A P2P Approach to Many Tasks Computing for Scientific Workflows*

Eduardo Ogasawara[1], Jonas Dias[1], Daniel Oliveira[1], Carla Rodrigues[1], Carlos Pivotto[1], Rafael Antas[1], Vanessa Braganholo[1], Patrick Valduriez[2], Marta Mattoso[1]

[1] Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
{ogasawara, jonasdias, danielc, carlarod, pivotto, zuquim, marta}@cos.ufrj.br,
braganholo@dcc.ufrj.br
[2] INRIA & LIRMM, Montpellier, France
Patrick.Valduriez@inria.fr

Abstract. Scientific Workflow Management Systems (SWfMS) are being used intensively to support large scale *in silico* experiments. In order to reduce execution time, current SWfMS have exploited workflow parallelization under the arising Many Tasks Computing (MTC) paradigm in homogeneous computing environments, such as multiprocessors, clusters and grids with centralized control. Although successful, this solution no longer applies to heterogeneous computing environments, such as hybrid clouds, which may combine users' own computing resources with multiple edge clouds. A promising approach to address this challenge is Peer-to-Peer (P2P) which relies on decentralized control to deal with scalability and dynamic behavior of resources. In this paper, we propose a new P2P approach to apply MTC in scientific workflows. Through the results of simulation experiments, we show that our approach is promising.

Keywords: Scientific experiments, scientific workflows, Scientific Workflow Management Systems (SWfMS), many tasks computing (MTC), peer-to-peer (P2P).

1 Introduction

The evolution of computer science in the last decade has enabled the exploration of a new type of scientific experiments based on computer simulations, known as *in silico* experiments [1]. In such experiments, scientists may use different programs to perform an activity. In these scenarios data produced by one activity needs to be passed as input to another activity, and conversion steps may need to be performed along the execution. This chain of programs that composes a scientific experiment is usually represented as scientific workflows. A scientific workflow is an abstraction that allows the structured composition of programs as a sequence of activities aiming a desired result [1]. This sequence is supported by Scientific Workflow Management Systems (SWfMS), which are software packages that enable to define, execute, and monitor scientific workflows. SWfMS are responsible for coordinating the invocation

* Work partially sponsored by CNPq and INRIA within the Sarava équipe associée and by CAPES.

(also called orchestration) of programs (specified within scientific workflows) to be executed either locally or in remote environments.

Due to the exploratory nature of the scientific method [2], an *in silico* experiment may require the exploration of a certain scientific workflow using different parameters or input data. This situation occurs, for example, in Monte Carlo simulations, parameter sweep, and data mining, where the same workflow is exhaustively executed until the exploration finishes. Recently, these experiments were grouped into a new computational paradigm called Many Tasks Computing (MTC) [3]. It consists on using various computing resources over short periods of time to accomplish many (dependent or independent) computational tasks.

SWfMS have successfully exploited workflow parallelization in homogeneous computing environments, such as multiprocessor or cluster systems. These environments rely on centralized control of resources which eases parallelization and exploit high-throughput, low-latency communication networks which bring performance in order to confirm or refute a hypothesis. However, there are now many different heterogeneous computing environments which could be exploited for executing scientific workflows, for instance, grids, desktop grids, volunteers computing projects (such as BOINC [4] and World Community Grid [5]), or hybrid clouds [6], which may combine users' own computing resources with multiple edge clouds. The main problem is that each of these computing environments requires different efforts, resources and scientists skills to be applied within a solution for workflow parallelization. However, a major requirement for scientists is to model activities or workflows to be parallelized in an implicit way, independent of the target environments. Another important requirement for scientists is provenance gathering [7], *i.e.*, the capability of reproducing the results of a scientific experiment. Nevertheless, provenance gathering in heterogeneous distributed environments is still an open problem.

Peer-to-Peer (P2P) is a promising approach to address the aforementioned challenges of applying MTC to scientific workflows. Unlike traditional distributed (client-server) computing, P2P relies extensively on decentralized control and the ability of any node (peer) to perform any task which makes it possible to deal with scalability and dynamic behavior of nodes (including churns, *i.e.*, frequent joins and leaves of the P2P network).

In previous work [8], the authors have already discussed how P2P techniques can be very useful to large-scale grids. More generally, we believe that, in heterogeneous computing environments, P2P is a powerful approach to support MTC of scientific workflows, with integrated provenance gathering.

In this paper, we explore a new P2P approach to support MTC through scientific workflows in heterogeneous computing environments. We propose SciMule, a P2P middleware that allow transparent parallelization of workflow tasks and dynamic scheduling of MTC. To evaluate our approach, we developed a SciMule simulation environment and performed several experiments under typical scientific workflow scenarios.

The paper is organized as follows. Section 2 introduces a comparative study based on qualitative aspects of the state of art P2P approaches suitable to deal with workflow activity distribution. Section 3 describes the SciMule conceptual

architecture and presents our parallel workflow execution strategies. Section 4 presents experimental results. We conclude in Section 5.

2 Backgrounds on P2P Networks

P2P systems can be classified into two types: centralized and decentralized [9]. Centralized systems use one or more servers (also called super-peers) which are responsible for storing information about other peers and solving P2P network requests such as joining the network. Decentralized systems do not rely on specific peers for network control, because information is totally distributed between all peers. Furthermore, there are hybrid solutions which combine characteristics of the two aforementioned types.

The hierarchical P2P approach is a hybrid solution to decrease complexity and topology maintenance overhead of large-scale decentralized P2P systems. This approach creates several smaller network groups, where the peers still form a decentralized network. Some peers from a group on a given hierarchy level connect with peers from a higher hierarchy level, thus forming the hierarchical structure.

Canon [10] is a DHT-based hierarchical approach. In Canon, each peer creates links to other peers on its own hierarchical level generating a domain. The hierarchy is constructed by adding links from each peer in one domain to some set of peers in other domains, resulting in a higher level group. Messages are routed inside domains. If a query regarding some resource discovery cannot be resolved locally, then it is routed to a peer *p* which keeps connection with the next higher level. At Canon, churns at lowest levels do not affect the entire network and only internal neighborhood tables need be updated.

Garcés-Erice *et al.* [11] proposes a hierarchical approach using the super-peer model. They propose a network formed by two types of nodes: ordinary nodes and super-peers. Ordinary nodes are organized into groups and exchange messages only inside their group. Super peers are gateways between groups. They create links to others super-peers for inter-group message exchanging and keep information about ordinary nodes and their content.

Considering the super-peer model, hierarchical structures can be used in Grid Systems as shown in Mastroianni et al. [12], where each virtual organization in the Grid has a single super-peer which establishes connection with other super-peers to form a network at a higher level. Super peers control membership requests and resource discovery services on the Grid. A similar architecture is discussed in a previous work [13], where a CAN-based P2P network is built according to the super-peer model using a more sophisticated message routing. Super-peers keep information about local nodes and neighboring super-peers.

The main focus of this work is on executing scientific workflows in large-scale P2P networks and includes the network type specification. There are many aspects that should be considered to select a suitable approach to deal with distributed workflow activities execution. A single workflow activity can generate thousands of tasks that would be distributed over the P2P network. Thus, the *load balancing* factor is very important to keep the network stable without overloading a set of peers. *Scalability* is also important since it is a large-scale network that deals with thousands of nodes. P2P is supposed to be fault-tolerant but churn events are predicted. So, the

churn risk, i.e., how impactful is a churn event on the worst case scenario in the network, is also very important on any P2P system. *Maintenance cost* of the P2P topology may also be an important aspect since it affects the scheduling process, since the node submitting a workflow activity should know where it can distribute tasks. Table 1 summarizes an analysis considering the types of P2P networks we previously discussed and the following factors: (i) Load Balancing; (ii) Scalability; (iii) Churn Risks and; (iv) Maintenance Cost.

Table 1: Analysis of the P2P networks approaches

Factor \ Network type	**Centralized**	**Decentralized**	**Hierarchical**
Load Balancing	None	High	Moderate
Scalability	Low	Moderate	High
Churn Risks	Total Failure	Ponctual Failures	Domain Failures
Maintenance Cost	Low	High	Moderate

Centralized P2P networks have no load balancing since the super-peers centralize resource discovery and search services. The decentralized approach fully distributes control over the network, thus providing high load balancing. The hierarchical approach centralizes part of the services on some special (inter-group) peers, thus providing a moderate load balancing. The centralized approach relies on the central node capacities, so it does not scale very well. The decentralized approach has a moderate scalability since it is hard to maintain a huge network with fully distributed control. The hierarchical approach scales better since it establishes part of the control on the inter-group peers.

Regarding the churn risks, in a centralized network, if a churn happens on the central nodes, the whole network fails. In a decentralized network, the churn represents just a punctual failure, since the nodes are independent. On a hierarchical P2P network, if a churn happens on an inter-group peer, the whole group fails. The maintenance cost of a centralized network is low, since only the information on the central nodes has to be updated. In the decentralized approach, though, the information is distributed and updates usually involve flooding algorithms. On the hierarchical approach, the inter-group peer keeps some information about its groups and it is the gateway between its group and the others. Flooding may happens only inside the groups.

Considering these aspects, we believe hierarchical P2P networks may be the most adequate solution for distributing scientific workflows activities, especially for large-scale networks that demand great performance.

3 Design of SciMule

SciMule is a middleware designed to distribute, control and monitor the execution of activities of a scientific workflow in a P2P environment. We consider P2P environment as a distributed and heterogeneous computing environment where activities, data and parameters are distributed over the network to promote workflow/activity parallelization. These activities can be programs or even

independent scientific workflows. SciMule was designed considering a hierarchical structured approach. We choose the hierarchical approach since it establishes a good tradeoff between the centralized and decentralized approaches, which generally scales well for large scale P2P networks, while tolerating churn effects. SciMule has a three-layer architecture: (i) a submission layer that dispatches activities to be executed in the P2P environment through a generic SWfMS, (ii) an execution layer that receives experiment packages (*i.e.* activities, parameters and data) that need to be executed, and (iii) the overlay layer that holds information about how peers are placed on the network and how they are related together. This type of information is important when a new node needs to be inserted on the overlay and also to keep the P2P network balanced.

During SciMule network lifetime, a peer may play several different roles. It may act as a client peer submitting new activities through the submission layer, or it may act as an executor peer, receiving tasks to be executed by the execution layer. A peer may also be elected as a gate peer. The gate peer is responsible for keeping and publishing the list of nearby peers and their subjects. A subject is an abstraction that represents a set of programs related to a certain domain of knowledge. The gate peer role is managed by SciMule through the overlay layer. Each role is strongly coupled to a specific layer, but they are all independent. Peers may act only as a client, others may act only as executors, but they usually act as both. An elected gate peer may also act as client and/or executor.

SciMule aims at isolating scientists from the complexity of distributing workflow activities (or entire workflows) using MTC paradigm over a P2P network. This is done by offering a transparent and explicit structure to distribute scientific workflows activities that demand high computation. In this way, SciMule is an adaptation of Hydra [14] for the P2P environment. Hydra is also a middleware which provides a set of components to be included on the workflow specification of any SWfMS to control parallelization of activities in clusters. While Hydra and SciMule share many conceptual behavior, their architecture design is completely different due to the intrinsic characteristics of P2P versus client server architecture. For example, in SciMule, a computer may distribute a set of tasks that compose an activity acting as a client peer, but it may latter run tasks that arrive from other peers acting as an executor peer. This behavior is not supported by Hydra.

3.1 SciMule Architectural Features

SciMule shares many important features with Hydra [14]. It was also designed to provide two different types of parallelization: parameter sweep parallelism and data parallelism. These two types of parallelism may normally represent a barrier for the scientists to control and register provenance, since they require a great effort and discipline to manage too much information when executed in an ad-hoc manner over any distributed environment. SciMule provides a systematic approach to support both types of parallelism with heterogeneous distributed provenance gathering. Some of these features are also available in Hydra, but SciMule aims a broader execution experience running asynchronously and distributed over the dynamic and heterogeneous P2P network. Meantime, SciMule architecture is still simple to deploy

and able to be architecturally linked to any SWfMS with minimum effort. The entire architecture is described in the following sub-sections.

3.2 SciMule Conceptual Architecture

Figure 1 presents the SciMule architecture. As mentioned before, it is composed by three layers: submission, execution and overlay. It is important to observe that each peer has all the three layers. Numbers alongside the arrows of Figure 1 denote the execution sequence of the architecture components in the scope of each layer.

Submission Layer Components. SciMule submission layer components provide transparent ways to parallelize scientific workflows and to distribute activities through neighbor peers using MTC paradigm. It is divided in two basic parts: workflow components and MTC controller components. The workflow components represent generic modules that are included in the SWfMS to enable the interaction with SciMule. SciMule has two main components to be plugged into the SWfMS: *Client Setup* and *Client Dispatch.*

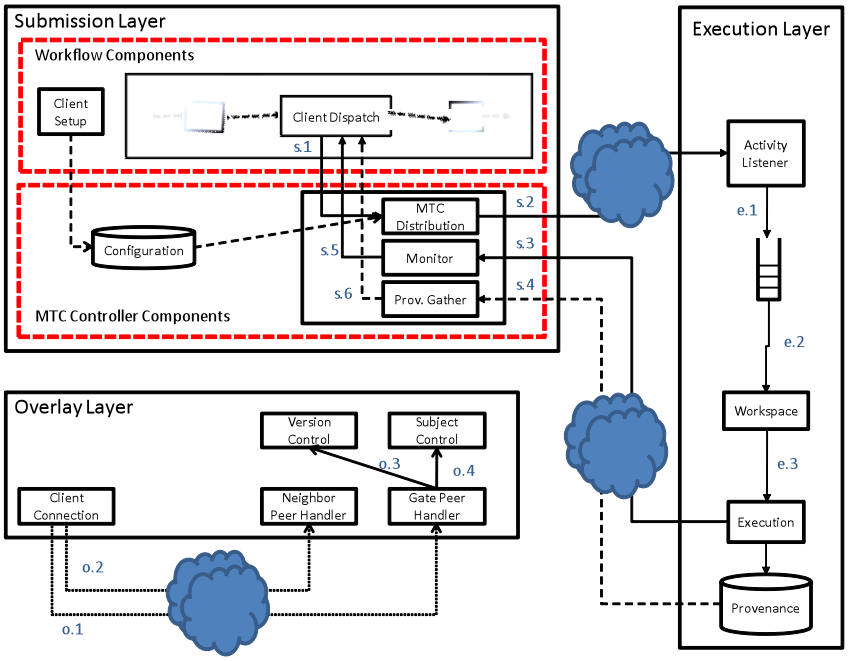

Figure 1 SciMule Architecture

Similar to Hydra, the *Client Setup* is the component responsible for general configuration of the type of parallelism to be used during workflow execution and the P2P environment, which includes both data files and template files [14] to be transferred over the P2P network. All these configurations are used by SciMule MTC controller. The *Client Dispatch* distributes activities over the P2P network through the MTC controller during workflow executions. The execution on the P2P network is

transparent while using this component. Once the distributed execution is finished, the *Client Dispatch* returns control to the SWfMS.

The MTC Controller Components represents the SciMule MTC engine that is responsible for distributing tasks, gathering distributed provenance data and handling churns during workflow execution. The MTC controller components are invoked by the Client Dispatch (event s.1 of Figure 1). An activity that needs to be distributed is divided into a set of tasks. A task is an experiment package which includes enough information for the execution, *i.e.*, workflow code, data and parameter for execution. Each task is scheduled considering the status of the peer neighbors (computing power, number of tasks being processed and the available bandwidth). The status information may be obtained through gossip [15]. After this point, each task contains the executor peer address. The peer address may be the local machine, which means that the task is going to be executed locally. All the optimization process is static for each distributed activity. This means that once a task is scheduled to be executed by peer α, it is executed by α, unless α suffers a churn (in this case the task may be redistributed to another peer).

The MTC distribution component distributes the tasks (event s.2 of Figure 1) and monitors all peers that are running one of the distributed tasks (event s.3 of Figure 1). Once a task finishes its remote execution, provenance data is collected from the remote peer (event s.4 of Figure 1). Like in Hydra [14], the provenance repository was modeled to link the prospective provenance and the retrospective distributed provenance [7] collected during the scientific experiment life cycle. Once all tasks are completed, the control is returned to the SWfMS (event s.5 of Figure 1), together with the collected provenance data (event s.6 of Figure 1).

Execution Layer. On the execution layer, there are three main components: the activity listener, the workspace handler and the task handler. The activity listener is the component responsible for listening to the connections with other peers in order to receive a task. Each task received is put into a queue (event e.1 of Figure 1) to be consumed by the task handler.

The workspace handler is responsible for setting up the executor peer environment to execute the task, *i.e.*, unpack the experiment package for execution (it creates the directory structure, sets parameters, etc). Once the task is ready (event e.2 of Figure 1), the task handler invokes the corresponding program using the parameters and data that were packaged in the task (event e.3 of Figure 1). When the execution is finished, the control returns to the MTC distribution.

Overlay Layer. In order to support MTC, SciMule overlay is defined to balance peers according to locality principles [15] and to subject. Each subject is a preset tool of a certain domain (*e.g.*, bioinformatics). The goal is that peers keep communication to other peers that enroll the same subject, which means that they may have the same set of programs. The principle of locality establishes that it is also important that peers cooperate with other peers that are close to them in locality. In this way, SciMule is a scalable P2P MTC network that is both locality and subject aware.

In the SciMule approach, most information about the overlay is stored on gate peers. Gate peers keep a list of nearby nodes, *i.e.*, nodes that once contacted them to join the P2P network. A gate peer has one or more subjects associated to it. Gate peers control versions of their related subjects, replicate them on nearby gate peers

and notify known peers enrolled with the same subject about the new versions, so they can update their files. The rules for deploying new versions of subjects assume concepts of reputation, *i.e.*, only Gate peers with greater reputation may deploy new versions of subjects. Gate peers are elected according to some metrics: they are assumed to have low churn frequency and high reputation. To avoid data loss, gate peers also have a backup node that replicates nearby peers list. Data management and replica control on large-scale P2P systems are well discussed by Akbarinia *et al.* [16]. Leader election occurs for a promotion of a peer into to a gate peer to guarantee a certain rate between gate peers and peers in the network.

In SciMule, each gate peer keeps a list of ordinary nodes that defines a group, and each ordinary node belongs to only one group, which means that it is registered in a single gate peer. However, a peer may have neighbors from its own group and from two other adjacent groups. Thus, when churn happens, three groups of the network are affected, at most. Any node inside a group can submit a task from a workflow activity directly to its neighbors. This decentralized submission enhances the load balancing.

The amount of gate peers in the network directly affects the maximum connectivity of a node in the network. On a given network with n peer nodes and g gate peers distributed on the network, the gate peers keep a mean of n/g nearby nodes. Each node is registered on the nearest gate peer, from where it obtains the list of other nearby nodes. The node also contacts another nearby gate peer to get a broader listing. Thus, a node registered on a given gate peer may have neighbors registered on the same gate peer and also from the other two adjacent gate peers. Since each gate peer has n/g nodes on its list, the maximum connectivity of the node is $3n/g$.

Although each peer contacts some gate peers to join SciMule network, it does not necessarily establishes inter-peer connections with them, since other peers may be better suited to the selection criteria. SciMule limits the number of neighbors that an incoming peer can get. A novice peer only has about half the average of neighbors that older peers have. This limitation aims at avoiding free riding [17]. The number of inter-peer is also important to maintain the network balanced, which means that new incoming peers connect to peers with lower inter-peer connection cardinality. The cardinality information of a group is stored in the gate peer and is periodically refreshed. When a peer requests the nearby peer list, it comes ordered by cardinality. We are also studying the possibility to spread this information using gossip [15]. The goal is to balance the P2P network structure along time. Figure 2 presents an example of P2P MTC configuration.

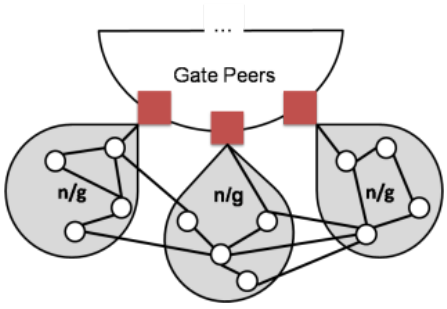

Figure 2 – P2P MTC configuration

4 SciMule Evaluation

We have built a simulation environment to evaluate the proposed architecture. We chose the simulation approach instead of a real life experiment since it is difficult and expensive to build a real P2P infrastructure with thousands of nodes just to test and evaluate our architecture [18].

SciMule Simulator was developed on top of PeerSim [19]. It was necessary to extend the PeerSim simulator and add some extra components to support SciMule architectural demands. The PeerSim node component was extended to store locality and subjects. It also got the ability to become a gate peer, which means that it can store a list of nearby nodes. We also modeled the link between nodes, the data package that transit on links, the workflow activity and its tasks. The data transferring system considers the bandwidth of both sender and receiver peers. Since a peer can perform multiple data transfers, the system uses a round robin approach to share the bandwidth. The workflow activity is modeled as an object that can be decomposed in a set of tasks. A task is an object that has a size and a processing cost. The size attribute is considered for data transfers while the cost is considered when the task is computed by one of the executor peers. Most parts of the simulator were developed following SciMule three-layer architecture.

The simulator allows the control of several variables, like the size of the network and characteristics of the workflow activities that are submitted during the simulation. For the sake of performance, the simulator uses PeerSim cycle-based approach. The variables were parameterized based on a real life experiment [14] assuming that a cycle corresponds to 30 seconds.

Our study has eight independent variables [20]: (i) the number of simulation cycles, (ii) the initial number of peers (*n*) in the network, (iii) average number (*k*) of neighbors, (iv) activities submission frequency (*f)* that obeys a Poisson distribution, (v) number of tasks (*tasks*) of an activity, (vi) task computational cost (*cost*) in processing units (p.u.), (vii) task data size (*size*) in kilobits and (viii) churn frequency (*churn*), also following a Poisson distribution. Five of these variables – *k*, *tasks*, *cost*, *size* and *churn* – are factors [20] which have from two to four treatments [20]. On this aspect, our simulation experiment is a five-factor, four-treatment study [20]. Table 2 summarizes our independent variables and factors with their respective treatments [14].

Table 2: Summary of the study variables

			Factors				
Independent Variables							
cycles	***n***	***f***	***k***	***tasks***	***cost***	***size***	***churn***
14400	4096	0.01	32	128	4000	12000	0.00
			64	512	8000	24000	0.05
			128		16000	48000	0.10
			256		32000	96000	

The combinations of all factors generated 384 instances of simulation. The dependent variables [20], *i.e.*, the values we assess, are the speed up and the time spent transferring data of each activity executed on the P2P network. However, our

analysis focuses on speed up results, in order to evaluate the general performance of the network due to increase of number of peers involved on execution.

Each one of the 384 instances was considered an independent simulation. All the simulations instances ran on a SGI Altix 8200 cluster using Hydra [14]. The simulation results were stored on a PostgreSQL database. We have made a statistical analysis of the data taking the average speed up from the completed executions of the distributed activities [21]. We have selected four representative activities for our experimental analysis. Two of them have Low task Cost (LCMS and LCBS) with different task sizes. And the other two have Small task Size (HCSS and MCSS) with different costs. They are described on Table 3. Since we are measuring speed up, the processing units (p.u.) do not need to be converted to a real life unit. On the simulations, the peers have a computational power that follows a gamma distribution with average of 80 p.u/cycle, with scale 30 and shape 2.

Table 3: Representative Activities for Performance Analysis

Activity Name	Task Cost (p.u.)	Task Size (MB)
LCMS - Low Cost Medium Size	4,000	6
LCBS – Low Cost Big Size	4,000	12
HCSS – High Cost Small Size	32,000	1.5
MCSS – Medium Cost Small Size	16,000	1.5

Figure 3shows the speed up curve of parameter sweep parallelization of the selected activities varying the churn events. It shows the scenarios without churn events, with a 5% and 10% frequency in a Poisson distribution. The first scenario is unrealistic, but it presents a baseline for measurements. In the first graphic, even without churn, activities with tasks of larger size do not scale very well. The peer that submits the activity is the responsible for transmitting all the data to the other selected execution peers. Transmitting several huge tasks may overload the submitter peer network bandwidth. Thus, the data set delivery takes more time, slowing down the overall execution process. It seems that involving fewer peers in the execution is more convenient, since, with the same data set, a peer may run different executions just by assigning a different set of parameters. Therefore, scheduling several tasks of an activity to a small set of peer saves data transmissions and speeds up the overall execution.

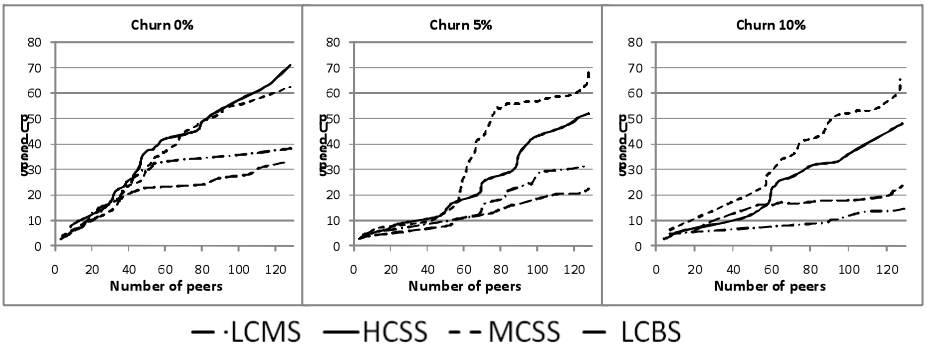

Figure 3: Statistical results for parameter sweep for the selected activities

The scenarios with churn show a decrease in speed up, especially on the activities with larger size. The larger the number of nodes involved, the higher is the chance of a churn in the executor nodes group. When a churn happens, the task needs to be rescheduled and, possibly, the task data set needs to be retransferred to another peer, if the new peer does not have it. Since smaller tasks are easier to transport in the network, activities HCSS and MCSS suffer less impact from churn events because its tasks have only 1.5MB. LCMS and LCBS, in contrast, suffer more impact since their tasks have 6MB and 12MB, respectively.

Figure 4 shows the speed up results for the same activities, but now considering the data fragmentation scenario. Different from the parameter sweep case, each task of the activity has a specific data set. Thus, involving fewer peers in the execution may not be the best strategy. In Figure 4, the speed up rate is positive in all scenarios. However, the impact of churn events is still clear. Compared to the parameter sweep cases, the data fragmentation is more sensitive to churn events. This is reasonable since, when a parameter sweep task is rescheduled, the chosen node has possibly the necessary data packages to process the new assigned task. On a data fragmentation task, though, the data set necessarily has to be retransmitted.

Just like in the parameter sweep case, activities with smaller tasks (HCSS and MCSS) also scale better with churn. The activity cost seems to have little influence in performance, since data transmission appears to be the major bottleneck. However, it is possible to assess that MCSS activity scales better than HCSS.

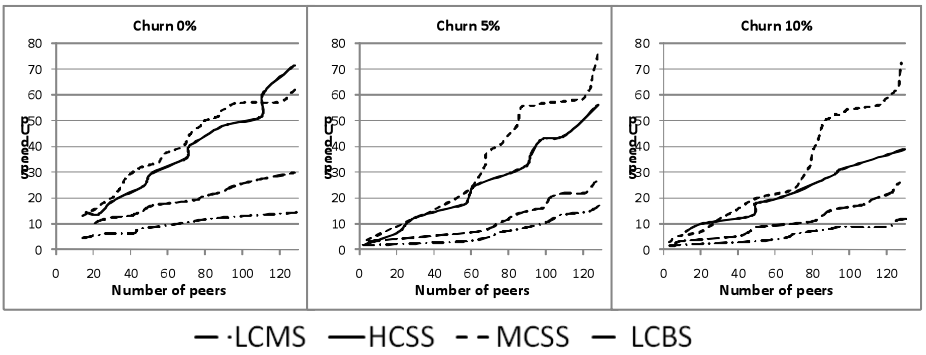

Figure 4: Statistical results for data fragmentation for the selected activities

These initial results show that P2P networks are suitable environments to distribute workflow activities. However, the results also suggest that a lot of improvements need to be made in the scheduling and data transmission mechanisms. It is important to optimize the ideal number of peers involved in the processing of an activity, the choice of the more suitable node to receive a rescheduled task and a better data distribution to not overload the bandwidth of some peers. To minimize churn effects, it is possible to use replication of data sets on other available nodes. When a churn happens, the task can be quickly rescheduled to nodes that already have a replica.

5 Conclusions

SciMule is a hierarchical P2P architecture that provides very good scalability for distribution of workflow activities in heterogeneous environments. We conducted a simulation to evaluate SciMule architecture and obtained positive results regarding the overall performance. From our initial results, we believe that hierarchical P2P is a promising approach to deal with MTC on heterogeneous environments such as hybrid clouds.

From the simulation results, we observed that different parallelization methods should have different approaches for execution. In the case of data fragmentation, it is preferable to have a large number of tasks of small size, than small numbers of tasks of larger size. This is promising since data fragmentation parallelism allows this kind of optimization. In the case of parameter sweep, it is preferable to restrict the number of nodes involved to avoid unnecessary data transfer and minimize the impact of churn events.

Finally, we observed that some issues must be addressed in future work, such as: improving the scheduling mechanism to choose a better number of peers to process an activity of a given size; and a better data discovery and distribution system to not overload the bandwidth of the submitter peers. The scheduler should consider data migration costs before scheduling a task to a node that do not have the data set. Many improvements are still possible to be explored using the SciMule simulator, but we have observed that our strategy is promising and may be an alternative to other heterogeneous distributed environments.

Acknowledgements

The authors are grateful to the High Performance Computing Center (NACAD-COPPE/UFRJ) where the experiments were performed.

References

[1] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, 2009, pp. 528-540.

[2] R.D. Jarrard, *Scientific Methods*, Online book: Url.: http://emotionalcompetency.com/sci/booktoc.html, 2001.

[3] I. Raicu, I. Foster, and Yong Zhao, "Many-task computing for grids and supercomputers," *Workshop on Many-Task Computing on Grids and Supercomputers*, Austin, Texas: 2008, pp. 1-11.

[4] D. Anderson, "BOINC: a system for public-resource computing and storage," *Proceedings. Fifth IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA: 2004, pp. 10, 4.

[5] WCG, *World Community Grid*, http://www.worldcommunitygrid.org, 2009.

[6] Grid4All Consortium, "Towards Hybrid Clouds - A Grid4All perspective on cloud computing," White paper. www.grid4all.eu: 2009.

[7] J. Freire, D. Koop, E. Santos, and C.T. Silva, "Provenance for Computational Tasks: A Survey," *Computing in Science and Engineering, v.10*, 2008, pp. 11-21.

[8] E. Pacitti, P. Valduriez, and M. Mattoso, “Grid Data Management: Open Problems and New Issues,” *Journal of Grid Computing*, vol. 5, 2007, pp. 273-281.

[9] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, “A Survey and Comparison of Peer-to-Peer Overlay Network Schemes,” *IEEE Communications Surveys and Tutorials*, vol. 7, 2005, pp. 72-93.

[10] P. Ganesan, K. Gummadi, and H. Garcia-Molina, “Canon in G Major: Designing DHTs with Hierarchical Structure,” *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, Tokyo, Japan: 2004, pp. 263-272.

[11] L. Garcés-Erice, E.W. Biersack, P.A. Felber, K.W. Ross, and G. Urvoy-Keller, “Hierarchical Peer-to-peer Systems,” *Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing*, Klagenfurt, Austria: 2003, pp. 643-657.

[12] C. Mastroianni, D. Talia, and O. Verta, “A Super-Peer Model for Building Resource Discovery Services in Grids: Design and Simulation Analysis,” *Advances in Grid Computing - EGC*, Amsterdam, The Netherlands: 2005, pp. 132-143.

[13] I. Martinez-Yelmo, R. Cuevas, C. Guerrero, and A. Mauthe, “Routing Performance in a Hierarchical DHT-based Overlay Network,” *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 508-515.

[14] E. Ogasawara, D. Oliveira, F. Chirigati, C.E. Barbosa, R. Elias, V. Braganholo, A. Coutinho, and M. Mattoso, “Exploring many task computing in scientific workflows,” *MTAGS 09*, Portland, Oregon: ACM, 2009, pp. 1-10.

[15] M.E. Dick, E. Pacitti, and B. Kemme, “Flower-CDN: a hybrid P2P overlay for efficient query processing in CDN,” *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, Saint Petersburg, Russia: ACM, 2009, pp. 427-438.

[16] R. Akbarinia, E. Pacitti, and P. Valduriez, “Data currency in replicated DHTs,” *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, Beijing, China: ACM, 2007, pp. 211-222.

[17] M. Karakaya, İ. Körpeoğlu, and Ö. Ulusoy, “A connection management protocol for promoting cooperation in Peer-to-Peer networks,” *Comput. Commun.*, vol. 31, 2008, pp. 240-256.

[18] E.C.D. Almeida, G. Sunyé, Y.L. Traon, and P. Valduriez, “A Framework for Testing Peer-to-Peer Systems,” *Proceedings of the 2008 19th International Symposium on Software Reliability Engineering*, Los Alamitos, CA, USA: 2008, pp. 167-176.

[19] M. Jelasity, A. Montresor, G.P. Jesi, and S. Voulgaris, *The PeerSim simulator*, http://peersim.sourceforge.net, 2010.

[20] D. Freedman, R. Pisani, and R. Purves, *Statistics, 4th Edition*, W. W. Norton, 2007.

[21] A.M. Law, “Statistical analysis of simulation output data: the practical state of the art,” *Proceedings of the 39th conference on Winter simulation*, Washington D.C.: IEEE Press, 2007, pp. 77-83.