Modeling Distributed Platforms from Application Traces for Realistic File Transfer Simulation

Anchen Chai[1,2] Mohammad-Mahdi Bazm[1,2] Sorina Camarasu-Pop[1]
Tristan Glatard[4] Hugues Benoit-Cattin[1] Frédéric Suter[2,3]

[1] Université de Lyon, CREATIS,
CNRS UMR5220, Inserm U1044, INSA-Lyon, Université Lyon 1
[2] IN2P3 Computing Center, CNRS
[3] Inria Avalon project team
[4] Department of Computer Science and Software Engineering
Concordia University

Abstract

Simulation is a fast, controlled, and reproducible way to evaluate new algorithms for distributed computing platforms in a variety of conditions. However, the realism of simulations is rarely assessed, which critically questions the applicability of a whole range of findings.

In this paper, we present our efforts to build platform models from application traces, to allow for the accurate simulation of file transfers across a distributed infrastructure. File transfers are key to performance, as the variability of file transfer times has important consequences on the dataflow of the application. We present a methodology to build realistic platform models from application traces and provide a quantitative evaluation of the accuracy of the derived simulations. Results show that the proposed models are able to correctly capture real-life variability and significantly outperform the state-of-the-art model.

1 Introduction

Numerical simulation has become a critical tool to study distributed systems. Not only does it allow researchers to quickly evaluate new algorithms in a variety of conditions with literally no cost, but it also enables reproducible experiments whereas real systems are hampered by various uncontrolled factors such as background network traffic, concurrent CPU usage, and hardware faults. Simulation also provides access to variables that could hardly be monitored in real systems, for instance network congestion.

Simulating application executions requires several components to complement the core simulation kernel. Simulators usually build on models of (i) the hardware platform, (ii) the software services deployed on the platform, e.g., schedulers, and (iii) the application itself, i.e., the workload characteristics. Each aspect needs to be accurately validated since it may have a serious impact on the simulated performance.

In practice, however, the realism of simulations is rarely assessed, which critically questions the veracity of a whole range of findings derived from simulations. Even more importantly, studies such as [21] have shown that widely-used simulation toolkits patently lack realism, in particular regarding the estimation of file transfer times across the network.

Yet, file transfers are critical to the performance of distributed applications. The variability of file transfer times increases with the number of storage and computing sites, which has important consequences on the dataflow of the application. A subtle time difference in a file transfer may delay a task, change the schedule of subsequent tasks in the application workflow, impact the data storage policy, and eventually greatly affect the execution time. Therefore, file transfers have a substantial impact on the realism of simulations.

This paper presents our efforts to build platform models that accurately simulate file transfers across a distributed infrastructure of computing and storage resources. It proposes a methodology to build platform models from application traces and it provides a quantitative evaluation of the accuracy of the derived simulations. The methodology highlights and provides a detailed analysis of the phenomena involved in file transfer simulation on large distributed systems.

We focus on the platform as defined in the SimGrid toolkit [9], which includes the compute nodes and the network links and routes. The network topology is an important part of the platform that describes how resources are clustered into sites (called Autonomous Systems or ASes [3]) and how sites are interconnected, possibly in a hierarchical way. The platform model also describes the resource properties, including their processing speed, network bandwidth and latency, or sharing policy. Other simulation toolkits may have different representations for platforms, but they usually include (i) a set of computation and communication resources and (ii) the network interconnection. OptorSim [2] and SimGrid use external text files for the platform description while GridSim [5] defines the platform directly in the code of the simulator.

We adopt a trace-based approach where platform models are built strictly from application logs, i.e., without requiring specific monitoring probes. Compared to invasive approaches such as network tomography [19], the trace-based approach might be less accurate but also easier to use on infrastructures where probing is not possible. Ultimately, we aim at replaying application traces as accurately as possible in a simulated environment. In this paper, we rely on a corpus of traces extracted from the Virtual Imaging Platform (VIP) [12], a web portal for medical image analysis and simulation.

The paper is organized as follows. In Section 2, we present the real system that we aim at simulating, the traces that will be used to define the simulated platform, and the simulation framework we developed based on the SimGrid toolkit. Section 3 presents the platform construction methodology, i.e., techniques to estimate network bandwidths from application traces. Starting from a rough platform model that represents the state of the art used in works such as [8], we present six successive improvements that we motivate through illustrative cases extracted from the trace corpus. Section 4 presents the final simulation results that are evaluated against the state-of-the-art platform model. All the code and data used in this article are available online, to allow readers and reviewers to reproduce and further investigate our results [10]. Section 5 presents related work on simulation, platform models, trace-based approaches, and simulation evaluation. Finally, Section 6 summarizes our findings and details our future work.

2 Technical Context and Traces

2.1 Real System

Our target system is the distributed execution of an application submitted through the VIP portal. VIP gives access to several applications while completely hiding the underlying storage and computing infrastructure. It leverages the resources of the Biomed Virtual Organization (VO) within the European Grid Infrastructure (EGI). This VO is supported by about 65 sites world-wide, offering a total of 130 computing clusters and 5 PB of storage. The high variability of the static and dynamic characteristics of these heterogeneous resources scattered across multiple organizations makes it particularly challenging to build realistic models for such a system.

In this work, we focus on the GATE application [14], whose workflow is described in Figure 1. In the *computing* phase, a set of independent workers: (1) download the input files; (2) execute a single GATE job each; and (3) upload partial results on a storage element (SE) upon completion. Once all the GATE jobs are completed, the workflow enters a *merging* phase, where all the partial results are downloaded from different SEs (4) and aggregated (5). The final result is then copied back to a SE (6). The data exchanges between the tasks of the workflow are file-based and require transfers over the network and through SEs, since the worker nodes do not share a common file system.

2.2 Execution Traces

We collected execution traces of 60 workflow executions submitted by 14 distinct VIP users over a period of one month. These workflows are all different in terms of software release used, input files, number of jobs, and execution times. However, they do not all match the current capabilities of our simulator [18], which assumes that all the partial results files produced by the GATE jobs are downloaded by the Merge

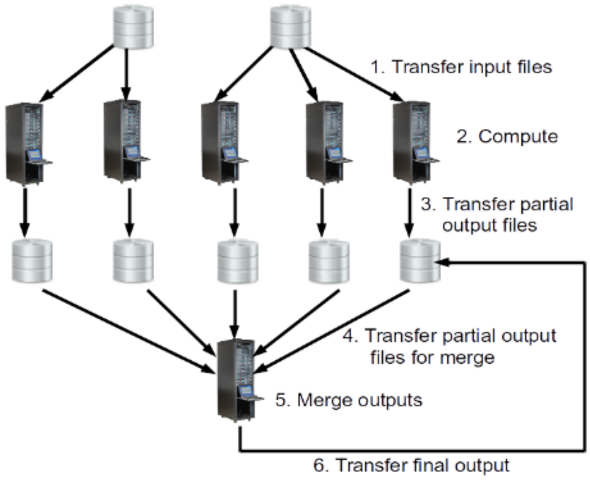
1. Transfer input files
2. Compute
3. Transfer partial output files
4. Transfer partial output files for merge
5. Merge outputs
6. Transfer final output

Figure 1: GATE workflow including a computing phase, a merging phase, and associated file transfers.

jobs, and only them. In some workflows, this assumption is not verified due to canceled or failed jobs. We thus discarded 31 out of the initial 60 workflows where a majority of the logs files were empty (1/60), there were less (22/60) or more (1/60) downloads than uploads, or some downloaded files are not those uploaded by completed jobs (7/60). We also discarded 5 workflows that have been severely impacted by the scheduled maintenance shutdown of a specific storage element. It left us with 24 workflows submitted by 6 distinct users from Sep. 8, 2015 to Oct. 1, 2015.

In these 24 workflows, 1,796 jobs were executed by worker nodes across 32 different computing sites, which represents almost half of all the sites in the Biomed VO. These jobs performed 8,932 file transfers using 32 different storage elements.

Each job creates its own structured log file. It starts with a header that comprises the *name of the workflow* the job belongs to and the *job id*. Then, information on the compute node that runs the job is logged, i.e., its *name*, *number of cores*, *processing speed*, and the *bandwidth* of the network interface.

The main steps of a job execution are listed as well. In the following, we present the steps for GATE job. First, the job tests whether it can reach its default SE by doing an *upload test*, i.e., a copy-and-register operation of a 12-byte file. Second, the job *downloads* the needed input files: (i) a wrapper script automatically created (73 kB); (ii) a tarball containing the GATE executable along with all the required libraries needed for execution (from 121.6 MB to 501.8 MB); (iii) a tarball containing the user-defined inputs, e.g., macro files describing the simulation, images, or ROOT files (from 4 kB to 130 kB). Third, the *Monte-Carlo simulation* is executed. Fourth, the job *uploads* its result file to its *default SE*. For all the occurring transfers, the *source*, *destination*, *size*, and *duration* are logged.

The second source of information is a database, which contains a timestamp for each milestone on the job execution timeline (i.e., creation, entering the queue, starting the download, computation, and upload phases, and termination).

We developed the `log2sim` framework [17] to parse these log files and the database and then produce three CSV files. One describes the worker nodes, the second the file transfers, and the third lists the milestones of the execution. We then use these CSV files to to generate and instantiate a *model of the platform* which will be an input of our simulator.

2.3 The VIP Simulator and the SimGrid Toolkit

We developed a simulator that implements several VIP services to simulate the execution of a GATE workflow. It assumes that the tasks composing the workflow have already been created and submitted and that the compute resources have been acquired. It also implements several services from EGI related to data transfers. They are designed to be as close as possible to the actual behavior of the production system. A central *Logical File Catalog* (LFC) stores information about file replicas. To download a file a worker node first has to contact the LFC to know from which SE the requested file can be retrieved. To ensure that simulated file transfers occur between the same hosts as in the real execution, we enforce the use of the file replica involved in the corresponding real transfers. Then the worker node contacts the targeted SE to download the file. Conversely, a worker uploads a file to a SE and then notifies the LFC to register its location. For all file transfers, and based on an analysis of actual transfer durations, our simulator models these interactions between the different services by a 900ms delay that encompasses to the cost of processing transfer requests and storage access latency in addition to the network latency.

This simulator is based on the SimGrid toolkit [9], that provides all the core functionalities for the simulation of distributed applications in heterogeneous distributed environments. SimGrid relies fast and tractable *fluid* models, whose validity has been thoroughly assessed [21], to simulate network traffic and ensures the realism of the simulated communication times. SimGrid also provides a scalable engine that allows us to simulate large workflows with many concurrent transfers on large infrastructures in a reasonable time.

SimGrid usually requires a model of the platform as input, in our case the EGI infrastructure accessed through VIP. This model describes the characteristics of the compute (i.e., hosts with a certain number of cores and processing speed) and network (i.e., links of a given bandwidth, latency, and contention model) resources. It also describes how hosts are grouped (e.g., in clusters, data centers, ...) and interconnected through a network topology. SimGrid formalism adopts a hierarchical organization based on Autonomous Systems (ASes) similar to that of the Internet [3]. The correct instantiation of this platform model is key to simulation realism. To the best of our knowledge, there exists no full and faithful description of EGI that has been built for simulation purposes.

3 Modeling Platforms from Execution Traces

In this section, we detail incremental improvements to the platform model to increase the accuracy of file transfer simulation thanks to a thorough analysis of both trace contents and simulation results. For each improvement, we motivate and illustrate the impact of the proposed solution on a particular case extracted from the traces.

3.1 Baseline Model

The quality of file transfer simulation is mainly impacted by two main features of a platform model: the interconnection topology and the instantiation of network link bandwidth. To define a baseline for our study, we consider a first platform model that only minimally relies on the execution traces for these two important features. Nevertheless, the characteristics of the computing nodes (e.g., name or processing speed) are extracted from the execution traces, as well as their hierarchical organization in clusters and sites.

Without information on the interconnection topology coming from the traces, we have to assume a simple and uniform connectivity among the compute nodes and the storage that compose the distributed platform. A way to model such a platform is to connect each SE to all the computing entities through a single backbone (Figure 2 (A)).

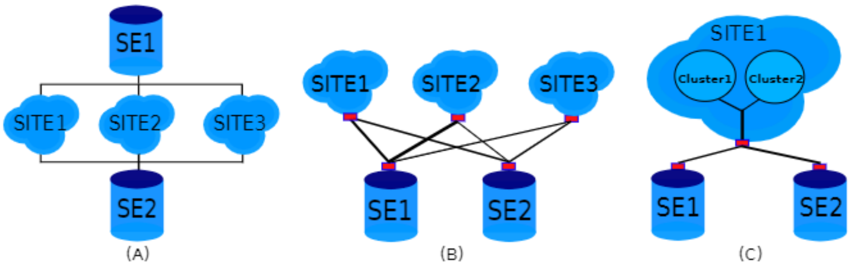

Figure 2: Representation of different platform models. Red boxes represent limiter links.

To instantiate the different links, we use the nominal bandwidth value of the network card for the compute nodes, which is typically of *1 Gb/s*, and a latency of 500 microseconds. This information is extracted from the traces, but does not depend on a specific run. For the storage elements, we consider an interconnection bandwidth of *10 Gb/s* and a network latency of 750 microseconds. The rationale is

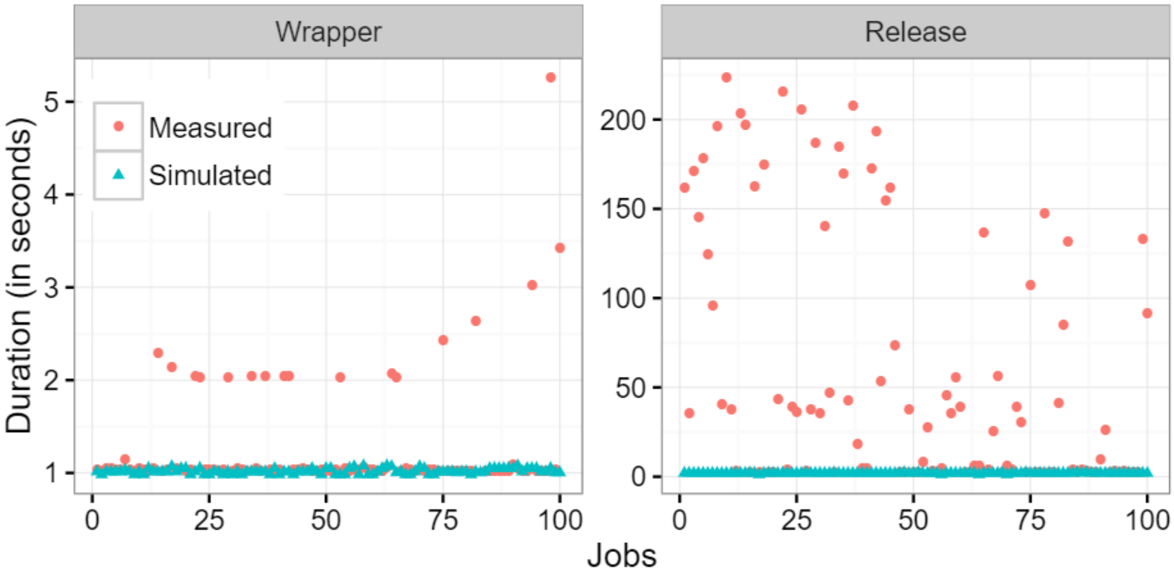

Figure 3: Measured[1] and simulated (with a 10 Gb/s link per SE) transfer durations for two input files downloaded by each job in the `fNUfzs` workflow instance.

that SEs usually are large disk bays, hence with a better connectivity than simple compute nodes to ensure a good throughput. Such a model can be considered as state-of-the-art as it has been used for the simulation of similar workflows running on the same distributed platform [7,8], but in a context where the focus was not on file transfers.

Figure 3 presents the file transfer durations (in seconds) of two of the input files downloaded by each of the hundred jobs composing one specific workflow instance, as logged in the traces and simulated with the aforementioned model. The respective sizes of these two files, which are downloaded in the majority of the studied workflows are 73 kB (wrapper file) and 121.57 MB (GATE release file). These transfers are representative of all the other file transfers for this workflow, but also of all the other studied workflows.

This figure illustrates two major drawbacks of relying on the "theoretical" bandwidth of 10 Gb/s to instantiate the platform model. First, the duration of the file transfers are globally and severely underestimated. For the release file, the simulated durations are in the [1.186s; 2.437s] range, while the measured transfer times are in the [2s; 223.5s] range. Similar differences occur for the wrapper file. Second, and more importantly, this model fails to reproduce the variability because it does not properly capture the competition for resources between certain transfers due to inaccurate transfer time estimations.

3.2 Leveraging Trace Contents to Instantiate the Platform

To address the two issues raised by a uniform bandwidth instantiation of the link that connects a SE to the rest of the platform to a nominal value of 10 Gb/s, we exploit the contents of the execution traces. Our aim is at using values that are more representative of the actual execution conditions. For each link, we consider two common-sense aggregation methods, which have their respective pros and cons. First, we compute the **average** value over all the observed transfers to/from a given storage element. Using average values is likely to be more realistic when it comes to reproduce the behavior of a single workflow execution. It reflects the network connectivity as experienced by the application. The sharing of network resources by concurrent transfers is thus directly captured in the model and not handled by the simulation kernel. The drawback of this approach is that the resulting instantiation of the model is limited to the workflow used to produce it and it cannot be used to simulate the execution of another workflow even though the same resources are involved. However, the accurate simulated replay of a given trace already allows for the investigation of several interesting what-if scenarios. Second, we determine the observed **maximum** value over all these transfers. This allows us to get an approximation of the nominal capacity of the network link. In that case, we let the simulation kernel determine how the network resources are shared among concurrent transfers. The main advantage of such an instantiation of the platform model is that it can be reused beyond the simulated replay of the execution that led to its generation. It can also be aggregated, both spatially and temporally, with information obtained from other traces.

Figure 4 presents the same type of results as Figure 3, but here the simulation results obtained with a uniform 10 Gb/s instantiation have been replaced by those obtained with an average or maximum bandwidth value for each storage element. We use the same interconnection topology.

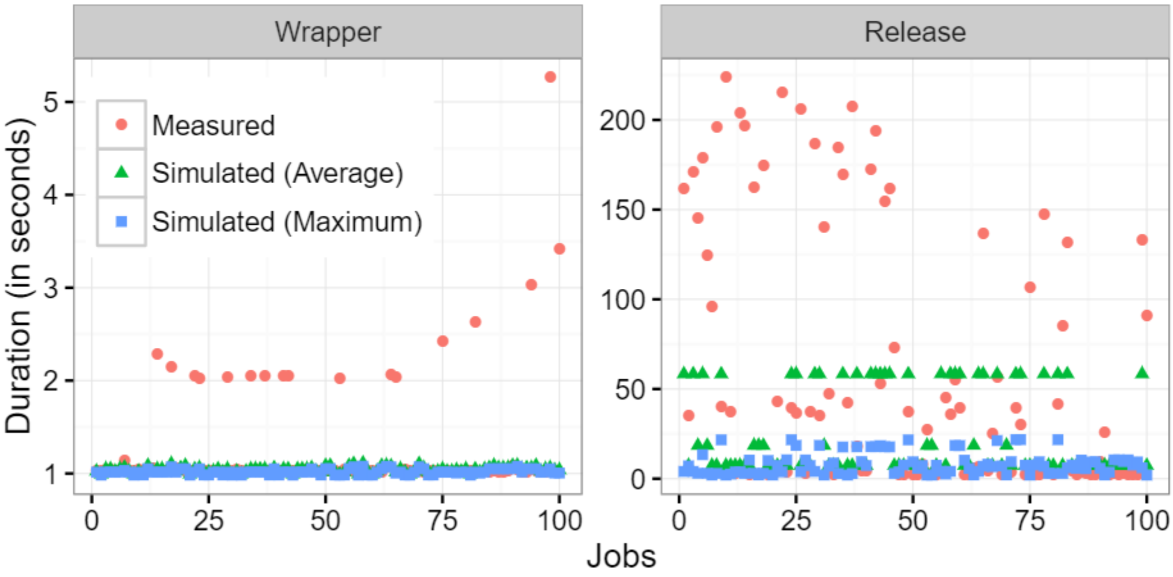

Figure 4: Measured[1] and simulated (with average or maximum bandwidth for each SE) transfer durations for two input files downloaded by each job in the fNUfzs workflow instance.

Using trace contents to instantiate the bandwidth values of the network links partially addresses the capture of the variability of file transfer duration, especially for the large release file. However, both the aggregation methods fail to be accurate. Averaging the observations tends to overestimate the transfer times (by a factor of 2 in average with a median of 1.5, and a maximum of 14.5), while using the maximum leads to an underestimation of at least a factor of 2 for half of the transfers. This denotes biases in our model that we propose to investigate and solve in the subsequent sections.

3.3 Distinguishing Transfer Type and File Size

We first looked at the distribution of the individual bandwidths derived from the file transfers that involve a given storage element in a given workflow trace. We notice great difference in some cases, sometimes of more than two orders of magnitude. We identified three root causes to this variability. First, some of the initial upload tests can suffer a delay. This delay, of usually one second, is negligible with regard to the total transfer time, but as these transfers involve a 12-byte file, this leads to unrealistically low bandwidth values, i.e., of less than a kilobit per second. Second, the limited precision of the timings logged in the traces leads to almost instantaneous transfer times for files of a few kilobytes. Then the derived bandwidth is unrealistically high, i.e., greater than 10 Gb/s. Third, we observed a sometimes large difference between the upload and download of the partial result files produced by the jobs. Each job uploads such a file to its local storage element which is then downloaded by the merge job from the same storage element. However, the concurrency conditions in which these transfers occur are different and impact the transfer times. The merge job downloads all the partial results in sequence while several

[1] For the sake of readibility, a data point corresponding to a transfer of the wrapper file that lasts for 65 seconds is not displayed.

worker nodes can upload their files to the same storage element simultaneously. Consequently, depending on the direction of the transfer, i.e., to or from a given SE, the derived bandwidth may greatly differ.

All these observations are likely to negatively impact the computation of the average or maximum bandwidth of the link connecting a storage element to the rest of the platform.

To address these issues, we propose to group the transfers by type (i.e., upload test, each of the three input files, and both upload and download of partial result files) and direction (i.e., to or from a storage element) before computing the average and maximum bandwidth values. Bandwidth estimations are then performed based on specific types for which the measured file transfer durations are assumed more reliable. For transfers from a storage element, we favor that of the release file, whose larger size prevents to be hit by the timing precision. Similarly, for transfers to a storage element, upload of partial results is preferred over the more sensitive initial upload test.

Figure 5 shows the impact of the average bandwidth computation method on the simulation of the transfer of the release file by jobs in the `fNUfzs` workflow instance. The transfer of this particular file is the one for which this improvement is the most noticeable, hence our focus on these results. The *Average (all)* data are the same as in Figure 4, while the *Average (grouped)* data refer to the distinction of transfer type and file size. In that case, the average bandwidth from a given storage element to any compute element is computed based only on the transfers of the release file found in the traces. We also distinguish the storage element from which the file has been downloaded in this figure.

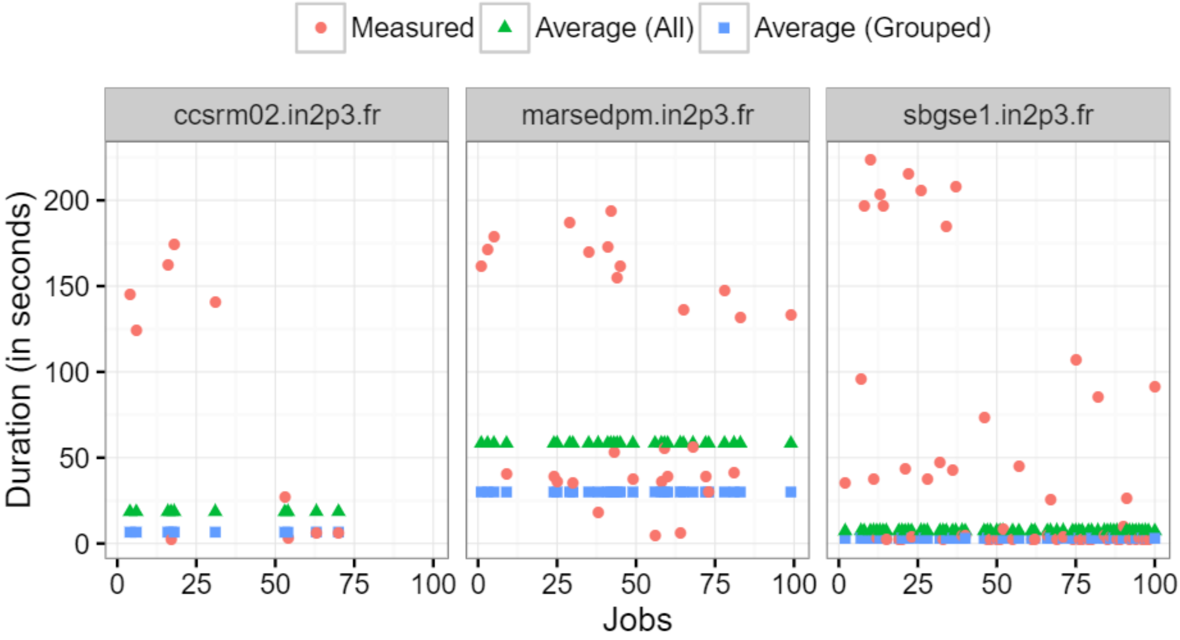

Figure 5: Measured and simulated (with global or grouped average bandwidth for each SE) durations for the download of the release file by each job in the `fNUfzs` workflow instance.

The proposed improvement reduces the overestimation of the transfer duration. Simulated transfers, whose durations were twice as long in average (and up to 14.5 times longer) than the actual transfers and are now 1.2 times shorter in average (with a maximum overestimation by a factor 7.5). However, we can clearly see in Figure 5 that, for a given storage element, the model still fails to capture the variability in transfer durations. To find the origin of this modeling bias and further improve our platform model, we zoom in on a specific storage element in the next section.

3.4 Distinguishing Computing Sites

When we isolate the transfers for a given storage element and input file size, we clearly and obviously observe (as in Figure 5) that using an average bandwidth leads to a uniform transfer time to all the computing sites. However, the bottom part of Figure 6 shows an important variability in the average bandwidth when we distinguish the downloads of the release file from the `marsedpm.in2p3.fr` storage element in the `fNUfzs` workflow instance by the *computing site* downloading this file.

The global average bandwidth for the SE, as computed in the previous section and depicted by a dashed line, is of 33.52 Mb/s. This value is a good approximation for the INFN-BARI and UKI-LT2-RHUL sites but an overestimation by a factor of around 5 for the INFN-PISA and UKI-LT2-IC-HEP sites and a clear underestimation for the UKI-LT2-QMUL site.

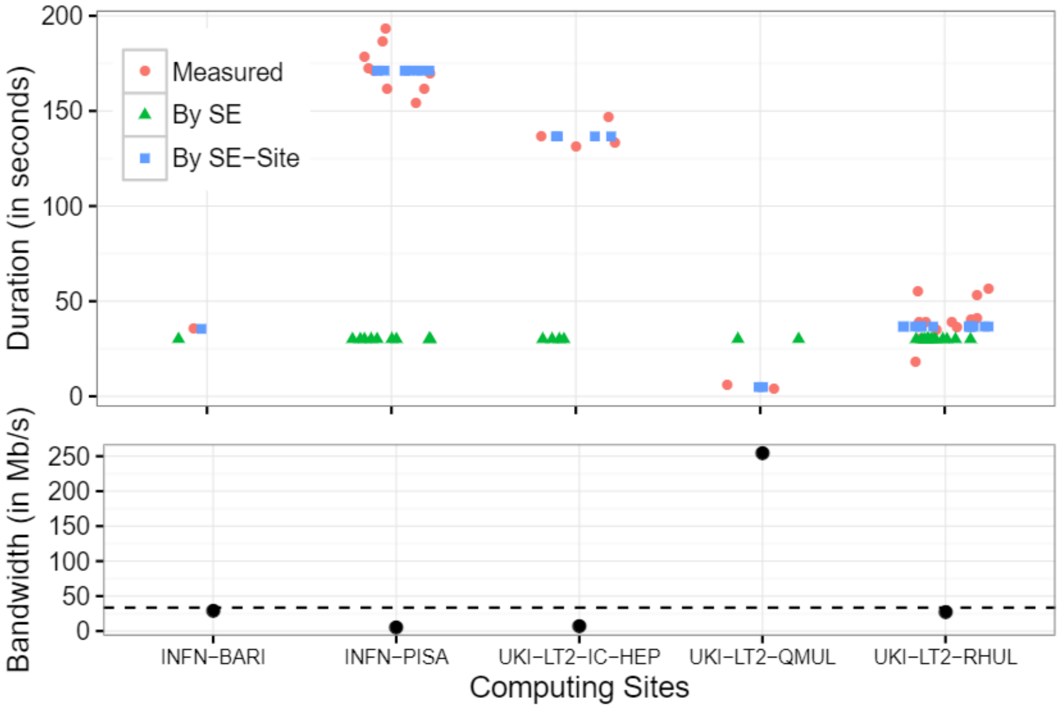

Figure 6: Measured and simulated durations for the downloads of the release file from the `marsedpm.in2p3.fr` SE in the `fNUfzs` workflow instance (top). Simulated times are obtained with a single average bandwidth (SE) and distinct average bandwidths (SE-Site) for each site (bottom).

This deviation to the global average directly explains the results presented in the top part of Figure 6. The "*By SE*" data corresponds to the middle panel of Figure 5 while the "*By SE-Site*" data relies on the distinct average value computed for each computing site. To improve the readability of the figure, we introduce a horizontal jitter to separate data points corresponding to similar durations.

Now we make a similar analysis for the alternate approach based on the determination of the maximum observed bandwidth. As for the average, we see in Figure 7 (bottom) an important deviation from the global maximum value of 321 Mb/s for most of the computing sites. We also see in Figure 7 (top) that the use of a single value for all the sites also leads to similar simulated durations for all the transfers.

In this version of the model, it should be noted that distinguishing the links between a storage element and the computing sites may bias the way the simulation kernel handles the sharing of the network resources. Indeed, separate links mean independent traffic and this can lead to a cumulative simulated bandwidth greater than what is observed from both the SE and computing site point of views. To prevent such a bias, we introduce the notion of *limiter* links whose bandwidths are determined by taking the maximum value observed over all the transfers to/from a SE or a given site (Figure 2 (B)).

Unlike the improvement observed for the average-based model, we see here that determining a distinct maximum bandwidth for each site dramatically degrades the quality of the simulation. This suggests that our estimation of the maximum value is biased and returns underestimations of the nominal bandwidths of the links between the storage element and the different computing sites.

More precisely, the modification of the model improves the simulation accuracy when there are only a few transfers from the SE to a computing site (i.e., for the INFN-BARI and UKI-LT2-QMUL sites).

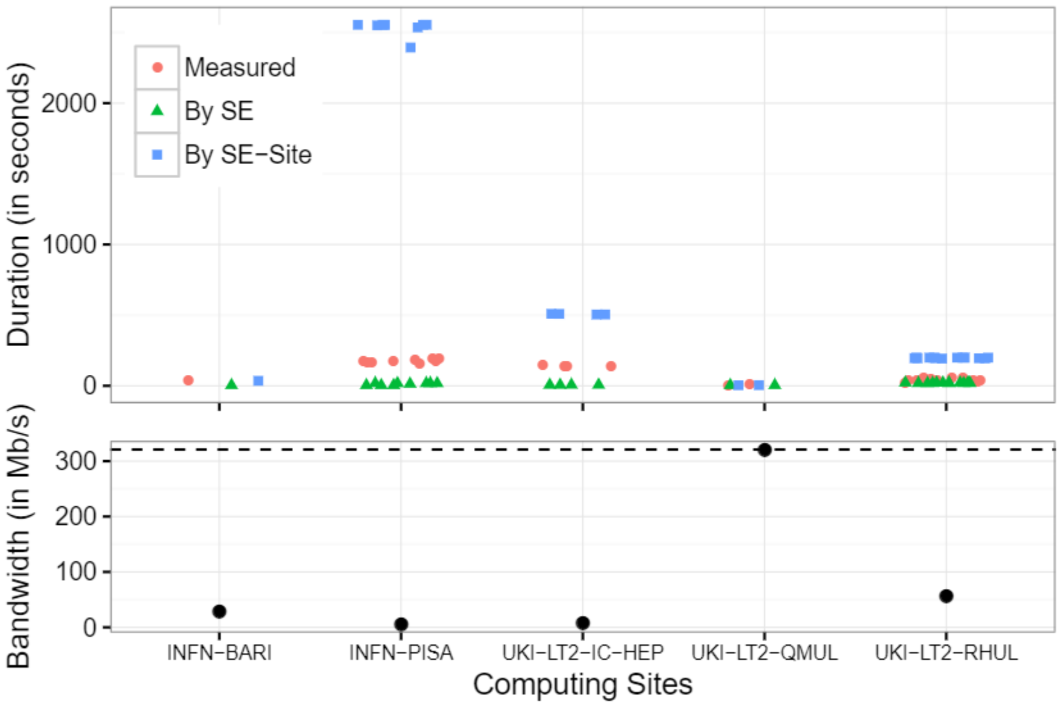

Figure 7: Measured and simulated durations for the downloads of the release file from the `marsedpm.in2p3.fr` SE in the `fNUfzs` workflow instance (top). Simulated times are obtained with a single maximum bandwidth (SE) and distinct maximum bandwidths (SE-Site) for each site (bottom).

When there are more transfers from the SE to a computing site, the durations become, sometimes largely, overestimated. We thus assume that our model fails to capture an important phenomenon, that we identify and take into account in the next section.

3.5 Correcting the Maximum Bandwidth

Figure 8 presents the part of the Gantt chart of the execution of the `fNUfzs` workflow instance, when the nine downloads of the release file by worker nodes in the INFN-PISA site from the `marsedpm.in2p3.fr` storage elements take place.

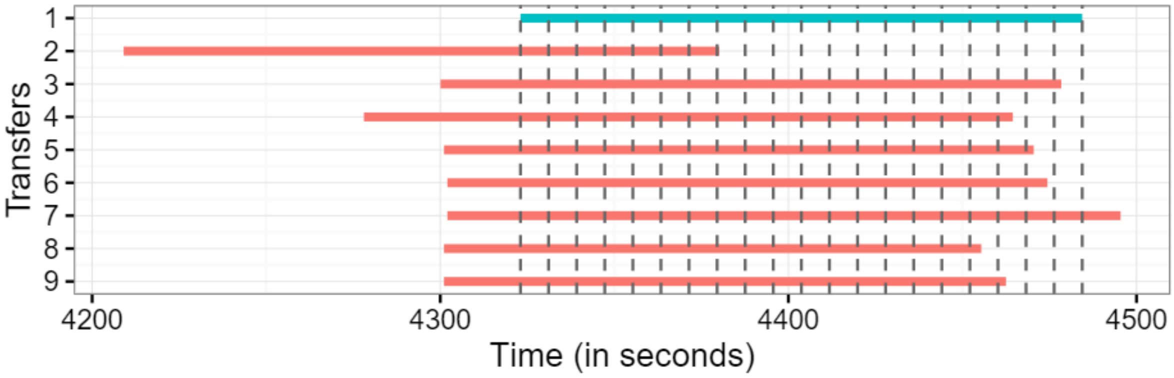

Figure 8: Gantt chart view of the transfers of the release file from the `marsedpm.in2p3.fr` SE to worker nodes in the INFN-PISA site.

We can see that almost all these transfers are simultaneous and of durations in the same range (from 154.4 to 193.4 seconds). As a consequence, the derived bandwidth of each individual transfer is impacted by the competition with the other transfers for a single shared network resource. In other words, the observed bandwidths correspond to the shares of the link capacity that are respectively allocated to each transfer, but they do not reflect the total capacity of the link itself.

To get a better estimate of the nominal capacity of the link, we thus have to first compute a correcting factor to the bandwidth derived from each transfer. The computation of this factor, during the generation of the platform model, somehow corresponds to the inverse of the computation made by the simulation kernel to assign a share of the network resource to a given transfer. We proceed as follows. First we sample the duration of the transfer in at most 50 uniform intervals. For instance, in Figure 8, we split the first transfer (in blue) in 20 intervals. For each of them, we estimate the number of concurrent transfers, hence how the network resource is shared. The first transfer will thus have only one ninth of the capacity of the link for the seven first interval but one third of it during the last interval. Then we compute the correcting factor as:

$$f = \frac{n}{\sum_{i=1}^{n} \frac{1}{ci}}, \tag{1}$$

where n is the number of intervals, and c_i the number of concurrent transfers in the i^{th} interval. In our example, we obtain a correcting factor of $20/(7/8{+}10/7{+}1/6{+}1/4{+}1/2) = 6.21$ for the first transfer.

For each transfer, we compute three different correcting factors by estimating the concurrency for each SE, each site and each (SE, Site) couple. This allows us to correct the instantiation of the link between a SE and a site and of the limiter links.

Figure 9 shows the impact of this correction of the maximum observed bandwidth on the results presented in Figure 7. We can see that the important overestimation of the transfer durations disappears and the simulation accuracy is globally improved. However, this model now leads to a moderate underestimation of the durations for two sites (INFN-PISA and UKI-LT2-RHUL), which is caused by different phenomena.

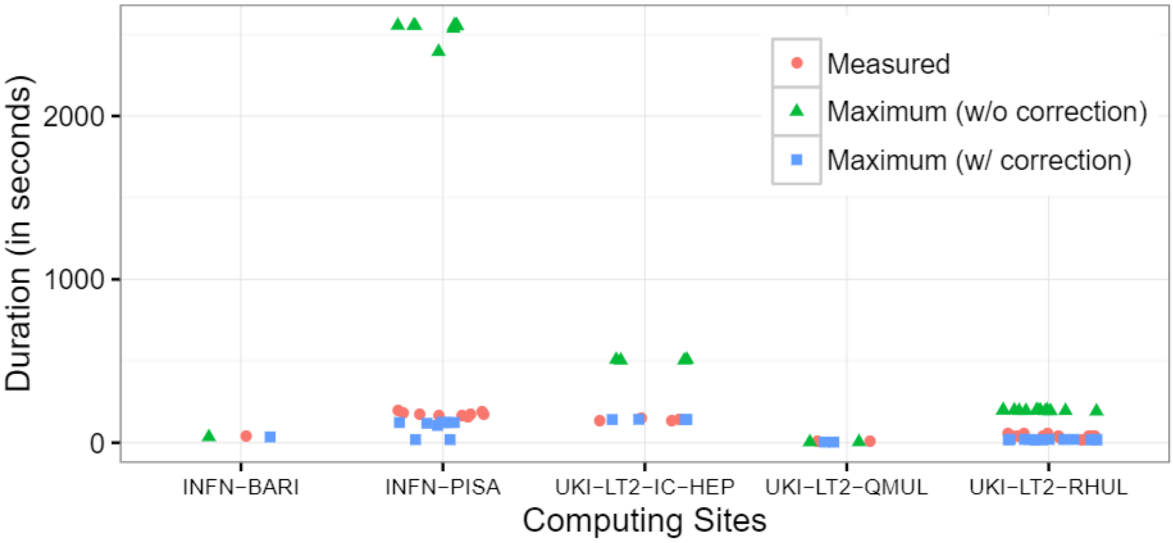

Figure 9: Measured and simulated durations for the downloads of the release file from the `marsedpm.in2p3.fr` SE in the `fNUfzs` workflow instance with and without correction of the maximum observed bandwidth.

For the INFN-PISA site, two transfers (second and fourth from top in Figure 8) start before a bulk of seven transfers. While these two transfers last for roughly the same time as the others in the real execution, they will experience less concurrency in simulation and will then be faster to complete (in around 20 seconds). More importantly they will end before the other transfers start, which in turn impacts the concurrency they experience and their simulated duration. We suspect the influence of some external load during the first two transfers that our model currently fails to capture.

For the UKI-LT2-RHUL site, we observed that the transfer durations differ depending on which cluster in the site is involved. We propose to illustrate and address this phenomenon in the next section using a more glaring example.

3.6 Distinguishing Clusters in Sites

In our analysis of the execution traces, we sometimes observed an important variability in transfer durations for a given (SE, Site) couple. This is particularly striking for downloads of the release file from the `sbgse1.in2p3.fr` SE to the INFN-PISA site in the `LQz3XJ` workflow instance. In this particular case, the durations are in a [4.04s; 257.40s] interval. This variability is related to the cluster where the nodes downloading the file are located, as shown in Figure 10. Using a single bandwidth value for the site, be it the average or the maximum, thus fails to capture that cluster-related difference in performance, and leads to inaccurate simulations.

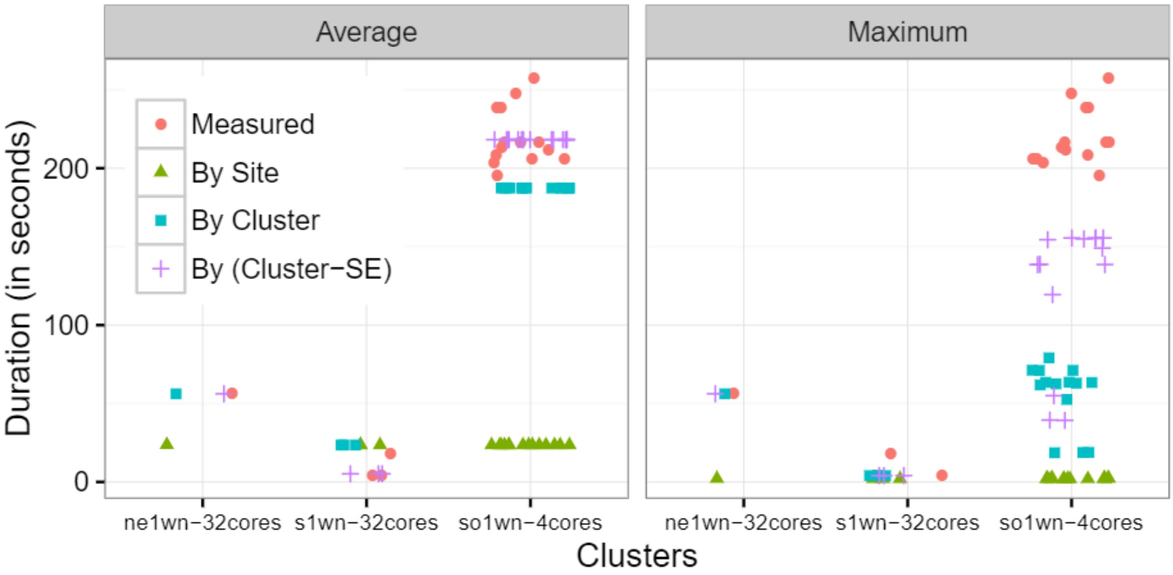

Figure 10: Measured and simulated durations for the downloads of the release file from the `sbgse1.in2p3.fr` SE to the INFN-PISA site in the `LQz3XJ` workflow instance with and without distinction of the clusters.

To improve our models, we propose to differentiate the links that connect the different clusters in a site to the rest of the platform (see Figure 2 (C)). Note that the clustering of worker nodes is based only on information available in the trace, i.e., name, number of cores, and processing speed, and do not leverage information by the sites themselves. As for the previous improvements, we modify the way we aggregate the individual bandwidth values to take this distinction into account. We also compute specific correction factors for the model based on the maximum bandwidth.

Unfortunately, this modification improves the accuracy of the simulations only partially. Figure 10 highlights two limitations with different causes but a common solution. For the average-based model, we observe an overestimation of the durations for the `s1wn-32cores` cluster. This is because the link between the SE and the site becomes a bottleneck. Indeed, the average bandwidth we compute for this link is hindered by the bad performance obtained for the `so1wn-4cores` cluster. Conversely, in the maximum-based model, we see an underestimation of the durations for the `so1wn-4cores` cluster. In this case it is the better bandwidth from another SE (not displayed in Figure 10) that defines the maximum observed bandwidth for this cluster. These two situations advocate for a common improvement that consists in differentiating the links not between a site and a SE, but between each individual cluster and a SE.

The results obtained with this final improvement are labeled as "*By Cluster-SE*" in Figure 10. We can see that, as expected, it completely solves the inaccuracy issue for the average-based model. For the maximum-based model, the proposed distinction improves the accuracy only partially. This can be explained, as in the previous section, by some abnormally long transfers in the real execution that have a shorter simulated duration which in turn modifies the level of concurrency on the network resource. We can see three such transfers in Figure 10.

4 Experimental Evaluation

In this section we evaluate our improved platform models by confronting simulation results obtained using these models to other simulation results using the state-of-the-art model presented in Section 3.1 and to the reference given by the real transfer durations extracted from traces. Then we provide an analysis of the causes of the main simulation inacurracies.

4.1 Analysis of Simulated Transfer Durations

The evaluation uses transfer durations extracted from the execution traces described in Section 2.2, as well as simulated transfer durations obtained with the following platform models: *10G-SotA* (as described in Section 3.1), *Average* and *Maximum* (as described in Section 3.6). For each model, we simulate 5,316 download transfers. Among these, 1,772 files (one third) correspond to the GATE release file and are larger than 121 MB. The other files correspond to the GATE inputs and wrappers and are smaller than 130 kB. Figure 11 shows a summary of the measured and simulated transfer durations for these two sets of files.

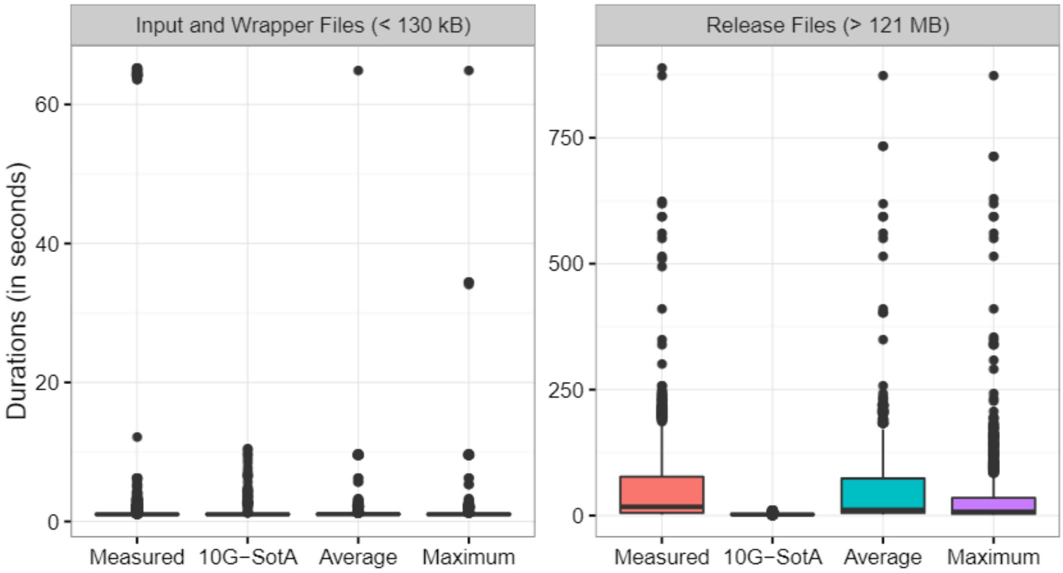

Figure 11: Graphical summary of measured and simulated (with the 10G-SotA, Average, and Maximum models) file transfer durations (in seconds).

For the GATE inputs and wrappers files, 80% of these 3,544 small file transfers last for less than 1.3 seconds in the real execution. The minimal incompressible delay imposed by control and network latency being of one second, there is no possible discrimination between the models and very few misestimations. We also observe 26 transfers (of the 73 kB wrapper file) whose durations are close to 64.5 seconds. The only common point to all these transfers is the storage element use to download this wrapper file. This highlights an issue with this specific SE rather than a modeling problem. However, it is interesting to note that both the *Average* and *Maximum* models are able to correctly reproduce one of this longer transfers. The proposed models are also able to partially capture the variability of 15% to 19% of the remaining transfers, while the *10G-SotA* model cannot. In the remaining of this section, we focus our analysis on the transfers of the larger GATE release file. The longer durations allow us to get a better insight on the respective strengths and limitations of the different models.

Table 1: Statistics of measured and simulated (with the 10G-SotA, Average, and Maximum models) durations (in seconds) of the transfer of the GATE release files (> 121 MB).

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Measured	2.00	5.01	17.42	49.91	77.31	888.80
10G-SotA	1.19	2.02	2.03	2.55	2.15	11.52
Average	2.01	5.21	11.44	45.17	73.62	873.80
Maximum	1.98	3.21	7.44	31.72	35.09	873.80

We complete Figure 11 with Table 1 that gives the corresponding statistics for the transfers of the release file. The poor quality of the *10G-SotA* model is blatant: it largely underestimates all the transfer durations and cannot capture the variability that characterizes the real executions. The proposed *Average* and *Maximum* models, however, are able to reproduce that variability and correctly simulate the longest transfers. We also note a tendency of the *Maximum* platform model to underestimate the transfer durations, which can be explained by the fact that this models relies on estimations of the nominal bandwidths of the links that are less accurate than the average of the bandwidths as perceived by the application.

4.2 Analysis of Errors

Another way to assess the respective quality of the different platform models and to measure the benefits of the proposed trace-based approach in terms of simulation accuracy is to compare errors with respect to the real transfer times. The relative difference between simulation results and the reference of a real execution can be computed in many ways. For instance, the relative error is a standard error measure which gives a good indication on the precision of a simulation and whether it under- or overestimates the reality. However, this method has a certain number of disadvantages such as the fact that it is not symmetrical (the error lies in the interval]-∞, 1]) or the fact that the relative difference of bandwidths is different of the relative difference of transfer times, while there is a priori no reason to favor one over the other. As explained in [20], an absolute logarithmic error solves these two issues and allows us to compare the different models more easily. We define the absolute logarithmic error as follows:

$$LogErr = |\log(R) - log(S)|, \qquad (2)$$

where R is the real time and S the simulated time.

The absolute logarithmic error also allow us to apply additive aggregation operators such as the maximum or the mean, hence easing the comparison of the different models.

Figure 12 shows the Cumulative Distribution Functions of the absolute logarithmic errors obtained for each model. The maximum errors respectively are 6.09 for the *10G-SotA* model, 3.99 for the *Maximum* model, and 2.83 for the *Average* model.

This figure confirms the poor accuracy of the *10G-SotA* platform model that shows errors greater than two (i.e., relative error greater than 639%) for nearly half of the transfers. This corresponds to the systematic and large underestimations of the file transfer durations made by this model that was illustrated by Figure 11.

As expected, the *Average* platform model leads to best results as the sharing of network resources by concurrent transfers is directly captured in the model which reflects the network connectivity as experienced by the application. This model is thus able to simulate 75% of the transfers with a logarithmic error smaller than 0.35 (relative error: 42%) and 92.9% of the transfers with an error smaller than 1. Finally, the mean logarithmic error is 0.28 (relative error: 32%).

The *Maximum* platform model also clearly outperforms the *10G-SotA* model but leads to greater errors than the *Average* model. The mean logarithmic error is 0.62 (relative error: 85%) and 75% of the transfers are simulated with a logarithmic error under 0.97 (relative error: 164%). This loss of accuracy is the cost of the higher potential for further studies of this model that offers a better reusability beyond the simulated replay of the execution that led to its generation. It also confirms the tendency of this

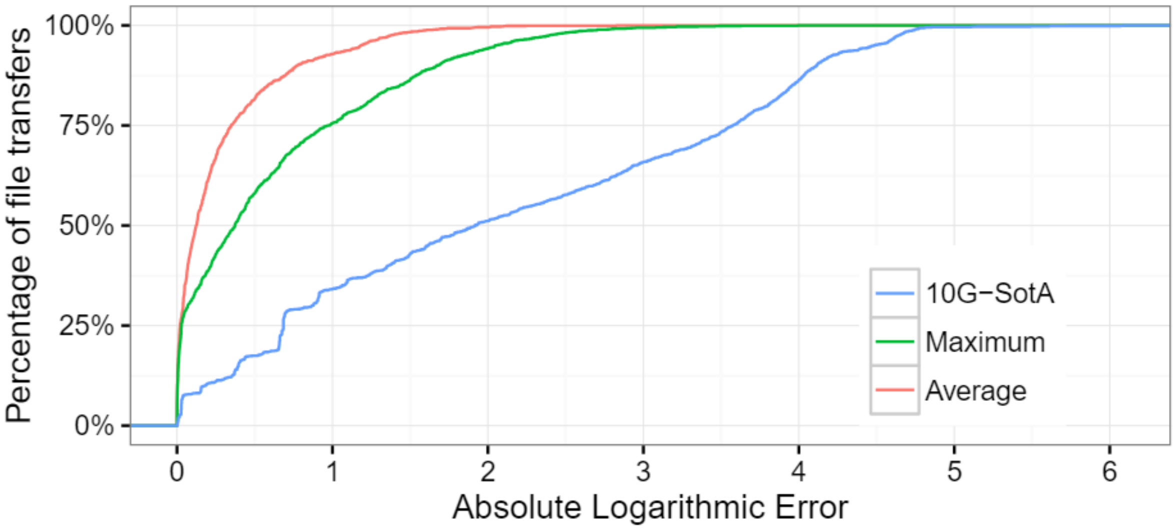

Figure 12: Cumulative Distributed Functions of the absolute logarithmic error achieved by the three platform models over the whole set of transfers of the release files.

model to be too optimistic in its determination of the bandwidth values, and thus to underestimates the transfer durations. A deeper analysis of the individual simulation results would be needed to determine whether the way to compute the bandwidth correction factor can be improved.

To summarize, our analysis of the distribution of the simulated transfer durations and of the logarithmic errors demonstrates that a correct modeling of the interconnection topology and a good instantiation of the characteristics of the network resources are key to the quality of simulations of file transfers on a large-scale distributed infrastructure. It also shows that leveraging the contents of execution traces is a sound approach that enables a pretty accurate simulated replay of a given execution with the *Average* model or even beyond with the *Maximum* model, with an affordable accuracy loss.

4.3 Analysis of the Root Causes of Main Simulation Errors

While the obtained results are promising and greatly outperform the state-of-the-art model, they also include some prohibitive errors. In this section, we analyze such errors to determine whether they correspond to a modeling flaw or to phenomena that we cannot, or do not even want to, model. For instance, the cause behind the outlying durations for small file transfers does not have to be part of the platform model, but rather to be injected into the simulation as an additional parameter. Then, and before presenting two phenomena that we identified as sources of high inaccuracy, we recall that Table 1 shows that all the simulated durations lie in the same range of values as the actual durations.

The first identified source of high inaccuracy is illustrated by Figure 13 that shows a Gantt chart view of five transfers of the release file from the same SE to worker nodes in the same cluster. We see that four files (1 to 4) are transferred much faster (from 14s to 45s) than a fifth one (in 172s).

In the *Maximum* model, the bandwidth derives from transfer 4 and is, after correction, of 968 Mb/s. However, the actual bandwidth for transfer 5 is only of 23.5 Mb/s. This difference leads to a dramatic underestimation of the simulated duration (only 5.15s) and one of the largest errors. With the *Average* model, this also impacts the simulation accuracy by lowering the computed average (at 163 Mb/s). Again the great deviation from the actual bandwidth leads to an important error.

From the point of view of our workflow execution, transfer 5 is neither impacted by another transfer to other nodes, not even from another SE, nor by transfers done by other jobs executed on the same node. The problem thus does not come from a bad handling of network contention by our model. This particular transfer has to be impacted by some external transient load that we do not model.

The second main identified source of large errors pertains to transfers from the declared local SE. Such transfers can be very fast (only 2s to transfer 121 MB) but are also subject to an important variability that follows a certain pattern. We observed, for several workflow instances, durations that increase

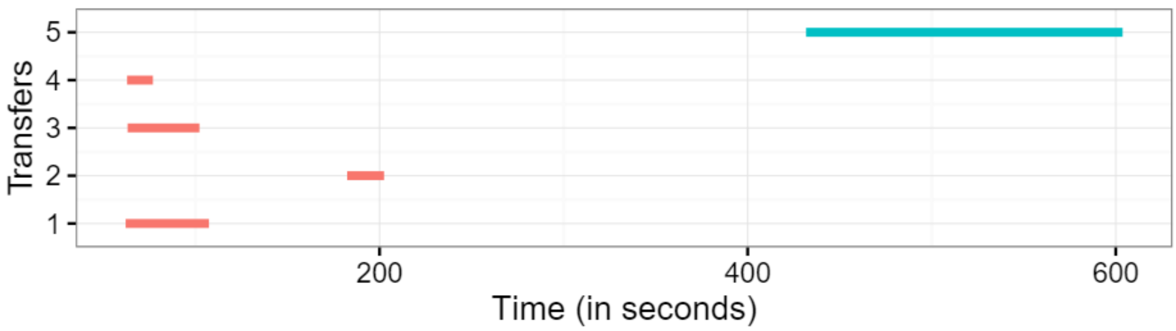

Figure 13: Gantt chart view of the transfers of the release file from the `ccsrm02.in2p3.fr` SE to worker nodes in the AUVERGRID site in the `2RcwCY` workflow instance.

exponentially. We suspect the influence of some local configuration of the storage elements that limits the number of concurrent transfers and might trigger some timeout-retry mechanisms. Unfortunately, the traces do not make the distinction between the time spent waiting for the SE to be available from the time actually spent to transfer the file over the network. Our models are thus unable to capture this effect.

5 Related Work

The simulation of large-scale distributed computing infrastructures has received an enormous amount of attention in the literature. In the area of grid and cloud computing, many simulators have been developed and used by researchers over the last decade. Some were made available to the community but proved to be short-lived perhaps because they were too project-specific. For instance, ChicSim [16] and OptorSim [2], designed to study data replication in grids, have been discontinued. SimGrid [9] and GridSim [5] are two simulation toolkits widely used in grid computing research. More recently, simulators have been proposed for simulating cloud computing platforms and applications. GroudSim [15] is a framework that enables the simulation of both grid and cloud systems. CloudSim [6] builds on the same simulation internals as GridSim but exposes specific cloud interfaces.

Platform models are mandatory to the simulation of distributed computing infrastructures. However, large production infrastructures such as EGI are difficult to model. Then, studies such as [7,13] come to either use existing descriptions of controlled environments, such as that of the Grid'5000 [4] experimental testbed or more simplistic descriptions instantiated with homogeneous bandwidths (e.g., 10 Gb/s).

Trace analysis is a common approach to model large scale distributed platforms and predict the performance of individual operations, e.g., file transfers. In [1], authors propose a hybrid simulation model for data grids based on the statistical analysis of traces. In [11], authors present a practical machine learning method to predict job characteristics by conducting an analysis of the workload traces.

Simulation results should ideally be close to results that would be obtained in a production system. The accuracy of a simulator can be evaluated by confronting simulation results to the ground truth of actual experiments. However, published evaluations are often weak and/or incomplete. Some evaluations are merely qualitative. When quantitative, evaluations often consist in exhibiting a few good cases in which simulation results lead to reasonable trends (few real executions are actually attempted). Recent work [8] emphasizes the need for realistic simulators and the use of multiple approaches (mathematical models, discrete-event simulation, and real production experiments) for cross-validation, allowing for a good level of confidence in the obtained results.

6 Conclusion and Future Work

Simulation is a powerful tool that allows for easier prototyping and testing of new methods, enabling at the same time their thorough evaluation through deterministic and reproducible experiments. However, simulators are rarely tuned to faithfully model large-scale distributed computing platforms and simulation results are seldom compared to the ground truth of actual production runs of real applications.

In this paper, we proposed a methodology for building realistic platform models that allow us to accurately simulate file transfers across a distributed infrastructure. The models were built and instantiated from execution traces obtained from a production environment. Last but not least, we evaluated the proposed models by confronting our simulation results to (i) the ground-truth of the real traces and (ii) simulation results using a more homogeneous state-of-the-art platform model.

Results show that the proposed platform models are able to correctly reproduce real-life variability, as opposed to the state-of-the-art platform model. Indeed, both the *Maximum* and *Average* models manage to correctly capture the distribution of the transfer durations. The analysis of the absolute logarithmic errors also shows that the proposed models clearly outperform the *10G-SotA* model, which largely underestimates a vast majority of the transfer durations.

Based on the the obtained results, we conclude that realistic platform models are essential for the accuracy of simulations aiming at reproducing file transfers across a distributed infrastructure. They largely outperform the more simplistic, homogeneous existing models, thus fully accounting for the efforts needed to build such realistic models.

As future work, we plan to build on these very promising results and address the different limitations highlighted by our analysis of the root causes of the main simulation errors. It would also be interesting to confirm our findings on a larger set of execution traces, and to conduct a deeper invalidation study. The spatial and temporal aggregations of different execution traces is a challenging task that would allow us move away from a specific trace and generate simulated platforms that mimic the characteristics of a production system.

We also aim at extending the simulation capacities of the VIPSimulator that underlies this study, not only to handle more complex scenarios that includes failed jobs, but also to simulate the other components of the workflow execution.

Acknowledgments

This work is partially supported by the LABEX PRIMES (ANR-11-LABX-0063) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR) and the SONGS ANR project (11-ANR-INFR-13).

References

[1] Martin Barisits, Eva Kühn, and Mario Lassnig. A Hybrid Simulation Model for Data Grids. In *Proc. of 16th IEEE/ACM Intl. Symp. on Cluster, Cloud and Grid Computing*, pages 255–260, 2016.

[2] William H. Bell, David G. Cameron, A. Paul Millar, Luigi Capozza, Kurt Stockinger, and Floriano Zini. OptorSim - A Grid Simulator for Studying Dynamic Data Replication Strategies. *IJHPCA*, 17(4):403–416, 2003.

[3] Laurent Bobelin, Arnaud Legrand, David Alejandro González Márquez, Pierre Navarro, Martin Quinson, Frédéric Suter, and Christophe Thiery. Scalable Multi-Purpose Network Representation for Large Scale Distributed System Simulation. In *Proc. of the 12th IEEE/ACM Intl. Symp. on Cluster, Cloud and Grid Computing*, pages 220–227, 2012.

[4] Raphaël Bolze et al. Grid'5000: A Large Scale And Highly Reconfigurable Experimental Grid Testbed. *IJHPCA*, 20(4):481–494, 2006.

[5] Rajkumar Buyya and Manzur Murshed. Gridsim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *CCPE*, 14(13-15):1175–1220, 2002.

[6] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *SPE*, 41(1):23–50, 2011.

[7] Sorina Camarasu-Pop, Tristan Glatard, and Hugues Benoit-Cattin. Simulating Application Workflows and Services Deployed on the European Grid Infrastructure. In *Proc. of the 13th IEEE/ACM Intl. Symp. on Cluster, Cloud, and Grid Computing*, pages 18–25, Delft, Netherlands, May 2013.

[8] Sorina Camarasu-Pop, Tristan Glatard, and Hugues Benoit-Cattin. Combining Analytical Modeling, Realistic Simulation and Real Experimentation for the Optimization of Monte-Carlo Applications on the European Grid Infrastructure. *FGCS*, 57:13–23, 2016.

[9] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. *JPDC*, 74(10):2899–2917, 2014.

[10] Anchen Chai, Mohammad Bazm, Sorina Camarasu-Pop, Tristan Glatard, Hugues Benoit-Cattin, and Frédéric Suter. Companion of the article on Modeling Distributed Platforms from Application Traces for Realistic File Transfer Simulation, 2016. Available at: https://dx.doi.org/10.6084/m9.figshare.4253426.

[11] Rafael Ferreira da Silva, Mats Rynge, Gideon Juve, Igor Sfiligoi, Ewa Deelman, James Letts, Frank Würthwein, and Miron Livny. Characterizing a High Throughput Computing Workload: The Compact Muon Solenoid (CMS) Experiment at LHC. *Procedia Computer Science*, 51:39–48, 2015.

[12] Tristan Glatard et al. A Virtual Imaging Platform for Multi-Modality Medical Image Simulation. *IEEE Transactions on Medical Imaging*, 32(1):110–118, 2013.

[13] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. Preprint available at: http://arxiv.org/abs/1606.02007, 2016.

[14] Sébastien Jan et al. GATE: a Simulation Toolkit for PET and SPECT. *Physics in Medicine and Biology*, 49(19):4543, 2004.

[15] Simon Ostermann, Radu Prodan, and Thomas Fahringer. Dynamic Cloud Provisioning for Scientific Grid Workflows. In *Proc. of the 11th ACM/IEEE International Conference on Grid Computing*, pages 97–104, 2010.

[16] Kavitha Ranganathan and Ian Foster. Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. In *Proc. of the 11th IEEE Intl. Symp. on High Performance Distributed Computing*, pages 352–358, 2002.

[17] Frédéric Suter, Anchen Chai, and Mohammad Bazm. The Log2sim Framework. Available at: http://github.com/frs69wq/log2sim, 2016.

[18] Frédéric Suter, Anchen Chai, and Sorina Camarasu-Pop. VIPSimulator: a Simulator of Gate Workflow Execution. Available at: http://github.com/frs69wq/VIPSimulator, 2016.

[19] Yehuda Vardi. Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data. *Journal of the American statistical association*, 91(433):365–377, 1996.

[20] Pedro Velho and Arnaud Legrand. Accuracy Study and Improvement of Network Simulation in the SimGrid Framework. In *Proc. of the 2nd Intl. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems*, 2009.

[21] Pedro Velho, Lucas Mello Schnorr, Henri Casanova, and Arnaud Legrand. On the Validity of Flow-Level TCP Network Models for Grid and Cloud Simulations. *ACM TOMACS*, 23(4):23, 2013.