTRODUCTION

In silico drug discovery (*in silico* means computer-assisted) offers an efficient alternative to reduce the cost of drug development and to speed-up the discovery process. Virtual screening is achieved through a pipeline analysis whose first step requires using a docking software such as Autodock, DOCK or FlexX to predict potential interacting complexes of small molecules in protein binding sites. From now, we use VS to denote virtual screening.

Large scale virtual screening, and especially its docking step, consumes large computing resources. As docking is an embarrassingly parallel process where thousands to millions of compounds are tested in silico against a biological target, it was successfully deployed on grid computing to reduce the computation time. Some large scale VS projects in the past have been deployed successfully on grids such as WISDOM-I (Jacq et al., 2006) and WISDOM-II (Kasam et al., 2009) on malaria or Avian Flu Data Challenge (Jacq et al., 2008).

Pilot-agent platforms are tools used for submitting and controlling a large number of user jobs on grid infrastructures. Several pilot agent platforms have been developed such as WPE (Kasam et al., 2009), DIRAC (van Herwijnen et al., 2003), DIANE (Mościcki, 2003), glideinWMS (Sfiligoi, 2008), PanDA(Maeno, 2008). The DIRAC pilot-agent platform is now available to the users of several multi-disciplinary virtual organizations on EGI (the European Grid Initiative). Because many users share the DIRAC pilot-agent platform, it is important to define a scheduling policy to ensure a certain degree of fairness so that users receive a fair share of system resources. The scheduling policies used on the existing pilot-agent platforms on EGI,

Bui T., Medernach E., Breton V. and Nguyen H..
Optimizing the Stretch for Virtual Screening Application on Pilot-agent Platforms on Grid/Cloud by using Multi-level Queue-based Scheduling.
DOI: 10.5220/0004962801990204
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 199-204
ISBN: 978-989-758-019-2

are respectively First In First Out (FIFO) policy in WPE platform or Round Robin policy in DIRAC platform. The VS project has specific properties such as divisibility in many independent docking tasks, no order of execution constraints and comparable execution time of all docking tasks. In this paper, we focus on evaluating suitable online scheduling policies for the VS application on the pilot-agent platform to improve user's satisfaction in the system.

Our research is also relevant to applications which have the same properties of VS application (divisibility in many independent tasks, no order of execution constraints and comparable execution time of all tasks) on pilot agent platform on grid/cloud. These applications are used in a variety of scenarios, including data mining, massive searches (such as key breaking), parameter sweeps, simulations, fractal calculations, computational biology, and computer imaging.

1.1 Pull Model, 2-Level Scheduling and Limited Machine Availability Property of Scheduling

A pilot-agent platform uses pull model for efficient submission and controlling of user tasks: tasks are no longer pushed through the grid scheduler but are put in a master pool and pulled by pilot agents running on computing nodes. Scheduling job is the process of ordering tasks in this pool. List scheduling is applied in it. The pilot-agent itself is a regular grid job, which is started through a grid resource manager. It is submitted automatically by platform and run on a computing machine on grid. We can see pilot agent as container of jobs. Pull model adapts to heterogeneous property of grid (faster machine will pull more tasks than the other), reduces faults (resubmission of failed tasks) and improves latency (the waiting time of job in grid scheduler is reduced).

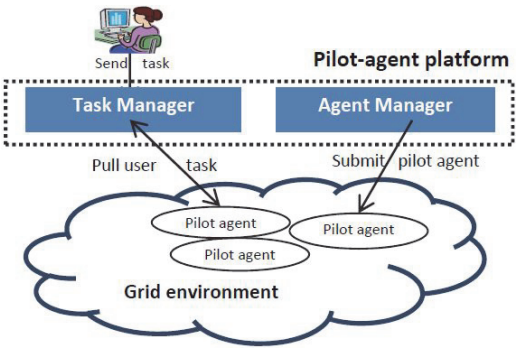

Figure 1: Pull model in pilot agent platform on grid.

As shown on figure 1, a pilot-agent platform has two main modules, the Task Manager and the Agent Manager.

Pilot agent is submitted automatically to grid by Agent Manager. Then it communicates with Task Manager and asks a user task to be executed. Task Manager receives tasks from user and control queue of user tasks. It receives also request of pilot agent, choose some task from queue and sends task to pilot agent.

Scheduling of pilot agent platform on grid takes place at site and platform level. The site level scheduling takes place in the site scheduler. Pilot agents sent by the platform are distributed to the sites according to the grid scheduling policy. The platform level scheduling is done by the platform's Task Manager. User sends the VS project to the Task Manager, where docking tasks are put in the task queue. The Task Manager Scheduler calculates task priorities, and responds to pilot agents requests by sending to them tasks ranked with the highest priority. There are underlying grid architectural scheduling and logical scheduling for the specific grid applications. And in our research, we concern with scheduling issue for many virtual screening application users who share the grid resources given to the same group privilege.

Moreover, platform level scheduling has limited machine availability property. Because each computing center imposes some limits to the maximum computing time for grid job, each pilot agent is available for a limited period. Therefore the number of machines available for the platform changes over time. This specific property makes our analysis directly relevant to cloud infrastructures where users buy computing resources for a limited time.

This paper focuses on finding out the most suitable scheduling policy at platform level to optimize the satisfaction of VS users.

1.2 The Stretch – A Measure for User's Satisfaction in Platform

To work on the fairness of scheduling policy, we need to define a good metric for the satisfaction of an individual user on platform. In the parallel scheduling literature, metrics used to measure the performance of scheduling policies can be classified in two groups: System-centric metrics and Job-centric metrics:

- System-centric metrics to assess platform utilization: C_j denotes the completion time of job j. Makespan (the maximum of the job

termination time, $\max_j C_j$) or the sum of completion time ($\sum_j C_j$) are common objective functions. Minimization of makespan or sum of completion times is conceptually a system-centric approach.

- Job-centric metrics (Flow time, stretch...) to assess user experience: Flow time F is the time an individual job spends in the platform. The stretch S (also called slowdown) is a particular case of weighted flow time: for job j with size W_j and flow time F_j, the stretch is defined as F_j/W_j. In the context of variable job sizes, the stretch is more relevant to describe user experience than the flow time (Legrand et al., 2006).

This paper focuses on user-centric metric. The stretch is here measured for a group of jobs all belonging to the same user. Assume that user has U jobs, F_j is flow time of job j, W is total size of U jobs, the stretch is then defined as $\max_{j \epsilon U} F_j/W$

Figure 2: Example of the stretch for two users.

Figure 2 illustrates the mean of stretch in the case of two users sending jobs to a platform (figure 2). User 1 has a job with size 10 that reaches the platform at t=0s. User 2 has a job with size 5 that reaches the platform at t=5s. User 1 job is executed before user 2 job. As in figure 2, user 1 finishes at t=10s. Because user 2 has to wait user 1 job completion, he finishes at t=15s. Although the two users spent the same time on the platform (10s), user 2 satisfaction is worse than user 1's. The stretch of user 1 is equal to 10/10=1 while the stretch for user 2 is equal to 10/5=2. This example illustrates why the stretch S measures the user experience on the platform: the larger the stretch, the lower the satisfaction.

Our goal is to identify the best scheduling policy to minimize the stretch of all VS users of a shared pilot-agent platform. We use the max-stretch (Smax) metrics as measures of fairness in our scheduling problem. In our latest research (Quang et al., 2013), we have compared two well-known scheduling policies, Shortest Processing Time (SPT-the user with the least number of tasks has the highest priority) and Longest Processing Time (LPT: the user with the greatest number of tasks has the highest priority), to the scheduling policies currently used on the existing platforms (FIFO and Round Robin). Simulation result and experimentation result on real platform has shown that SPT is the best policy in these 4 policies for online job-centric stretch optimization with virtual screening application on pilot-agent platform on grid/cloud(Quang et al., 2013).

Although SPT policy is very good online algorithm for optimization of the stretch but this policy has a disadvantage: it has the tendency to push users with many tasks to the end of the task queue. Sometimes these users have to wait a long time for a long series of users with less number of jobs. In the worst case, they have to wait forever. Furthermore, research on grid workloads in (Medernach, 2005) showed that there are two types of grid users: normal users and data challenge users. Normal user submits to the grid little number of tasks, but the number of user in this group is very large. While data challenge user group submit to the grid very large number of jobs, but the number of users in this group is small. If we use the original SPT policy for all of user in pilot agent platform, data challenge user will be negatively affected due to large number of user in normal group.

In this paper, we propose a new scheduling policy named SPT-SPT for platform level scheduling in pilot agent platform which uses multi-level queue scheduling techniques to improve the stretch of VS users. Instead of using one task queue which is implemented by SPT policy for all users, we use two separate task queues: one for normal user group and another one for data challenge user group. These two task queues are both using SPT policy to optimize the user's stretch in each one. Moreover, a task queue is assigned a parameter p ($p \epsilon [0, 1]$) and the rest one with 1 - p. This parameter is the probability that task queue will be selected by Task Manager when Task Manager receives request from pilot agents. We can see that for $p > 0$, this policy ensures that the data challenge group did not have to wait for the normal group to be entirely empty. Therefore all users will get an opportunity to utilize grid resources efficiently. The rest of paper is organized as follows. Section 2 presents the related works. In section 3, the SPT-SPT scheduling policy is proposed. Finally, section 4 is our conclusion and perspective on this research.

2 STATE OF THE ART

2.1 Grid Scheduling

Grid scheduling has been abundantly studied: some

surveys of grid scheduling algorithms are proposed in (Maruthanayagam, 2010); (Jiang et al., 2007) and performance of some priority rule scheduling algorithms is presented in (Azmi, 2011). DIET platform (Marrow et al., 2003) is a GridRPC middleware relying on the client/agent/server paradigm. The scheduling on DIET changes from FIFO, Round Robin and CPU-based scheduling. But the operation of DIET platform is different with pilot-agent platform: DIET use both "push" and "pull" scheduling. Mandatory requests are pushed from clients to resources, whereas optional requests are pulled by resources from clients. Pilot-agent platform takes most scheduling decisions in a centralized agent, in contrast, each client and each server contributes to taking scheduling decisions in DIET. Therefore, the solutions brought by research of scheduling problem on DIET platform are not directly applicable to our problem statement.

In (Berman et al., 1996), author presents a scheduling solution in application level called AppLeS. They describe an application specific approach to scheduling individual parallel applications on production heterogeneous systems. They utilize comprehensive information about application and resource to optimize execution time of application on grid. Our goal is not to optimize the execution time of all users but the quality of service for each user.

Existing pilot agent platforms such as DIANE (Mościcki, 2003), WPE(Kasam et al., 2009) and DIRAC (van Herwijnen et al., 2003) have different scheduling policies: WPE and DIANE platforms use FIFO while DIRAC uses Round Robin policy. The VS projects have specific properties such as divisibility in many docking tasks and no order of execution constraints. Therefore we need to find a suitable online scheduling policy for the VS application on the pilot-agent platform. Fortunately, in some platform such as DIRAC platform, we can configure the specific scheduling policy for a user group sharing the same application. So we can apply suitable policy in a VS user group to improve fairness.

2.2 Cloud Scheduling

As mentioned earlier, the limited machine availability property of the scheduling problem on pilot agent platform is similar with scheduling on cloud environment because on cloud environment, user buys some resources with limited duration. When a VS project is deployed on an IAAS cloud, docking task will be executed on a virtual machine with limited availability.

Some researches on cloud scheduling such as (Pandey et al., 2010); (Li et al., 2011) have presented their scheduling algorithms on cloud to optimize the speed of resources allocation, the price to pay and the utilization of system resource. But our object is optimization of the fairness of users when they share pilot-agent platform together.

In (Luckow et al., 2010), author proposed the design and implementation of a SAGA-based Pilot-Job system, which supports a wide range of application types, and is usable over a broad range of infrastructures from grids/clusters to cloud computing. In (Fifield et al., 2011), author showed also an extension of the pilot agent platform DIRAC on cloud computing by submitting pilot agent on Virtual Machine on cloud such as Amazon EC2. Therefore our research is also relevant to pilot-agent platforms on Cloud environments.

2.3 Scheduling for Stretch Optimization with Limited Machine Availability Constraints

Many groups have conducted research on optimizing job-centric stretch in the context of dedicated machines (i.e. always available). In (Muthukrishnan et al., 1999), S. Muthukrishnan presented the efficiency of the optimal on-line algorithm SPT on uniprocessor and multi-processor. Their objective is optimizing the average of the stretch. In (Legrand et al., 2006), Legrand has shown that SPT is quite effective at max-stretch and sum-stretch optimization in problems with continuous machines. But compared to these studies, our scheduling problem uses a user-centric definition of stretch and adds an additional constraint: machines have limited availability. With this property, the number of machines available for platform changes over time and the complexity of problem increases. In (Schmidt, 2000), authors have reviewed some scheduling algorithm in the context of limited machine availability. LPT is one of the online scheduling algorithms proposed in this research. But these researches are done on system-centric metrics (makespan, sum of completion time…). In our latest research for scheduling for stretch optimization with limited machine availability constraints, we compared two well-known scheduling policies, SPT and LPT, to the scheduling policies currently used on the existing platforms (FIFO and Round Robin). Simulation result and experimentation result on real platform showed that SPT policy is the best policy in these 4 policies for optimization of user stretch in

the context of limited machine availability.

3 SOLUTION PROPOSED

In this section, we briefly explain the proposed solution using multilevel queue technique in Task Manager of pilot agent platform. Administrator of pilot agent platform creates two groups for VS users: Normal group and Data Challenge group. VS user is assigned to Normal group by default. When someone needs to process a big virtual screening project, he will contact with administrator of pilot agent platform to change his role to Data Challenge group in some days or some weeks.

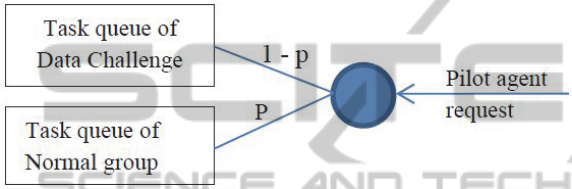

Figure 3: SPT-SPT policy with two task queues.

In Task Manager module, we build two separate task queues: one for normal group and another for data challenge group. We assign a priority p to normal group task queue and 1 – p to the data challenge group one. The Task Manager will use this probability as index in order to choose task queue to send their task when it receives request from pilot agent.

According to our latest research, SPT is better than FIFO, LPT, RR for minimizing the user stretch. Therefore these both task queues use SPT policy for optimizing the stretch of user on each one (We tried with policy SPT-RR: SPT on normal group task queue and Round Robin in data challenge task queue and SPT-FIFO: SPT on normal group task queue and FIFO on data challenge task queue. The result of SPT-RR and SPT-FIFO is worse than SPT policy). We use algorithm SPT-SPT to control user task in Task Manager as algorithm 1. Administrator can change value of parameter p in the configuration of pilot agent platform.

We can see that with $0 < p < 1$ there are always pilot agents which take a task from Data Challenge group. Therefore, Data Challenge user does not have to wait for a long series of normal users which have less number of tasks. We will use our simulator to find out the best value of p to decrease S_{max} of Data Challenge group and do not increase very much S_{max} of Normal group.

Algorithm 1: SPT-SPT policy in Task Manager scheduler.

```
p = parameter of task queue of normal user group
while (1)
    do
        if (receive request from pilot agent) then
            if (data challenge task queue is empty
                AND normal task queue is empty)
                Push pilot agent to pilot agent queue
            else if (data challenge task queue is
                                                empty)
                    Send task of normal task queue to
                                                pilot agent
            else if (normal task queue is empty)
                    Send task of data challenge queue to
                                                pilot agent
            else
                a = random(0,1)
                if (a ≤ p) then
                        Send task of normal task queue
                                                to pilot agent
                else
                        Send task of data challenge task
                                                queue to pilot agent
                end if
            end if
        end if

end while
```

4 CONCLUSIONS

The paper describes a new scheduling policy for virtual screening application on pilot agent platform for optimizing the stretch of user. We proposed SPT-SPT policy which uses multilevel queue technique for platform level scheduling on pilot agent platform. This approach based on the research of grid workload which has showed that there are two types of user (many users submitting small number of tasks and a little user submitting large number of tasks). Because Infrastructure as a Service cloud users buy access to computing resources for a limited time. This is similar with limited availability of pilot agent on grid. Therefore, we can also propose to implement SPT-SPT in deployment of virtual screening application on cloud environments.

REFERENCES

Azmi, B., 2011. Performance Comparison of Priority Rule Scheduling Algorithms Using Different Inter Arrival Time Jobs in Grid Environment. *International Journal of Grid and Distributed Computing*, 4(3), pp.61–70.

Berman, F. et al., 1996. Application-level scheduling on distributed heterogeneous networks. In *Supercomputing, 1996. Proceedings of the 1996 ACM/IEEE Conference on*. p. 39.

Fifield, T. et al., 2011. Integration of cloud, grid and local cluster resources with DIRAC. *Journal of Physics: Conference Series*, 331(6), p.062009.

Van Herwijnen, E. et al., 2003. Dirac-distributed infrastructure with remote agent control. In *Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*.

Jacq, N. et al., 2008. Grid-enabled virtual screening against malaria. *Journal of Grid Computing*, 6(1), pp.29–43.

Jacq, N. et al., 2006. Large scale in silico screening on grid infrastructures. *arXiv preprint cs/0611084*. Available at: http://arxiv.org/abs/cs/0611084 [Accessed December 31, 2013].

Jiang, C. et al., 2007. A survey of job scheduling in grids. In *Advances in Data and Web Management*. Springer, pp. 419–427.

Kasam, V. et al., 2009. WISDOM-II: screening against multiple targets implicated in malaria using computational grid infrastructures. *Malaria journal*, 8, p.88.

Legrand, A., Su, A. & Vivien, F., 2006. Minimizing the stretch when scheduling flows of biological requests. *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pp.103–112.

Li, W., Tordsson, J. & Elmroth, E., 2011. Modeling for dynamic cloud scheduling via migration of virtual machines. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. pp. 163–171.

Luckow, A., Lacinski, L. & Jha, S., 2010. SAGA BigJob: An extensible and interoperable pilot-job abstraction for distributed applications and systems. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*. pp. 135–144.

Marrow, P. et al., 2003. DIET-a scalable, robust and adaptable multi-agent platform for information management. *BT technology journal*, 21(4), pp.130–137.

Maruthanayagam, D., 2010. Grid scheduling algorithms: A survey. *International Journal of Current Research*, 11(2), pp.228–235.

Medernach, E., 2005. Workload analysis of a cluster in a grid environment. *Job scheduling strategies for parallel processing*, (June).

Mościcki, J.T., 2003. Distributed analysis environment for HEP and interdisciplinary applications. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 502(2), pp.426–429.

Muthukrishnan, S. et al., 1999. Online scheduling to minimize average stretch. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*. pp. 433–443.

Pandey, S. et al., 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. pp. 400–407.

Quang, B.T. et al., 2013. Stretch optimization for virtual screening on multi-user pilot-agent platforms on grid/cloud. In *Proceedings of the Fourth Symposium on Information and Communication Technology*. pp. 301–310.

Schmidt, G., 2000. Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1), pp.1–15.