Multi-Level Queue-Based Scheduling for Virtual Screening Application on Pilot-Agent Platforms on Grid/Cloud to Optimize the Stretch

Bui The Quang, Nguyen Hong Quang
IFI, Equipe MSI; IRD, UMI 209 UMMISCO
Vietnam, Hanoi
e-mails: nguyen.hong.quang@auf.org,
buithequang@gmail.com

Emmanuel Medernach, Vincent Breton
Laboratoire de Physique Corpusculaire
France, Aubière
e-mails: breton@clermont.in2p3.fr,
medernach@clermont.in2p3.fr

***Abstract*— Virtual screening has proven a very effective method on grid infrastructures. Operating a dedicated virtual screening platform on grid resources requires optimizing the scheduling policy. The scheduling can be done at 2 levels; at site level and at platform level. Site scheduling is done at each site independently. Each site allocates time slots for different groups of users. Platform scheduling is done at group level: inside a time slot jobs from many users are allocated. Pilot agents are sent to sites and act as a container of actual users jobs. They pick up users jobs from a central queue where the platform scheduling is done. This paper 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.**

Keywords-Virtual screening; grid computing; scheduling; fairness; stretch; online-algorithm; cloud computing; multilevel queue scheduling; SimGrid.

I. INTRODUCTION

In silico (i.e., computer-assisted) drug discovery [1] offers an efficient alternative to reduce the cost of drug development and to speed-up the discovery process. Virtual screening (VS) is achieved through a pipeline analysis, first step of which requires using a docking software, such as Autodock [2], Dock [3] or FlexX [4] to predict potential interacting complexes of small molecules in protein binding sites. 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 [5][6], and WISDOM-II [8] on malaria, and Avian Flu Data Challenge [7].

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 [8], DIRAC [9], DIANE [10], glideinWMS [11], and PanDA [12]. The DIRAC pilot-agent platform is now available to the users of several multidisciplinary virtual organizations on EGI (the European Grid Initiative) [9]. As 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, are respectively FIFO policy in WPE platform and 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, parameter sweeps [38], simulations, fractal calculations, computational biology [39], and computer imaging [40][41].

This paper is organized as follows. Section 2 describes the problem and our research objectives. Section 3 presents related works. Section 4 introduces our solution based on multi-queue scheduling. Section 5 discusses our results and presents our simulator based on SimGrid [26]. Finally, we conclude in Section 6 and give some perspectives on this research.

II. PROBLEM STATEMENT

In this paper, we evaluate the performances of many scheduling policies applied on the central pool of pilot jobs. The criterion used is the stretch for all users. We, then, explore new scheduling policy based on multiplexing user group queues depending onsome probability parameter. We will first describe our computing platform in details, then we will explain our scheduling policies and evaluate them with the help of simulation based on real workload traces.

 ISBN: 978-1-61208-339-1

A. *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 that is started through a grid resource manager. It is automatically submitted by platform and run on a computing machine on grid. We can see a pilot agent as container of jobs. The 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).

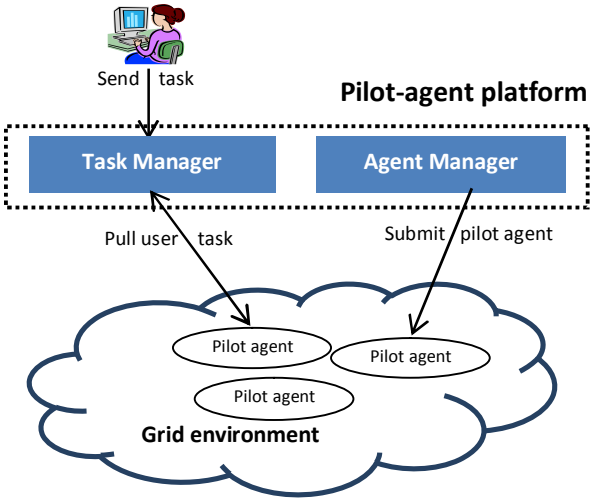

Figure 1. Pull model in pilot agent platform on grid.

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

The pilot agents are submitted automatically to grid by Agent Manager. Then each pilot agent communicates with Task Manager and asks a user task to be executed. Task Manager receives tasks from user and control queue of user tasks. Also, it receives request from pilot agent, choose some task from queue and sends it 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. We are concerned with scheduling issue for many virtual screening application users who share the grid resources given to the same group privilege.

Moreover, the platform level scheduling has limited machine availability property. Because each computing center requires 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.

B. *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, etc.) 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 [24].

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 $\max_{j\epsilon U} F_j/W$

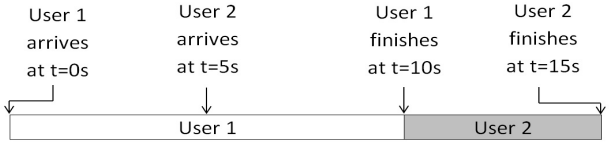

Figure 2. Example of the stretch for two users.

Figure 2 illustrates the mean of stretch in the case of two users sending jobs to a platform. 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

 ISBN: 978-1-61208-339-1

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 (S_{max}) metrics to measure of fairness in our scheduling problem. In our latest research [37], 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 [37].

Although SPT policy is very good online algorithm for optimization of the stretch, this policy has a disadvantage, i.e., 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 [27] 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 users in the 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 using multi-level queue scheduling techniques to improve the stretch of VS users. Instead of using one task queue 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 ($p \in [0, 1]$) 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.

III. RELATED WORK

A. Grid scheduling

Grid scheduling has been abundantly studied: some surveys of grid scheduling algorithms are proposed in [14][15] and performance of some priority rule scheduling algorithms is presented in [33]. DIET platform [17] 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.

Berman et al. [18] presented 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 [10], WPE [8], PanDA [12], DIRAC [9] and glideInWMS [11] 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.

B. 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 Infrastructure As A Service (IAAS) cloud, docking task will be executed on a virtual machine with limited availability.

Some researches on cloud scheduling, such as [19][20], 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.

Luckow et al. [21] 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. Fifield et al. [22] 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.

 ISBN: 978-1-61208-339-1

C. *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). Muthukrishnan et al. [23] presented the efficiency of the optimal on-line algorithm SPT on uniprocessor and multi-processor. Their objective is optimizing the average of the stretch. Legrand et al. [24] 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. Schmidt [16] 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, etc.). 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.

D. *Multi-level queue scheduling*

Various algorithms for multilevel queue are discussed in [34] to improve different CPU scheduling factors as turnaround time, waiting time, starvation problem, etc. These researches are done on multi-level queue scheduling technique used for CPU scheduling in operating system in a computer. In contrast, grid is a distributed computing environment. User tasks are executed by many distributed pilot agent on grid. Therefore, we need to evaluate multi-level queue technique on distributed computing environment of grid computing.

Chouhan et al. [35] and Kumaresh et al. [36] presented a scheduling policy for grid computing using multilevel feedback queue scheduling technique and multilevel queue scheduling to avoid the starvation of low priority jobs in the global scale of grid. However, in our context we need to find out a policy for platform level scheduling of pilot agent platform. And our objective is minimization of the user stretch, a measure for user experience.

In conclusion, to the best of our knowledge, no research on optimizing user stretch was conducted in the case of limited machine availability. In the next section, we describe solution proposed and our simulator used to evaluate and compare the performance of new policy to original SPT scheduling policy.

IV. 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.

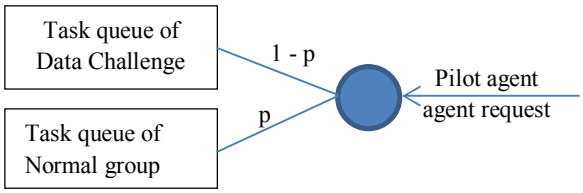

Figure 3. SPT-SPT policy with two task queues.

In the Task Manager module of the platform, we build two separate task queues: one queue for normal group and another one for data challenge group. Task queue of normal group is assigned priority p and task queue of data challenge group is assigned priority 1 – p. These indices are the probability that task queue is chosen by Task Manager to send pilot agent their task when Task Manager 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 described in Algorithm 1. The platform administrator can change value of parameter p in the configuration of pilot agent platform.

Algorithm 1: SPT-SPT policy in Task Manager scheduler

INPUT:
p = normal user group task queue parameter
Pilot agent requests are received online
OUTPUT:
Scheduling of tasks to pilot agents

for all pilot agent request received
do
if empty(data challenge task queue)
AND empty(normal task queue)
then
Push pilot agent to pilot agent queue
else if empty(data challenge task queue)
then
Send task of normal task queue to pilot agent
else if empty(normal task queue)
then

```
  Send task of data challenge task queue to pilot agent
 else
  if ( random(0,1) < 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
```

This is an online scheduling algorithm and it is linear in time complexity. We can see that with $0 < p < 1$ there are always pilot agents taking a task from Data Challenge group. Therefore, Data Challenge user does not have to wait for a long series of normal users having less 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.

V. EXPERIMENTATION ON SIMULATOR

A. Scenario description

There are three parameters for our experimentation: Configuration of grid infrastructure, VS user workload and parameter for pilot-agent platform.

1) Configuration of grid infrastructure

To simulate realistically the operation of a pilot-platform, we used archives of the AuverGrid regional multidisciplinary grid infrastructure in Auvergne (France) in 2004-2005, available on the Grid Workload Archive site. The AuverGrid infrastructure in this period is detailed in Table 1, including machines relative speeds. All machines in a cluster have the same speed.

TABLE I. CONFIGURATION OF THE AUVERGRID INFRASTRUCTURE

Cluster name	Number of Worker Node	Relative speed of Worker Node	Limitation of computing time (second)
CLRLCGCE01	112	1	258220
CLRLCGCE02	84	1.1	258220
CLRLCGCE03	186	1.6	258220
IUT15	38	0.8	172800
OPGC	55	1.4	172800

2) Virtual screening user workload

According to the research on the real grid workload done by Medernach [27], we use their model to generate workload example for virtual screening user as below:

Normal group has X^{normal} users, for each user U_i^{normal} in this group (ref. section 3.1: Mathematical model):

- $N_{j(i)}^{normal}$, the number of docking tasks of project j submitted by user i, is generated by a Geometric random distribution : $\gamma = a \times b^i$, parameter a corresponds to the first user mean number of docking tasks and parameter b is the geometric progression.
- $[r_{j(i)}^{normal}, r_{j+1(i)}^{normal}]$, the interval between submissions of two projects consecutive of user i, is generated within a Poisson random distribution with parameter $\lambda = c \times d^i$, parameter c corresponds to the first user mean inter-arrival time and parameter d is the geometric progression.
- We require max_time = 400 seconds to generate VS user workload example: $r_{j(i)}^{normal} < \text{max\_time}$

The same model is used for generating workload for data challenge group. In our simulation, we used the following parameters:

- Normal user group has parameters: $a = 1, b = d = \sqrt[40]{20}, c = 600$
- Data Challenge group has parameters: $a = 60000, b = d = \sqrt[20]{10}, c = 30000$

The workloads of normal user and data challenge user are combined in VS workload example. We generated 500 VS workload examples for each dataset. There are 4 datasets: case 00, case 01, case 02 and case 03 with different numbers of users in each group as table 2.

TABLE II. NUMBER OF USER IN EACH GROUP ON DATASET

	Number of Normal users	Number of Data Challenge users	Max time (second)
Case 00	119	1	400.000
Case 01	195	5	400.000
Case 02	190	10	400.000
Case 03	185	15	400.000

B. Simulation result and analysis

For each dataset (from case 00 to case 03), we run simulation on 500 VS workload examples for FIFO policy, SPT policy and SPT-SPT policy with the percentage of pilot agent for normal group p = 0%, 10%, 30%, 50%, 70%, 90% and 100% (ref. Algorithm 1). We calculate the $S_{\max}^{DC}$ and $S_{\max}^{normal}$ on each VS workload example as formula 1 and 2. Next, we figure out the average of $S_{\max}^{DC}$ and $S_{\max}^{normal}$ in each dataset. Figure 4 presents simulation results for case 00, case 01, case 02 and case 03. The more percentage of pilot agents for normal group, the more grid resource is reserved for normal user group and the less grid resource for data challenge group. Therefore, we can see in Figure 4 that, when p increases, the $S_{\max}^{normal}$ decreases and the $S_{\max}^{DC}$ augments. From the result of case 00, case 01, case 02 and case 03, we chose p = 70%, where the max-stretch of normal user group changes a little but the max-stretch of data challenge user decreases very much in comparison with original SPT policy. With this value of p, SPT-SPT policy improves user experience compare to original SPT policy. The best value of p depends on the number of task of two

 ISBN: 978-1-61208-339-1

groups. In SPT-SPT policy, administrator of pilot agent platform can adjust this parameter according to the actual situation for optimizing the stretch of two groups.

Table 3 presents the number of users of group in each dataset, the value $S_{\max}^{normal}$ and $S_{\max}^{DC}$ in SPT-SPT policy with p = 70%, in the original SPT policy and in FIFO policy. We can see that, in all cases the $S_{\max}^{normal}$ and $S_{\max}^{DC}$ in FIFO policy are higher than this one in SPT and SPT-SPT policy. For example, a normal user arrives just after a data challenge user, he has to wait very long time, so that S_{max} is very large in FIFO policy. The result shows that FIFO policy is not good to minimize the user stretch. Comparison between SPT and SPT-SPT policy (with p=70%), we can see that $S_{\max}^{normal}$ is approximate but $S_{\max}^{DC}$ in SPT-SPT policy is smaller than SPT policy. Moreover, from case 00 to case 03, we can see that the more data challenge users, the more $S_{\max}^{DC}$ is smaller than this one in SPT policy. This means that in the context with many data challenge users, the SPT-SPT policy is very much better than SPT policy.

VI. CONCLUSION AND PERSPECTIVE

The paper described a new scheduling policy for virtual screening application on pilot agent platform for optimizing the stretch of user. We proposed SPT-SPT policy using multilevel queue technique for platform level scheduling on pilot agent platform. This approach, based on the research of grid workload, has shown that there are two types of users : many users submitting small number of tasks and a little number of users submitting a large number of tasks. Simulation results showed that SPT-SPT policy (with 70% of pilot agent reserved for normal user group and 30% of pilot agent reserved for data challenge group) has better result on user stretch than original SPT policy. The stretch of data challenge user group decreases and the stretch of normal user is almost unchanged.

Infrastructure as a Service cloud is similar to our problem with limited availability of pilot agent on grid because their users buy access to computing resources for a limited time. Therefore, we also propose to implement SPT-SPT in deployment of virtual screening application on cloud environments.

ACKNOWLEDGMENT

We are grateful to “Agence Universitaire de la Francophonie” for their grant support. The authors are also thankful to France Grilles and EGI for providing computing resources used on the Biomed Virtual Organization. We also acknowledge FKPPL and FVPPL LIAs support for travel and exchange between research groups in France, Korea and Vietnam. We warmly thank Vanessa Hamar and Andrei Tsaregorodtsev for their assistance with the DIRAC platform.

REFERENCES

[1] V. S. Rao and K. Srinivas,. “Modern drug discovery process: an in silico approach.”, Journal of Bioinformatics and Sequence Analysis, 2(5), 2011, pp. 89-94.

[2] D. S. Goodsell, G. M. Morris, and A. J. Olson, “Automated docking of flexible ligands: applications of AutoDock.”, Journal of Molecular Recognition, 9(1), 1996, pp. 1-5.

[3] R. G. Coleman and K. A. Sharp, “Protein pockets: inventory, shape, and comparison,” Journal of chemical information and modeling, 50(4), 2010, pp. 589-603.

[4] I. Schellhammer and M. Schellhammer, “FlexX-Scan: Fast, structure-based virtual screening,” PROTEINS: Structure, Function, and Bioinformatics, 57(3), 2004, pp. 504-517.

[5] N. Jacq, V. Breton, H. Y. Chen, L. Y. Ho, M. Hofmann, H. C. Lee, and M. Zimmermann, “Large scale in silico screening on grid infrastructures,” [arXiv preprint cs/0611084].

[6] N. Jacq, J. Salzemann, F. Jacq, Y. Legré, E. Medernach, J. Montagnat, and V. Breton, “Grid-enabled virtual screening against malaria,” Journal of Grid Computing, 6(1), 2008, pp. 29-43.

[7] H. C. Lee, J. Salzemann, N. Jacq, H. Y. Chen, L. Y. Ho, I. Merelli, and Y. T. Wu, “Grid-enabled high-throughput in silico screening against influenza A neuraminidase,”, IEEE transactions on nanobioscience, 2006, pp. 288-295.

[8] V. Kasam, J. Salzemann, M. Botha, A. Dacosta, G. Degliesposti, R. Isea, D. Kim, A.Maass, C. Kenyon, G. Rastelli, M. Hofmann-Apitus and V. Breton, “WISDOM-II: Screening against multiple targets implicated in malaria using computational grid infrastructures,” Malaria Journal, 2009, 8(1), pp. 88-103,

[9] E. van Herwijnen, J. Closier, M. Frank, C. Gaspar, F. Loverre, S. Ponce, and M. Gandelman, “Dirac—distributed infrastructure with remote agent control,” Conference for Computing in High-Energy and Nuclear Physics (CHEP 03), 2003.

[10] J. T. Mościcki, “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), 2003, pp. 426-429.

[11] I. Sfiligoi, “glideinWMS—a generic pilot-based workload management system,” Journal of Physics: Conference Series, vol. 119, No. 6, IOP Publishing, Jul. 2008

[12] T. Maeno, “PanDA: distributed production and distributed analysis system for ATLAS,” Journal of Physics: Conference Series, vol. 119, No. 6, IOP Publishing, 2008

[13] R. F. Da Silva, S. Camarasu-Pop, B. Grenier, V. Hamar, D. Manset, J. Montagnat, and T. Glatard, “Multi-infrastructure workflow execution for medical simulation in the Virtual Imaging Platform,” Proceedings of the 9th HealthGrid Conference. 2011, pp. 1-10.

[14] D. Maruthanayagam and R. Uma Rani, “Grid scheduling algorithms: a survey,” International Journal of Current Research. vol. 11 , Dec. 2010, pp. 228-235.

[15] C. Jiang, C. Wang, X. Liu, and Y. Zhao, “A survey of job scheduling in grids,” Advances in Data and Web Management, Springer Berlin Heidelberg, 2007, pp. 419-427.

[16] G. Schmidt, “Scheduling with limited machine availability”, European Journal of Operational Research, 121(1), 2000, pp. 1-15.

[17] P. Marrow, E. Bonsma, F. Wang, and C. Hoile, “DIET—a scalable, robust and adaptable multi-agent platform for information management,” BT technology journal, 21(4), 2003, pp. 130-137.

[18] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, “Application-level scheduling on distributed heterogeneous networks,” Proceedings of Supercomputing. vol. 96. 1996, pp. 1-28.

[19] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, “A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments,” AINA '10: Proceedings of the 2010, 24th IEEE International Conference on Advanced Information Networking and Applications. Washington, DC, USA, IEEE Computer Society, 2010, pp. 400-407.

 ISBN: 978-1-61208-339-1

[20] W. Li, J. Tordsson, and E. Elmroth, "Modeling for dynamic cloud scheduling via migration of virtual machines," Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011), 2011, pp. 163-171.

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

[22] T. Fifield, A. Carmona, A. Casajús, R. Graciani, and M. Sevior, "Integration of cloud, grid and local cluster resources with DIRAC", Journal of Physics: Conference Series, vol. 331, No. 6, 2011. [ref. 062009, doi:10.1088/1742-6596/331/6/062009]

[23] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J. E. Gehrke, "Online scheduling to minimize average stretch," IEEE Symposium on Foundations of Computer Science, 1999, pp. 433-442.

[24] A. Legrand, A. Su, and F. Vivien, "Minimizing the stretch when scheduling flows of biological requests," Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures, 2006, pp. 103-112.

[25] B. Chen, C. N. Potts, and G. J. Woeginger, "A review of machine scheduling: Complexity, algorithms and approximability" Handbook of combinatorial optimization, ch 3, 1998, pp. 21-169.

[26] H. Casanova, A. Legrand, and M. Quinson, "SimGrid: a generic framework for large-scale distributed experiments" Proceeding 10th International Conference Computer Modeling and Simulation, Mar. 2008, pp. 126-131

[27] E. Medernach, "Workload analysis of a cluster in a grid environment" Job scheduling strategies for parallel processing, Springer Berlin Heidelberg. 2005, pp. 36-61.

[28] E. L. Lawler, J. K. Lenstra, and A. R. Kan, "Sequencing and scheduling: Algorithms and complexity," Handbooks in operations research and management science, 4, 1993, pp. 445-522.

[29] T. C. E. Cheng and C. C. S. Sin, "A state-of-the-art review of parallel-machine scheduling research," European Journal of Operational Research, 47(3), 1990, pp. 271-292.

[30] Jain, Raj, "The art of computer systems performance analysis," vol. 182. Chichester: John Wiley & Sons, 1991.

[31] A. B. Downey, "A parallel workload model and its implications for processor allocation," Cluster Computing 1.1, 1998, pp. 133-145.

[32] Feitelson, G. Dror "Packing schemes for gang scheduling," Job Scheduling Strategies for Parallel Processing, Springer Berlin Heidelberg, 1996.

[33] Z. R. M. Azmi, K. A. Bakar, A. H. Abdullah, M. S. Shamsir, and W. N. W. Manan, "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), 2011, pp. 61-70.

[34] C. Vaishali. and R. Supriya "A Review of Multilevel Queue and Multilevel Feedback Queue Scheduling Techniques," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, iss. 1, pp. 110-113.

[35] D. Chouhan, S. M. Dilip Kumar, and B. P. Vijaya Kumar, "Multilevel Feedback Queue Scheduling Technique for Grid Computing Environments", in Proceedings of International Conference on Advances in Computing SE - 1, vol. 174, A. Kumar M., S. R., and T. V. S. Kumar, Eds. Springer India, 2012, pp. 1–7.

[36] V.S Kumaresh,S. Prasidh, B. Arjunan,S Subbhaash and M.K. Sandhya, "Multilevel Queue-Based Scheduling for Heterogeneous Grid Environment", International Journal of Computer Science Issues, vol. 9, Issue 6, No 3, Nov. 2012, Springer India, 2012.

[37] T. Q. Bui, E. Medernach, V. Breton, H. Q. Nguyen, and Q. L. Pham, "Stretch Optimization For Virtual Screening on Multi-user Pilot-agent Platforms on Grid/Cloud", Proceedings of the Fourth Symposium on Information and Communication Technology, 2013.

[38] D. Abramson, J. Giddy and L. Kotler. "High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid," IPDPS'2000, Cancun Mexico, IEEE CS Press, 2000, pp. 520-528.

[39] J. R. Stiles, T. M. Bartol, E. E. Salpeter, and M. M. Salpeter, "Monte Carlo Simulation of Neuromuscular Transmitter Release Using MCell, a General Simulator of Cellular Physiological Processes," Computational Neuroscience, 1998, pp. 279-284.

[40] S. Smallen, W. Cirne and J. Frey et al, "Combining Workstations and Supercomputers to Support Grid Applications: The Parallel Tomography Experience," Proceeding of the HCW'2000-Heterogeneous Computing Workshop, 2000

[41] S. Smallen, H. Casanova, and F. Berman, "Applying Scheduling and Tuning to On-line Parallel Tomography," Proceedings of Supercomputing 01, Denver, Colorado, USA, Nov. 2001

 ISBN: 978-1-61208-339-1

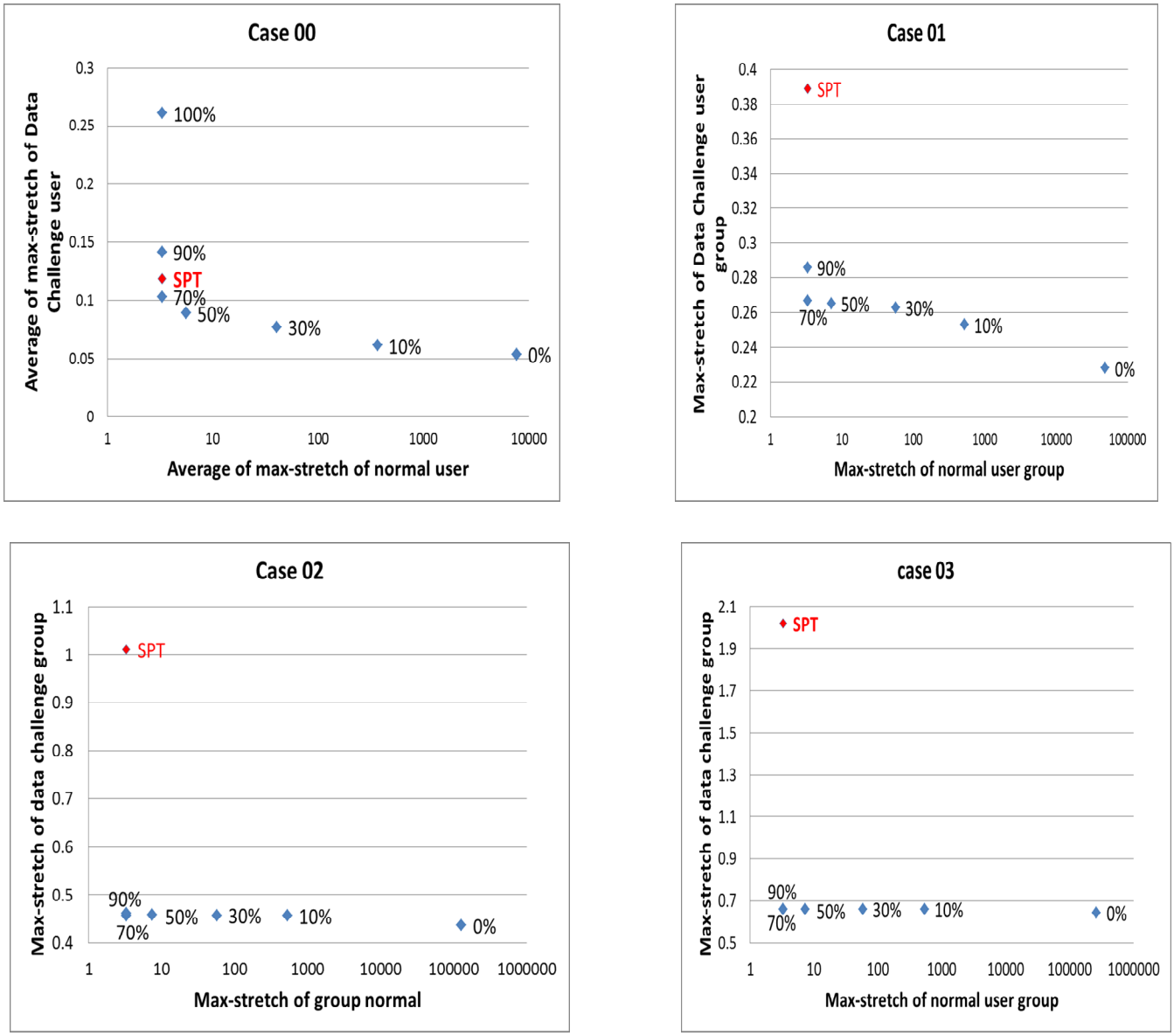

Figure 4. Average of max-stretch of two groups versus % pilot agent for normal user group in case 00, 01, 02, 03