Dynamic Network Forecasting using SimGrid Simulations

Matthieu Imbert, Eddy Caron

Dynamic Network Forecasting using SimGrid Simulations

Matthieu Imbert and Eddy Caron
Université de Lyon. LIP Laboratory.
UMR CNRS - ENS Lyon - INRIA - UCBL 5668, FRANCE
E-mail: Prenom.Nom@inria.fr

***Abstract*—In distributed system the knowledge of the network is mandatory to know the available connexion and from the performance point of view. Indeed, to be able to efficiently schedule network transfers on computing platforms such as clusters, grids or clouds, accurate and timely predictions of network transfers completion times are needed. We designed a new metrology and performance prediction framework called Pilgrim which offers a service predicting the completion times of current and concurrent TCP transfers. This service uses SimGrid to simulate the network transfers, and we will show some experimental results comparing the predictions obtained from Pilgrim to the real transfer completion times.**

***Keywords*-Network, Simulation, Forecasting**

I. Introduction

Data management in distributed environments is currently a topic of major interest to the grid and cloud computing community. Indeed, in many application, high performance computing requires to deal with a large amount of data. Moreover the Big Data management is one of challenges for the next years [1]. Sometimes the large data comes from the input data, or it is generated during runtime, or it concerns the output data, and of course all combinations of these three cases exists. Currently, the most widely-used approach for data management for distributed grid computation relies on explicit data transfers between clients and computing servers: the client has to specify where the input data is located, and to which server it has to be transferred. Then, at the end of the computation, the results are eventually transferred back to the client. Some systems such as [2], [3] provide mechanisms to explicitly locate, move, prefetch and replicate data according to the needs of a sequence of computations. It provides the user with an integrated interface to schedule data movement actions just like computational jobs. Again, data location and transfer have to be explicitly handled by the user. Efficient task scheduling may depend on these data management operations: for example is it relevant to move 1TB of data to a more powerful cluster in order to decrease the computing time of 2 hours? If the data transfer will take more than 2 hours, the answer is no.

Moreover, the knowledge of the latency and the bandwidth is not a sufficient information. The dynamicity of the network implies that a network forecast tool should deal with the current load of the network link and the contention ratio. Our objective is to develop a framework providing a network prediction performance service able to dynamically predict the completion times of TCP transfers. Such a service is mandatory for a good resource management system to take scheduling decisions efficiently. In Section II we discuss about the network forecasting issue. Section III presents related works. Section IV introduces the framework of our study before the experiments and results given in Section V.

II. Network forecasting

Different forecasting approaches exists as analytical solution [4], statistical methods [5], machine learning [6], simulation [7], or some combinations of those and appropriate heuristics. Ideally, a network forecaster needs to take into account (implicitly or explicitly) the link capacities, the network equipment capacities (line-cards or back-plane capacity), the interactions both with background traffic and between scheduled transfers themselves. The interactions between network traffics depend on the microscopic behaviors of the network protocols, which in turn, in the case of TCP, may depend on end hosts parameters like congestion window size, congestion avoidance algorithm, and characteristics of the links and equipments on the path, like latencies, packet loss ratios, packets reordering ratios, buffer sizes, etc. Actually all these elements,

combined with the state machine of TCP, generate a complex microscopic behavior which is slow to simulate at the scale of a large computing platform. Packet-level simulations are well suited for offline analysis, when we accept the simulation itself to take longer than the simulated phenomenon. In our case, we need the forecasts to be fast enough to be used in scheduling decisions. A promising approach is to model network transfers at the macroscopic, *i.e.,* flow, level. A flow level TCP model is used in the SimGrid [8] simulator and the speed gain is huge compared to packet level simulators.

We have currently chosen to forecast network transfers using simulations in SimGrid, and we will show that SimGrid simulations performances and accuracy are sufficient to get useful online predictions for the reasonably large-scale platform to which we have access: Grid'5000 [9].

III. RELATED WORK

A. Metrology tools

Existing metrology tools Munin [1] and Ganglia [10] record system metrics of compute nodes or server machines. For network metrology, there are tools like Cacti [2] and Smokeping [3]. Perfsonar [11] is a more advanced network monitoring tool, or actually a framework of tools, targeting large scale network infrastructures.

All these tools come primarily from the *sysadmin* community and are well-suited for tasks of supervision, capacity planning or for diagnosing performance issues. They offer no usable forecasting services and are not well-suited for being used by a resource management system. Indeed, they are geared towards human supervision tasks and their data is not easily accessible programmatically: these data are often stored in RRD files, the de-facto standard in the *sysadmin* community for time-series storage. RRD tools have several advantages, but are mainly designed for nice graphic rendering, not for being easily accessible remotely by other programs.

B. NWS

The NWS, *Network Weather Service* [5] provide accurate forecasts of dynamically changing performance characteristics from a distributed set of metacomputing resources. NWS is setup as a hierarchy of *cliques*, which represent units of measurements. System and network sensors are run inside these cliques and between them. Both system and network sensors use probes, to measure availability of computing or network resources. These probes are active, though they strive to perturbate the platform as little as possible. Forecasts are obtained by using different predictors on each probe time-series, and using an algorithm which continuously selects the best among the set of available predictors. These forecasts are programmatically available through an API.

NWS is the reference for forecasting of computing resource availability in the scheduling community and has no real contender yet. But this project is rather old and does not seem to be maintained anymore. Moreover, the *cliques* map well to hierarchical platforms like grids, composed of a hierarchy of clusters, but may not map so well to more modern platforms like large scale clouds. Also, the fact that probes are active may hinder the deployment of NWS on some platforms.

C. FAST

Relying on NWS, FAST [12] is base on NWS, and offered forecasts of computation times of functions based on benchmarking, at install time, of a representative set of parameters for these functions and polynomial data fitting of actual parameters. FAST suffers from the same drawbacks as NWS. Moreover, it is a complex tool, not easy to setup, and not maintained anymore.

D. Network simulators

Well known packet level network simulators are NS2 [4], NS3 [5], GTNetS [6]. These tools provide very accurate results but they are only interesting in particular situations, for example when we cannot have access to the simulated platform (*e.g.,* because it does not exist) or because we need a very high accuracy. Packet level network simulators are too slow for scheduling purposes, because in all cases, it will be faster to actually perform the network transfers rather than simulate it.

[1] http://munin-monitoring.org/

[2] http://www.cacti.net/

[3] http:oss.oetiker.ch/smokeping/

[4] http://nsnam.isi.edu/nsnam/index.php/Main_Page

[5] http://www.nsnam.org/

[6] http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/

IV. Framework: From simulation to network prediction services

A. The SimGrid simulator

SimGrid [7] is a toolkit that provides core functionalities for the simulation of distributed applications in heterogeneous distributed environments. It offers different programming interfaces MSG, SMPI or SimDag, depending on the formalism level at which we want to describe the simulated applications. In MSG, applications are modeled as a set of processes, running on a set of hosts, executing tasks or exchanging data through the network.

To run a SimGrid simulations, one needs a SimGrid *platform description* which describes the characteristics of the simulated hosts and network.

The SimGrid simulation kernel is based on discrete events evaluations, corresponding to resource state changes (*e.g.,*. start or stop a new computation or a new network transfer). At each event, resource sharing is evaluated, date of the next event is computed, and simulated time is fast-forwarded to the next event. This allows a fast and accurate simulation, provided that the resource sharing model is realistic.

The SimGrid TCP bandwidth sharing model is described in [13] and improved in [14], and models TCP traffic as a RTT-aware Max-Min flow-level model, where the bandwidth allocated to flows competing on a bottleneck link is inversely proportional to the flows' round trip time.

B. The Grid'5000 Platform

Grid'5000 [9] is a scientific instrument for the study of large scale parallel and distributed systems. It is made of more than 1200 compute nodes, totaling more than 5000 cores, distributed among 10 sites in France. See Figure 1.

Grid'5000 provides a set of introspective API which allow to query both its static (resources, network topology) and dynamic (current jobs, resource states, etc) characteristics. These API allow us to get the network topology of Grid'5000 and convert it to a SimGrid platform description.

C. The Pilgrim Metrology Tool

We are currently developing the Pilgrim framework [7], which aims to offer a set of metrology and performance prediction services. These services are implemented as REST-style [15] web-services: transport is HTTP, requests are HTTP GET whose parameters are embedded in the requested URI. Answers to requests are JSON formatted documents.

[7]http://pilgrim.gforge.inria.fr

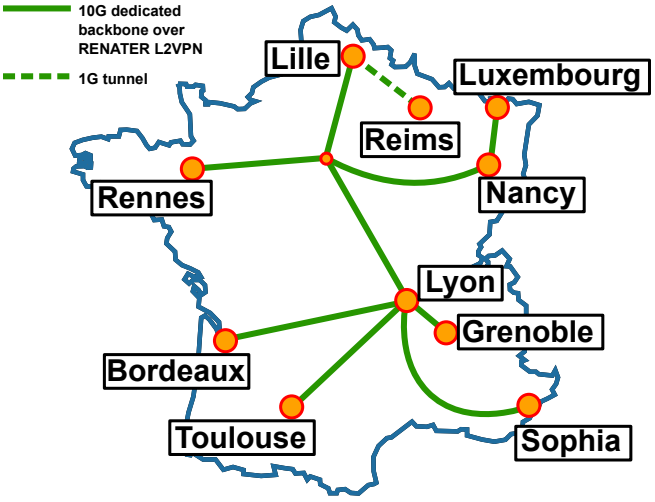

Figure 1. Grid'5000 overview

1) Pilgrim metrology service: Most existing metrology tools do not provide any network-transparent API to programmatically query their data. Thus the first service of the Pilgrim framework is a remote API for accessing RRD files. This allows getting some metrics from tools like Ganglia, Munin, Cacti, Smokeping. We have developed a thin web-service API for serving the data in RRD files, and hiding the complexities of these files (in particular the multiple precisions and timespans of round-robin archives per RRD file). The API offers the following service: for a given RRD, and for given lower and upper bound timestamps, the service will answer with all metric values between these bounds, automatically gathering the most accurate data from the different round-robin archives available in the RRD files.

In the following example, we request the power consumption metric data for compute node `sagittaire-1`, in `lyon` site (this metric data is stored as a Ganglia custom metric):

```
$ curl "http://localhost/pilgrim/rrd/ganglia\
/Lyon/sagittaire-1.lyon.grid5000.fr/pdu.rrd/\
?begin=2012-05-04%2008:00:00\
&end=2012-05-04%2008:01:00"
[[1336111215, 168.92933333333335],
 [1336111230, 168.88],
 [1336111245, 168.88],
 [1336111260, 168.77599999999998]]
```

2) Pilgrim Network Forecast Service (PNFS): The second and main service offered by Pilgrim is the network performance prediction service. Given a list of 3-uples *(source, destination, size)*, it will answer with the list of 4-uples *(source, destination, size, predicted TCP transfer completion time)*.

In the following example request to PNFS, we request the predicted transfer completion times for two concurrent transfers, both 500 MBytes, both from compute node `capricorne-36` in `lyon`, one to node `griffon-50` in `nancy`, one to node `capricorne-1` in `lyon`:

```
$ curl "http://localhost/pilgrim/predict_transfers\
/g5k_test?transfer=capricorne-36.lyon.grid5000.fr,\
griffon-50.nancy.grid5000.fr,5e8&\
transfer=capricorne-36.lyon.grid5000.fr,\
capricorne-1.lyon.grid5000.fr,5e8"

[ { "src": "capricorne-36.lyon.grid5000.fr",
    "dst": "griffon-50.nancy.grid5000.fr",
    "size": 500000000,
    "duration": 16.0044 },
  { "src": "capricorne-36.lyon.grid5000.fr",
    "dst": "capricorne-1.lyon.grid5000.fr",
    "size": 500000000,
    "duration": 4.76841 } ]
```

The prediction service implementation relies on having access to a model of the simulated platform. The Grid'5000 *Reference API* offers a static description of the Grid'5000 platform. Each Grid'5000 compute node is described, as well as its network interfaces, the cluster and site to which the node belongs. The network description provided by this API associates to each interface of each compute node its corresponding port on a network equipment. Links between network equipment ports are also described, as well as backbone links, and each port is tagged with its capacity (1 Gbit/s or 10 Gbit/s in most cases). Also, inside each network equipments, line-cards and backplanes capacities are described. All these information are semi-automatically gathered by scripts.

We developed a tool which is able to process this Grid'5000 self-description, and convert it to a SimGrid platform description. The SimGrid platform description is a hierarchical description based on the notion of Autonomous System (AS), which is, from the SimGrid point of view, an independent routing unit. Our Grid'5000 to SimGrid wrapper currently creates one SimGrid autonomous system per Grid'5000 site.

It is worth noting that the notion of Autonomous System was recently added to SimGrid (see [16]) and has greatly improved SimGrid performances regarding routing. Before the introduction of AS, routing was not hierarchical, thus we had to model Grid'5000 as a "flat" platform, leading to a huge routing table which would consume a lot of memory, to the point that it was impossible to wholly simulate Grid'5000. Since the introduction of Autonomous Systems and hierarchical routing, it is possible to efficiently simulate the whole platform.

In addition to link capacities, latencies are also needed for the SimGrid platform description. Currently, we hardcode two latency values when generating the SimGrid platform, one for intra-site links (10^{-4}s) and one for backbone latencies ($2.25.10^{-3}$s). In the future, we will get these latencies from periodic measures in SmokePing or Cacti, thanks to the Pilgrim metrology service.

Having the SimGrid model of Grid'5000, then, for each network forecast request, a SimGrid simulation is instantiated, containing one send and one receive process for each requested transfers. These processes do nothing except sending the data and waiting for it, and tracking the transfer completion time in the simulated world.

Currently, a typical request to a local Pilgrim instance (minimizing impact of network latency) for a prediction involving 30 concurrent transfers on Grid'5000 takes less than 0.1s.

V. Experiments

A. Experimental protocol

To validate the Pilgrim network forecasts service, we have run several experiments on Grid'5000. Each experiment consists in starting simultaneously actual TCP transfers between several Grid'5000 compute nodes, recording their completion time, and comparing the results with the Pilgrim forecasts. Each experiment depends on a set of parameters:

- Transfer size: 10 values on a geometrical progression between 0.1 MByte and 10 GBytes.
- Number of transfer sources: 1, 10, 30, 50 or 60.
- Number of transfer destinations: 1, 10, 30, 50 or 60.
- When $n_{sources} < n_{destinations}$, some will be source of more than one TCP transfer. When $n_{sources} > n_{destinations}$, some will be destination of more than one TCP transfer.
- Two Topologies:

- CLUSTER: all source and destination nodes are randomly drew from a single random cluster
- GRID_MULTI: all sources and destinations are randomly drew from all clusters and sites of Grid'5000. With the constraint that all transfers are across Grid'5000 site boundaries.

We have tested predictions based on two Grid'5000 platform descriptions:

g5k_cabinets shipped with SimGrid, built automatically by a script, based on the stable Grid'5000 *Reference API* which provides some basic network topology information.

g5k_test built by a Pilgrim script, based on the development version of Grid'5000 *Reference API* which provides a more detailed network topology information.

g5k_test is less optimized than *g5k_cabinets* (in size and loading time), because it does not abstract clusters and instead it enumerates all hosts, but it actually conforms more to the reality and we have found that all predictions based on *g5k_test* are better. In the results, we only present results with *g5k_test*. Currently, the generated SimGrid platform description does not yet contain network equipments bandwidth limits (linecard or backplane capacity).

The network topology description available in the development version of Grid'5000 *Reference API* is currently still a work in progress, and only available for three Grid'5000 sites: Lille, Lyon and Nancy. Thus, our experiments were limited to nodes and clusters from these three sites.

For a given parameter combination, each experiment is run 10 times and results are aggregated, in order to average the results, and especially to minimize the impact of perturbations by cross-traffic from other jobs. We also minimize the perturbations of cross-traffic by reserving whole Grid'5000 sites (during nights) when running our experiments.

The actual transfers are performed using *iperf*. Each experiment has the following order:

- TCP *iperf* servers (receivers) are started on all destination nodes.
- TCP *iperf* clients (senders) are simultaneously started on all source nodes, each transferring the same amount of data to its destination.
- Wait the end of the client transfers, record the completion time of all actual transfers.
- Record the Pilgrim predictions.

The automated execution of all these steps is performed using the Execo tool [8], which allows powerful scripting of the experiments in python.

Sender and receiver hosts are all running the exact same Linux Debian stable version 6.0 "squeeze". Kernel is 2.6.32-5-amd64. The TCP stack uses CUBIC TCP congestion control with HyStart disabled and the following tuning:

```
net.ipv4.tcp_rmem = 4096 87380 4194304
net.ipv4.tcp_wmem = 4096 65536 4194304
net.core.rmem_default = 87380
net.core.wmem_default = 65536
net.core.rmem_max = 4194304
net.core.wmem_max = 4194304
```

SimGrid is configured with option *network/TCP_gamma* = 4194304 to match the configured maximum congestion window size of the kernel network stack.

B. Results

We present here only a subset of our results. The aim of the plots is to show the comparison between the actual transfer completion times and the predictions. For each transfer, we define the error as $log_2(prediction) - log_2(measure)$. In the plots, the *error* line joins the medians of the errors for increasing transfer sizes. Error dispersion is showed by the boxes.

Additionally, to visually get a better insight of the evolution of transfer duration as a function of transfer size, we also draw on the right axis of the same plots the median of the actual transfer completion times.

1) Cluster experiments: The *sagittaire* cluster in Lyon is made of 79 nodes, dual-cpu single-core Opteron 250, 2.4 Ghz, 2 GBytes RAM. The gigabit ethernet cards of all nodes are connected directly to the main Lyon switch/router, which is an Extreme-Networks BlackDiamond 8810. See Figure 2.

The results for this cluster are shown in figures 3, 4, 5.

The *graphene* cluster in Nancy is made of 144 nodes, mono-cpu quadri-core Xeon X3440, 2.5 Ghz, 16 Gbytes RAM. The gigabit ethernet card of all nodes are connected in four groups to four aggregation switches, which are connected to the main Nancy

[8] http://execo.gforge.inria.fr

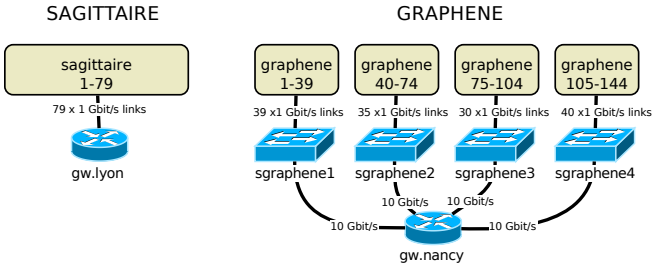

Figure 2. sagittaire and graphene clusters network

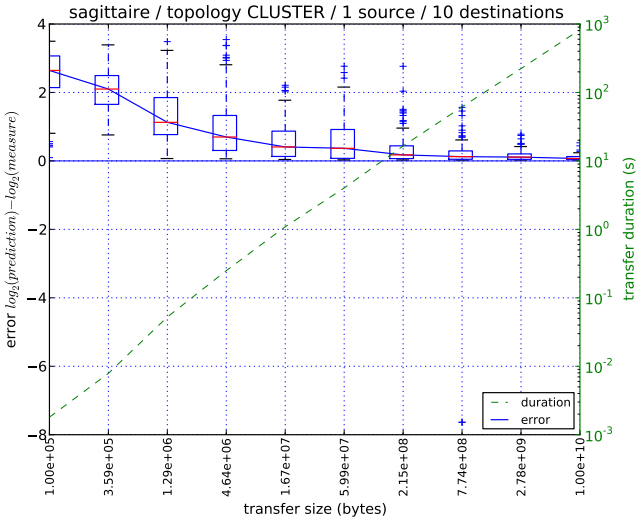

Figure 3. Predictions vs. measures for transfers on cluster sagittaire from 1 source to 10 destinations

router (see Figure 2). The results for this cluster are shown in figures 6, 7, 8, 9.

These results show that SimGrid's TCP model is accurate for large transfers, $size > 1.67 \cdot 10^7 bytes$. Its TCP model is not as good for smaller transfers. As was already showed in [13], [14], this is mainly due to the improper modelization of TCP slow-start. This is not a problem for us, as we aim to use this network forecast service for scheduling, and we think scheduling small transfers is unneeded. Different cluster topologies have some impact on the forecast quality: the flat topology of sagittaire gives the best results, whereas the hierarchical topology of graphene gives worse results for small transfers. For large transfers, prediction and actual measures converge almost identically in both clusters, except that whereas TCP bandwidth sharing model is very accurate for sagittaire, it does not work so well for graphene, as can be seen in experiments 30 sources to 30 destinations,

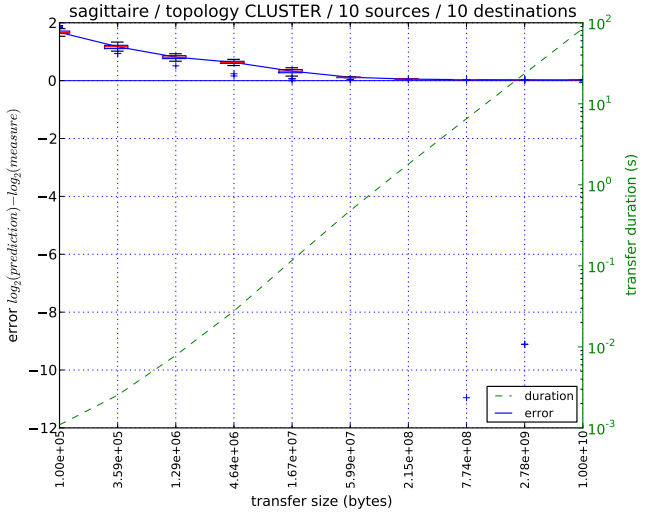

Figure 4. Predictions vs. measures for transfers on cluster sagittaire from 10 sources to 10 destinations

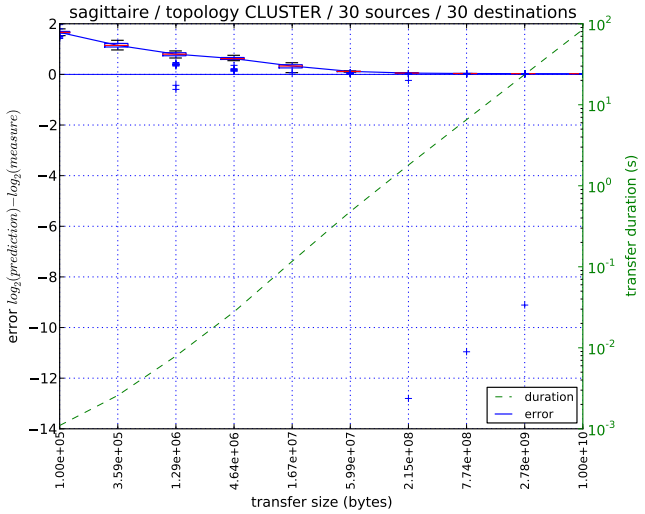

Figure 5. Predictions vs. measures for transfers on cluster sagittaire from 30 sources to 30 destinations

where for $size > 1.67 \cdot 10^7 bytes$, predictions are greater than reality by a constant factor ≈ 1.25. For experiment 50 sources to 50 destinations, this constant factor reaches ≈ 1.7.

The constant factor between predictions and measures for graphene, for a number of transfers ≥ 30 is the most annoying result, since we don't know yet how to explain it:

- We think it's not caused by the lack of modelization of the network equipment capacities, since it would cause the predictions to be lower than

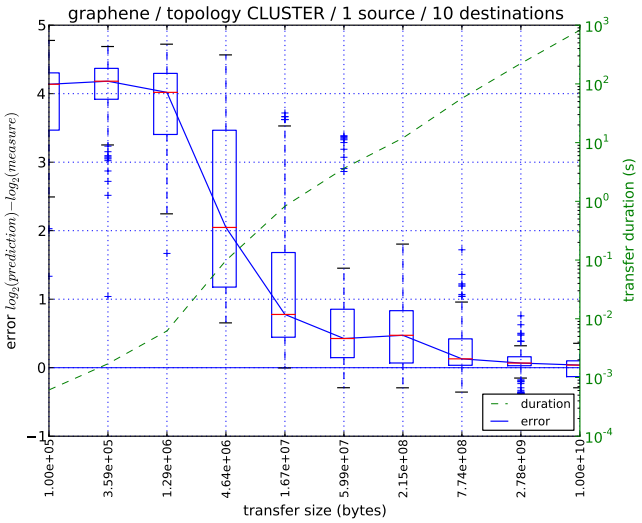

Figure 6. Predictions vs. measures for transfers on cluster graphene from 1 source to 10 destinations

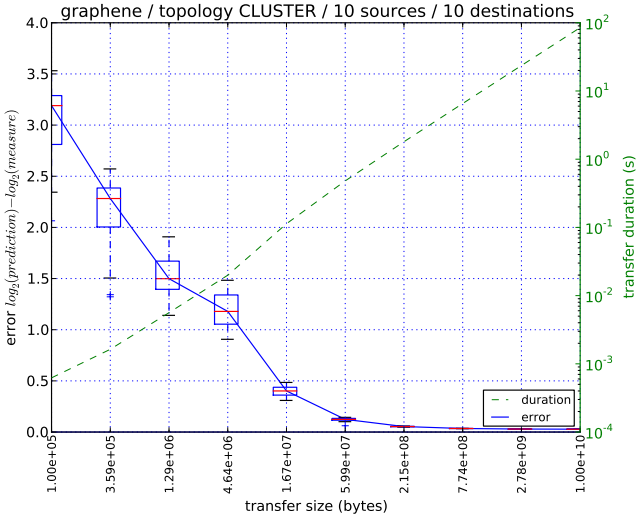

Figure 7. Predictions vs. measures for transfers on cluster graphene from 10 sources to 10 destinations

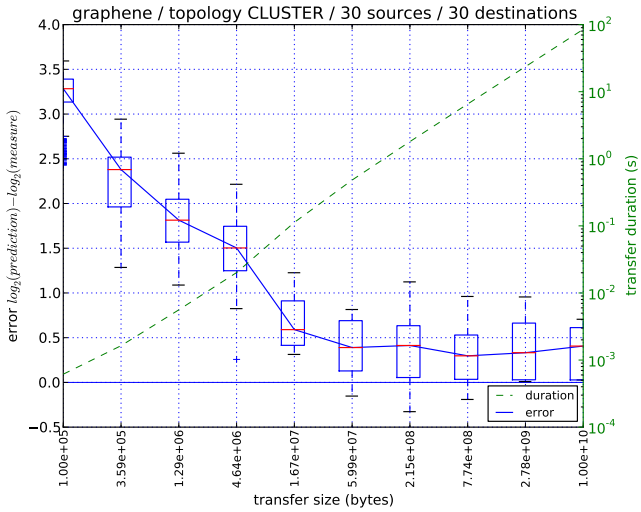

Figure 8. Predictions vs. measures for transfers on cluster graphene from 30 sources to 30 destinations

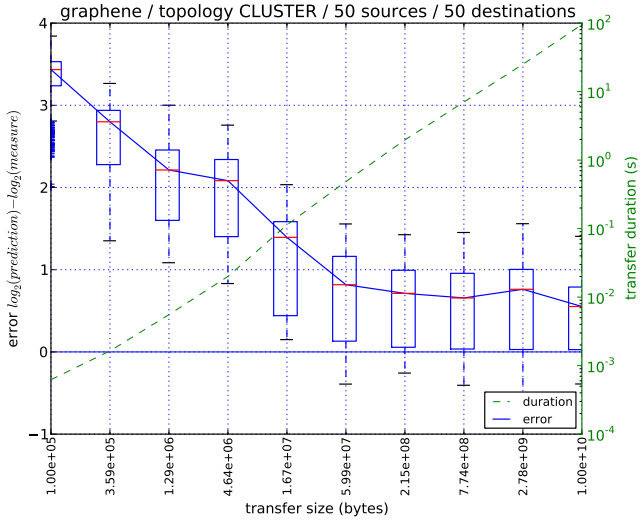

Figure 9. Predictions vs. measures for transfers on cluster graphene from 50 source to 50 destinations

measures.

- Some measures (not reproduced here) show that predictions and actual transfers on graphene, from 50 sources to 30 destinations or from 30 sources to 50 destinations, converge more nicely than 30 to 30 or 50 to 50.

2) Grid experiments: These experiments involve nodes distributed on three Grid'5000 sites in France: Lille, Lyon and Nancy. All the transfers are done across the grid, between different sites, so all transfer paths go through cluster links, cluster aggregation links, backbone links, and travel through several (a minimum of 3) network equipments. Additionally to the clusters already described for cluster experiments, nodes may be randomly chosen on other clusters of Lille, Lyon and Nancy. We don't detail all these clusters here, but these clusters are similar, and their software configuration is identical. Results are shown in figures 10 and 11.

These results show that at the grid scale, the forecasts are still relevant, and we see the same limitations

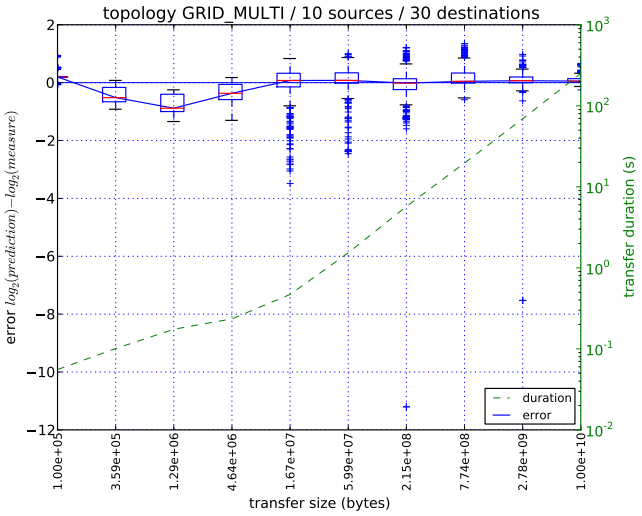

Figure 10. Predictions vs. measures for transfers on Grid'5000 from 10 sources to 30 destinations

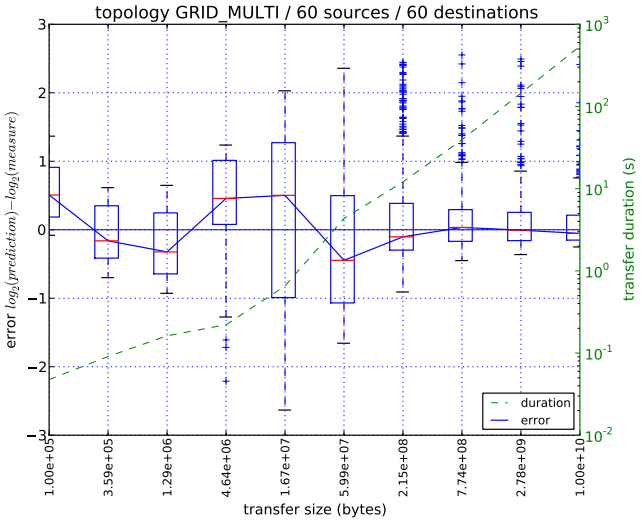

Figure 11. Predictions vs. measures for transfers on Grid'5000 from 60 sources to 60 destinations

for small transfer sizes.

Globally, if we consider all the results presented here, both in cluster and grid topologies, and if we consider only results for transfer whose $size > 1.67 \cdot 10^7 bytes$, the median of the absolute value of all the errors is 0.149, with a standard deviation of 0.532. This corresponds to half of the predictions differing from the measure by no more than a factor 0.11. 74% of the predictions have an absolute error less than 0.575, which corresponds to predictions differing from the measure by no more than a factor of 0.5.

VI. Conclusion and Future Works

The Pilgrim service, relying on SimGrid simulations, is able to accurately predict network transfer completion times for various network topologies, ranging from low-latency flat clusters to Grid'5000, a grid with a medium latency, country-wide, backbone connecting hierarchical sites. Network contentions are correctly handled.

We will try to improve the generation of the Grid'5000 simgrid platform model: grouping hosts in clusters, model the network equipments limitations (backplane, linecards), use automatic link latency measurements instead of arbitrary values.

We also plan to model the background traffic of Grid'5000, thanks to the ongoing work on this platform's network instrumentation. Of course, we will have to find a tradeoff between a very accurate dynamic model of the platform involving too much data (for example: current traffic of all network interfaces of all nodes and network equipments), or a coarse model.

More *clever* services could also be added to Pilgrim, *e.g.,* given n different transfer hypotheses, select the fastest one. As Pilgrim has some knowledge of the platform, it could use some heuristic to prune the n hypotheses and only simulate a subset of them, before returning an answer. In the future we plan to add some service which will not only forecast network transfers but also full workflows involving computations and network transfers. This is another reason why we chose SimGrid, as adding the simulation of computation will be straightforward.

The full set of our experiments (from which we have only showed a subset in this article) validates the network model of SimGrid, and at the same time allows further improvements of this network model, especially for the TCP slow start phase.

Finally, the drawback of a simulation based approach to performance prediction are that it may be difficult to scale to very large scale platforms, and also that it relies on having access to a detailed description of the platform, which is not always available. Thus other approach to performance predictions, like machine learning, are also worth studying.

Acknowledgment

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development

action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr).

This work was done through the financial support of SED (*Service Expérimentation et Développement*), INRIA Rhône-Alpes.

REFERENCES

[1] C. Lynch, "Big data: How do your data grow?" *Nature*, vol. 455, no. 7209, pp. 28–29, 09 2008. [Online]. Available: http://dx.doi.org/10.1038/455028a

[2] T. Kosar and M. Livny, "Stork: Making Data Placement a First Class Citizen in the Grid," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings*, 2004, pp. 342–349.

[3] F. Desprez, E. Caron, and G. Le Mahec, "Dagda: Data arrangement for the grid and distributed applications," in *AHEMA 2008. International Workshop on Advances in High-Performance E-Science Middleware and Applications. In conjunction with eScience 2008*, Indianapolis, Indiana, USA, décembre 2008, pp. 680–687.

[4] E. Caron, F. Desprez, M. Quinson, and F. Suter, "Performance evaluation of linear algebra routines," *International Journal of High Performance Computing Applications*, vol. 18, no. 3, pp. 373–390, 2004.

[5] R. Wolski, N. Spring, and J. Hayes, "The Network Weather Service: A distributed resource performance forecasting service for metacomputing," *The Journal of Future Generation Computing Systems*, vol. 15, no. 5-6, pp. 757–768, 1999.

[6] E. Ipek, B. R. de Supinski, M. Schulz, and S. A. McKee, "An approach to performance prediction for parallel applications," in *Euro-Par 2005, Parallel Processing, 11th International Euro-Par Conference (11th Euro-Par'05)*, ser. Lecture Notes in Computer Science (LNCS), J. C. Cunha and P. D. Medeiros, Eds., vol. 3648. Lisbon, Portugal: Springer-Verlag (New York), Aug.-Sep. 2005, pp. 196–205.

[7] H. Casanova, A. Legrand, and M. Quinson, "SimGrid: a Generic Framework for Large-Scale Distributed Experiments," in *10th IEEE International Conference on Computer Modeling and Simulation*, Mar. 2008.

[8] H. Casanova, "Simgrid: A toolkit for the simulation of application scheduling," in *CCGRID*. IEEE Computer Society, 2001, pp. 430–441.

[9] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard, "Grid'5000: A large scale and highly reconfigurable grid experimental testbed," in *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, ser. GRID '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 99–106. [Online]. Available: http://dx.doi.org/10.1109/GRID.2005.1542730

[10] M. Massie, B. Chun, and D. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, Jul. 2004.

[11] A. Hanemann, J. Boote, E. Boyd, J. Durand, L. Kudarimoti, R. Łapacz, D. Swany, S. Trocha, and J. Zurawski, "Perfsonar: A service oriented architecture for multi-domain network monitoring," *Service-Oriented Computing-ICSOC 2005*, pp. 241–254, 2005.

[12] M. Quinson, "Dynamic Performance Forecasting for Network-Enabled Servers in a Metacomputing Environment," in *International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS'02), in conjunction with IPDPS'02*, Apr 2002.

[13] H. Casanova and L. Marchal, "A Network Model for Simulation of Grid Application," INRIA, Rapport de recherche RR-4596, 2002. [Online]. Available: http://hal.inria.fr/inria-00071989

[14] P. Velho and A. Legrand, "Accuracy study and improvement of network simulation in the simgrid framework," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 13.

[15] R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[16] L. Bobelin, A. Legrand, M. David, Alejandro González, P. Navarro, M. Quinson, F. Suter, and C. Thiery, "Scalable Multi-Purpose Network Representation for Large Scale Distributed System Simulation," INRIA, Rapport de recherche RR-7829, décembre 2011. [Online]. Available: http://hal.inria.fr/hal-00650233