International Journal of Foundations of Computer Science

NETWORK MODELING ISSUES FOR GRID APPLICATION SCHEDULING

HENRI CASANOVA
San Diego Supercomputer Center, Dept. of Computer Science and Engineering,
University of California, San Diego,
9500 Gilman Dr., La Jolla, CA 92093-0114, U.S.A.

ABSTRACT

The dominant trend in scientific computing today is the establishment of platforms that span multiple institutions to support applications at unprecedented scales. On most distributed computing platforms a requirement to achieve high performance is the careful scheduling of distributed application components onto the available resources. While scheduling has been an active area of research for many decades most of the platform models traditionally used in scheduling research, and in particular network models, break down for platforms spanning wide-area networks. In this paper we examine network modeling issues for large-scale platforms from the perspective of scheduling. The main challenge we address is the development of models that are sophisticated enough to be more realistic than those traditionally used in the field, but simple enough that they are still amenable to analysis. In particular, we discuss issues of bandwidth sharing and topology modeling. Also, while these models can be used to define and reason about realistic scheduling problems, we show that they also provide a good basis for fast simulation, which is the typical method to evaluate scheduling algorithms, as demonstrated in our implementation of the SimGrid simulation framework.

Keywords: Grid Computing, Network Modeling, Simulation, Scheduling

1. Introduction

Parallel and distributed application scheduling, i.e., the decision process by which components of an application are assigned to resources that are distributed over a network, is key to achieving high performance and has thus been an exceptionally active research area for several decades. A scheduling problem is generally defined by three elements: (i) an *application model* that specifies the target application's structure, typically in terms of computational tasks and data to be communicated between tasks; (ii) a *platform model* that specifies the nature of the available resources and of the network by which they are interconnected; and (iii) an *objective* that must be achieved, such as minimizing application execution time, minimizing the monetary cost of execution, maximizing execution time predictability, etc. Note that this definition of a scheduling problem is for *application scheduling* in which one wishes to optimize the execution of a single application. The models we de-

This material is based upon work supported by the National Science Foundation under Grant ACI-0204007.

scribe can of course be utilized for studying *job scheduling* problems in which one considers several individual applications (typically belonging to different users) and one wishes to optimize for some aggregate notion of performance across all these applications (e.g., average performance, fairness). In this case there can be multiple application models and a "workload model" that describes the mix of applications.

The dominant trend in high performance computing today is the establishment of computing platforms that span large networks. The goal is often to support applications at scales that are beyond what can be achieved at a single site or institution. This *Grid Computing* [20] approach has been made possible through the development of appropriate middleware services [22, 21], and a number of Grid platforms have been put in production [56, 44, 16]. The key question that we attempt to answer in this paper is that of deciding which platform models should be instantiated for the purpose of scheduling research on large-scale platforms, both from the perspective of analysis and simulation. The challenge is to instantiate models that are simple enough that they can be used for analytical purposes and for fast simulations, but sophisticated enough that they capture the relevant characteristics of real-world platforms.

One key difference between Grid platforms and more traditional parallel computing platforms such as clusters is the nature of the network. The properties of wide-area networks are radically different from those of, say, a high-performance switch within a cluster, and must be well understood to instantiate realistic platform models. Platform models are one of the three defining elements of scheduling problems. Consequently, realistic platform modeling is necessary to ensure that defined scheduling problems and proposed scheduling algorithms are relevant to practice. There has traditionally been an intellectual disconnect between the wide-area networking community and the scheduling community: The former views wide-area networks as used for content delivery, and focuses on issues like the design of new network protocols; the latter views the network as a communication medium for coordinated computing, and as such is concerned with the performance of a single application. As a result, researchers in the area of scheduling tend to use simplistic network models. Some of these models, while perhaps appropriate for traditional parallel computing platforms, become radically inadequate when used to model wide-area networks. In this paper we advocate the use of more sophisticated models that capture some of the specific properties of these networks.

The models discussed in this paper can be used directly as a basis for developing and analyzing new scheduling algorithms. However, it is often impossible to obtain analytical results concerning the efficacy of different scheduling algorithms, and one has to rely on empirical evaluations and comparisons. One possible approach is to execute applications on real-world testbeds, but that is often impossible. First, it is difficult to perform reproducible and sufficiently diverse experiments on wide-area computing platforms. Second, and perhaps most importantly, the evaluation of scheduling algorithms often requires large numbers of experiments over large spaces of parameters defining the application and the platform. Consequently, fast simulations have long been the method of choice for evaluating the relative merit of

competing scheduling strategies. In the face of new emerging computing platforms such as Grids, there is a need for a simulation framework targeted to scheduling research that strikes a sound balance between realistic and fast simulations. In this view, we have implemented our models as part of the SimGrid simulation toolkit [52, 33].

This paper is organized as follows. In Sections 2, 3, and 4 we discuss three aspects of networks that must be taken into account for instantiating realistic platform models that are relevant to Grid computing practice: (i) network latency; (ii) bandwidth sharing; and (iii) network topology. For each we describe the issue, briefly review the typical modeling approach used in traditional scheduling work, and propose new models when necessary. In Section 5 we describe the SimGrid simulation toolkit, which implements most of the models discussed in the three previous sections. Finally, Section 6 concludes the paper with a brief summary of our results and a discussion of future directions.

2. Network Latency

It is well known that a reasonable approximation of the time required to send x bytes of data over a network link is affine of the form $\alpha + x/\beta$, where α is the *latency* (i.e., the time required for a zero-byte message to travel from the source to the destination), and β is the data transfer rate (in fact, efforts such as LogP [14] proposed and validated more sophisticated models as early as ten years ago). Nevertheless, many scheduling works have assumed linear transfer times x/β, for instance in the divisible load scheduling area [11, 6, 35]. Indeed, assuming a linear transfer time makes it possible to obtain elegant solutions to certain scheduling problems. However, ignoring latencies may lead to flawed solutions in certain cases as there is no prohibitive cost to sending large numbers of very small messages. This is seen for instance in [6], which develops a multi-round divisible load scheduling algorithm. In short, this algorithm is designed for a master-worker paradigm in which the master sends out carefully chosen amounts of work to workers in a sequence of rounds in a view to pipelining communication and computation. The principle of the solution is fundamentally interesting, but the authors themselves note that the linear model implies an infinite number of rounds with an infinitesimal amount of work being sent out to each worker at each round. This is clearly impractical and the authors point out that a reasonable number of rounds should be selected, or, alternatively, a reasonable minimum message size could be imposed. While this is essentially tenable, without a model that takes latencies explicitly into account it is sometimes difficult to quantify what "reasonable" may be. Typically, as seen for instance in [59] in the context of multi-round divisible load scheduling algorithms, taking latencies into account adds a significant amount of complexity to the scheduling problem and to its solution.

One may wonder whether the fixed part, α, of the transfer time is actually significant when compared to the proportional part, x/β. While this of course depends on the message size x, the current trend indicates that while latencies are bounded below by the speed of light, network bandwidth increases at an exponential rate.

For instance, the TeraGrid platform [56] has established a 40GBit/sec dedicated link between the San Diego Supercomputer Center (SDSC) and the National Center for Supercomputing Applications (NCSA). The expected network latency is in the 100ms range. A back of-the-envelope calculation assuming that a data transfer could use the full bandwidth (e.g., with parallel TCP connections and tuned congestion windows) indicates that one third of the time to transfer 1 GByte of data is due to the network latency on the TeraGrid. This is only a coarse estimate, but it is indicative of what is to be expected for these types of networks in the future. Similar trends are seen in current efforts to deploy long-haul optical networks with the possibility of using parallel lambdas (i.e., multiple optical wavelengths to provide independent communication channels along a strand of fiber optic cable) for transferring large amounts of data, such as in the OptiPuter project [53]. Therefore, the latency component of the communication time can be significant when compared to the bandwidth component even when relatively large data sets are transfered on Grid platforms, and this trend is likely to continue. While some applications may transfer amounts of data so large that the latency component is indeed negligible, for other application it is, or soon will be, imperative that scheduling strategies take network latencies into account.

It is interesting to note that latencies can also be experienced when using computational resources. Indeed, Grid applications require that (many) processes be initiated on remote resources, which generally involves calls to middleware services to perform authentication, resource acquisition, process creation, etc. This overhead can be significant. For instance, data obtained as part of the GRASP benchmarking project [13] shows that using the Globus Toolkit version 2.0 [24] to launch a no-op task on a remote compute resource can require up to 25 seconds in practice. Similar trends are observable with current implementations of the Open Grid Service Architecture (OGSA), as starting a task may require a sequence of Web service instantiations. These observations and trends show that computation latencies must be modeled when developing application scheduling strategies. This has recently been done by a few researchers (see for example the work in [7, 48, 1, 60]), who have sown that in some cases adding computation latencies lead to new difficulties as well as to new, non-trivial results.

3. Bandwidth Sharing

Solving a scheduling problem can be viewed as deciding how to share available resources, both in time and space, among components of an application or among different applications. This applies to network resources, whose adequate sharing is key for achieving high performance on distributed platforms. The most traditional assumption is that processors are interconnected with each other in a point-to-point fashion via network links. In the "one-port" model a processor can only send data over one network link at a time, while in the "multi-port" model a processor can send data over multiple links simultaneously. Therefore, there is no notion of bandwidth sharing among communications that involve distinct pairs of processors. In addition, it is generally assumed that a single data transfer may occur on a given

network link at a time, with no bandwidth sharing. When several communications are required between the same pair of processors, they are simply serialized. The above model can be representative of dedicated network links among pairs of processors (e.g., as in a switch in the case of a fully connected topology). However, it is not representative of network infrastructures that are inherently shared such as an Ethernet or most of the Internet. In these latter cases communications between distinct processor pairs often interfere with one another because they share some network resource. In fact, a multi-port model is often not realistic for this very reason: it is unlikely that a processor could always communicate with many processors without any interference between the communications. Indeed, these communications share the processor's network card, which may be a bottleneck.

Given the above, there are two options. First, one can model a shared network medium as a link used for communication between several pairs of processors, and just impose the constraint in the schedule that the link is never used by more than one communication at a time. Second, one can allow multiple simultaneous communications on the link. In this case the typical model is that the bandwidth of the network link is shared fairly among the connections: if the link can achieve a data transfer rate of B (in Mbit/sec) and there are n connections using this link, then each connection proceeds at a data transfer rate of B/n. It turns out that allowing for bandwidth sharing can actually be beneficial (although no more bytes are pushed through the network per time unit with the above bandwidth sharing model than when connections are assumed to be serialized). Let us demonstrate this with a simple, and admittedly contrived, example. Consider an on-line job scheduling problem in which requests for computation, or *jobs*, arrive at a server and must be dispatched to any of several identical worker processors over a network link. Say that a job arrives at time t minutes that requires 10 minutes of data transfer for 10 minutes of computation, and that a job arrives at time $t + 5$ minutes that requires 1 minute of data transfer time for 19 minutes of computation. Finally, say that the performance metric is the average slowdown over all submitted jobs, where the slowdown is the ratio of a job's effective turn-around time to the turn-around time that could be achieved if the platform were dedicated to that job. This metric is commonly used for on-line job scheduling problems and must be minimized. Three possible scheduling strategies are: (i) first come first serve; (ii) wait until both jobs have arrived and schedule the second job first; and (iii) start jobs exactly when they arrive allowing simultaneous communication on the network link and assuming that each communication gets half of the available bandwidth. Other strategies are possible but can easily be shown to reduce to or be outperformed by one of the above. In this example, the average slowdown obtained for the three strategies are respectively 1.125, 1.15, and 1.050, showing that allowing the link to be shared is the best option. Note that another possibility would be to allow for interruptible communications by which any communication can be stopped and resumed at will, which can theoretically achieve an even better slowdown as strategy (ii) above (1.025 in our example). However interruptible communications pose several implementation problems, and more importantly incur overhead in practice due to network latencies

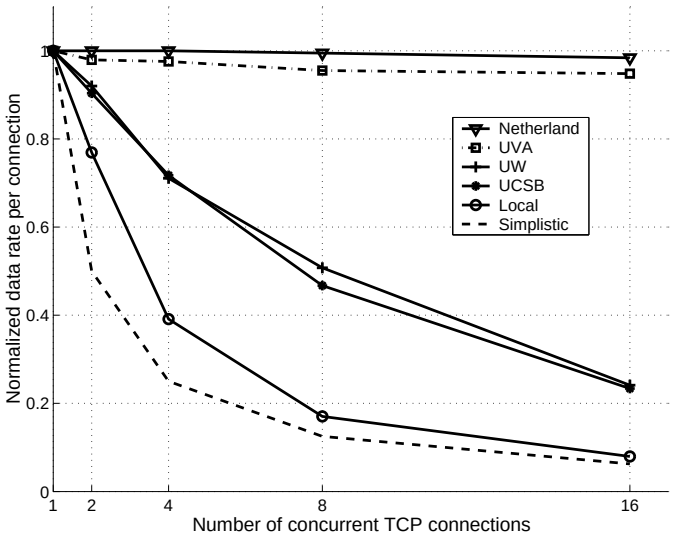

Fig. 1. Bandwidth-sharing experiments between a host at UCSD and one in the Netherlands, UVA, UW, UCSB, and the local-area network. The "Simplistic" model in which the bandwidth is shared proportionally to the number of connections is depicted as a dotted line.

and to TCP slow-start.

Consequently, not only is bandwidth sharing the more realistic model for many platforms, and in particular most Grid platforms, but it is justifiable to design scheduling strategies that exploit bandwidth sharing. In the next section, we will see that the simplistic bandwidth-sharing model introduced above is actually not representative of all networks. In fact, it is pessimistic in many cases, which observation makes a compelling case that there is an opportunity for scheduling algorithms to exploit the bandwidth sharing properties of the network infrastructure to achieve higher application performance.

3.1. A Bandwidth Sharing Experiment

In the example in the previous section we used the simplistic assumption that when x connections occur simultaneously on a network link that can achieve a data transfer rate B (in Mbit/sec) then each connection proceeds at B/x Mbit/sec. This model has been used in most previous scheduling work and is representative of what can be observed, for example, on a non-switched Ethernet local-area network. This model however is not adequate for wide-area networks. Figure 1 shows results from a simple experiments in which we performed data transfers of 100MB files over TCP between a host in our lab at the University of California, San Diego, and hosts in that same lab ("Local"), at the Delft Technical University, Netherlands ("Netherlands"), at the University of Virginia ("UVA"), at the University of Washington ("UW"), and at the University of California Santa Barbara ("UCSD"). (We also used larger file sizes but observed similar results.) For each of these destinations we initiated from 1 to 16 simultaneous TCP connections between a single host pair, with one of

the hosts always in our lab at UCSD. The graph in Figure 1 plots the achieved data transfer rate *per connection* versus the number of connections. The data transfer rate is normalized to that achieved when only one connection is used. Each data point corresponds to an average over 10 repetitions and the variances were small. In addition the graph plots the simplistic bandwidth-sharing model by which the total bandwidth is shared equally among connections (dashed line).

Note that this experiment has three limitations. First, the number of connections was kept relatively small, and the bottleneck link was constant and somewhere inside the network. This allows us to ignore, for now, the fact that a network path consists of multiple network links. All connections were initiated at approximately the same instant, and the file size used was large enough that we observe steady-state behavior, after the TCP slow-start phase. This means that our results may not be applicable to short-lived connections. But note that many Grid applications involve relatively large data transfers and would thus experience the same network behavior as in our experiments. Third, all connections are between the same host pair, so we can ignore, for now, all issues of network topology.

Based on the results shown in Figure 1 we can make the following observation: the simplistic bandwidth-sharing model, while somewhat accurate for the Local experiment, is widely inappropriate for any of the wide-area transfers. In fact, the curves for the two most distant sites, Netherlands and UVA, are roughly flat, meaning that connections beyond the first one get bandwidth "for free". The experiments for UW and UCSB show trends that are in between the Local experiment and the Netherlands and UVA experiments. The bandwidth-sharing behavior of the wide-area transfers is due to multiple factors. For instance, no matter how much bandwidth may be available on an Internet backbone, the portion of that bandwidth used by a TCP connection is limited by the sender's congestion window. Also, as backbone links typically support very large numbers of connections, the contention among the connections of a single application is often effectively negligible, at least up to a point. Our goal is not to obtain a rigorous explanation of the behaviors observed in our experiments, but rather to capture these behaviors in a high-level, empirical network model. Note that the fact that using parallel TCP streams can increase overall throughput as perceived from the application is a well documented fact (used for instance in Web browsers, or in data transfer tools like GridFTP [3]). Our point here is that the scheduling community has traditionally ignored this fact, and that, if relevant scheduling algorithms are to be developed for Grid platforms, it cannot be ignored any longer.

3.2. An Empirical Model for Bandwidth Sharing

Based on the results of the above experiment, we conclude that the simplistic bandwidth-sharing model is not applicable to wide-area links. For instance, for the long-haul connections, a more reasonable model is that each additional concurrent connection gets assigned the same amount of bandwidth as the previous connections. We term this last model "Ideal". Note that the curves corresponding to the UW and the UCSB experiment in Figure 1 exhibits a somewhat hybrid behavior, in

between that of a local-area network and a long-haul Internet connection. Rather than trying to fully understand the reasons for this behavior, we set out to obtain a simple empirical model of end-to-end data transfer rate per connection based on our experimental data, such as would be needed to develop scheduling algorithms that take such behavior into account.

Let B be the maximum data transfer rate that can be delivered by the bottleneck link on a path, b the observed data transfer rate per connection, n the number of connections opened by the considered Grid application, and N the total number of other connections on the same bottleneck network link due to other applications. The value of b is observed in our experiment, the value of n is set in our experiment, and the values of B and N are unknown. As a first approximation we take the following approach. We solve the equation:

$$B = b * (N + n),$$

which just says that the data transfer rate per connection times the number of connections is equal to the maximum achievable data transfer rate through the network. The above equation is only a very coarse approximation of the real world as, for instance, it is known that the round trip times (RTTs) of competing connections have a large impact on bandwidth sharing, and that on a bottleneck link inside the Internet competing connections exhibit a wide range of RTT values.

Nevertheless, by using two experimental points (say for $n = 1$ and $n = 8$) in our experimental data, the above equation can be solved to express b as a function of n, and the resulting model fits the experimental data quite well. For instance, when applied to the experiments between UCSB and UCSD, the average data transfer rate per connection for one connection is 3,000 kB/sec, and 1,400 kB/sec for eight connections. Therefore, after solving for N and B we obtain: $b = 18360/(n + 5.12)$. We can then compute the average relative error between this model and our experimental data. We did this for all our experiments, and Table 1 shows the results for: (i) our *empirical* model; (ii) the *simplistic* bandwidth-sharing model in which it is assumed that each connection achieved a transfer rate of b_1/n, where b_1 is the data transfer rate achieved for one single connection; (iii) the *ideal* bandwidth-sharing model in which every connection of the application achieves the same data transfer rate b_1. We can see that the empirical model tracks the behavior of the network within 10% accuracy for all our experiments, and does so better than the other two popular models. All details on the experiment and the development on the empirical model can be found in [58].

The conclusions from our preliminary evaluation of bandwidth sharing are as follows. First, the simplistic bandwidth-sharing model which is used traditionally in the scheduling literature is not applicable to wide-area transfers and thus cannot be used as a basis to develop algorithms that are appropriate for applications running on Grid platforms. Second, the ideal bandwidth-sharing model captures the behavior of long-haul wide-area transfers. Third, there are networks that exhibit a behavior that is in between the simplistic and the ideal models and for which it is possible to derive an accurate empirical model based on simple experimental

Table 1. Average relative error from experimental data for our empirical model, the simplistic model, and the ideal model, over all experiments.

Experiment	Empirical Model	Simplistic Model	Ideal Model
UNeth	3%	61%	3%
UVA	1%	67%	7%
UW	7%	57%	147%
UCSB	7%	51%	98%
Local	9%	19%	765%

measurements of bandwidth sharing. Fourth, to be general, a scheduling algorithm developed for Grid applications should most likely consider wide-area links that exhibit a range of behaviors, from the Simplistic to the Ideal models, with intermediate parameterized models similar to our empirical model.

4. Network Topology

The experimental results presented in the previous section suggest that a very large number of connections could be established "for free" over a long-haul backbone link, which is clearly not the case in practice. In fact, a host that participates in a distributed computation is never attached directly to a backbone link but, rather, via a network card with some limited capacity, which is itself connected to possibly multiple local-area links via a number of routers and eventually to the backbone. Therefore, when many connections are opened from a host to one or more remote hosts, the network bottleneck may shift from being some backbone link inside the network to being the host's network card, which in turns limits the number of connections that can be used effectively. This effect was not seen in Figure 1 given the moderate number of connections and the speed of the backbone link relative to that of the local links. Furthermore simultaneous communication from a single host to different sites (say from UCSD to both UCSB and UW) share network resources, in this case the network card and probably network links all the way to the first backbone link, and maybe other backbone links beyond the first one. Similarly, communication emanating from different hosts within a site most likely share local-area network resources on their way to the Internet backbone.

Consequently, a reasonable network model must consider a sequence of links, or a *path*, rather than only single-link connections. Note that these links can be logical links that each model a set of physical links, as further discussed in Section 4.2. The question then arises of how to model data transfers over network paths. One possibility is to use a store-and-forward model by which a message is sent in its entirety through each link in sequence. However, this is extremely unrealistic as in fact messages are split into packets, and packet transfers are pipelined over the network links. One possibility would then be to model the network at the packet level, accounting for the implementation of the TCP protocol. While this can be done for simulation (e.g., as in NS [45], DaSSF [36], OMNeT++ [46]), the resulting complexity would preclude the analyses necessary for the development of scheduling algorithms. In the next section we discuss higher-level models of TCP networks.

4.1. A Macroscopic Model of TCP

Researchers in the area of networking have striven to develop so-called *macroscopic* models of bandwidth sharing that ignore the details of packet-based network communications. Such models have been derived via an analogy between network connections, or *flows*, and fluids in pipes while ignoring packet granularity [38], and several authors have proposed theoretical models for TCP bandwidth allocation among flows [12, 19, 39]. The principle according to which n flows going through the same bottleneck link each receives $1/n^{\text{th}}$ of the bandwidth is called MaxMin fairness as discussed for example in [5]. By contrast, proportional fairness was introduced in [30] as an alternative to MaxMin fairness. In MaxMin fairness, a long flow gets assigned more overall network resources than a short flow (when considering the whole network path), which in some sense contradicts with the notion of a fair sharing of network resources. By contrast, in proportional fairness as described in [38, 47], flows sharing a bottleneck link receive bandwidth in a way that is inversely proportional to their RTT.

Other fairness schemes have been proposed, but the main question is: which macroscopic fairness principle does TCP implement? Or in other terms, considering that TCP operates at the microscopic level (i.e., at the packet level), what is the emergent behavior of TCP protocol at the macroscopic level? The consensus is that the fairness implemented by TCP networks is "close" to proportional fairness (for instance, it is known that TCP favors short flows). Note however that the work in [12] shows that TCP does not implement proportional fairness when flows exhibit different levels of packet loss. Nevertheless, considering that bandwidth is approximately shared inversely proportionally to the RTTs, it is commonly assumed that TCP does implement proportional fairness [19, 39].

Building on these considerations it is possible to design an efficient algorithm for computing the share of bandwidth allocated to TCP flows competing over multi-path routes. We have developed such an algorithm; due to lack of space, we only give here the main ideas and we refer the reader to [10] for all details. Consider a graph in which vertices are either endpoints or routers and edges are network links. Given a set of flows that each corresponds to communications between two endpoints over a fixed route, the algorithm first considers all links and determines bottleneck links for some of the flows. These flows are assigned bandwidth on these links inversely proportionally to their round-trip times, i.e., inversely proportionally to the latency for the corresponding network path. (Remember that as seen in Section 2 network link latencies are a fundamental part of the network model.) These flows consume this bandwidth end-to-end, and thus our algorithm reduces the bandwidth capacity of links traversed by all these flows in the network accordingly. This process is repeated until bandwidth has been allocated to all flows. The current version of this algorithm allows some links to exhibit the ideal bandwidth-sharing behavior described in Section 3.2 for long-haul wide-area links. It would be interesting to extend it to account for the empirical model developed in that section as well.

We have proved the correctness of our algorithm and validated it in [10] with the Network Simulator (NS) [45]. One key point here is that the bandwidth-sharing

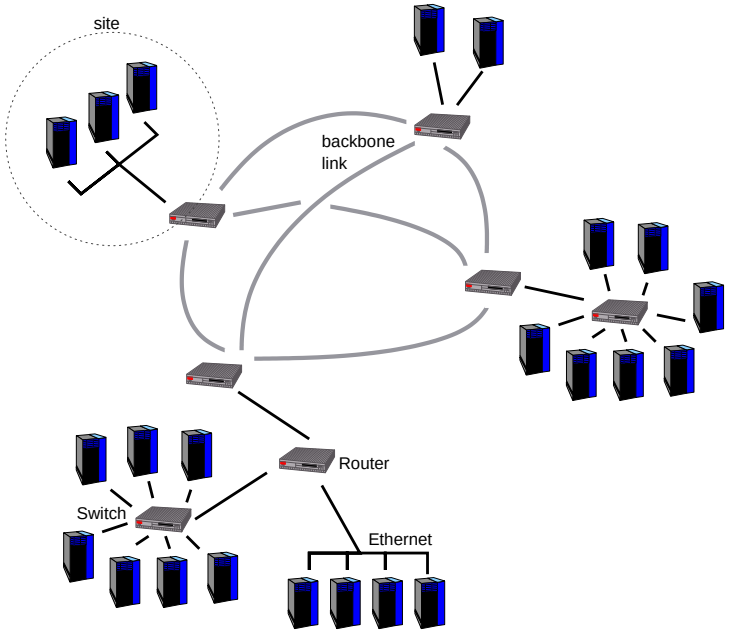

Fig. 2. Sample Grid Model

model that our algorithm implements is simple enough that it can be used directly for analyzing and developing scheduling strategies that take into account the different bandwidth-sharing properties of local-area and wide-area networks. See, for instance, our work in the area of divisible load scheduling over wide-area platforms [37].

4.2. A Grid Model

Our goal in this paper is to identify simple, yet realistic, platform models that should be used for conducting scheduling research that targets large-scale platforms. While it is difficult to precisely quantify this trade-off, our discussion so far has highlighted the following points: network latencies should be modeled, wide-area links should be modeled differently than local-area links, and a macroscopic model of TCP bandwidth sharing should be used. But the question of the network topology is still open.

Many measurement studies of the Internet topology have been conducted. Based on these measurements, simple power-laws properties (e.g., outdegree of node versus rank, eigenvalues of the adjacency matrix versus rank) as well as the structure of the Internet topology have been discovered [61, 18, 25, 41, 4]. These discoveries have led to several public-domain generators of "representative" Internet topologies [15, 27, 40, 54, 9, 2], which have been evaluated by several authors (see for instance [29, 8, 55]). While these generators constitute an excellent starting point for building a realistic simulation model, they do not provide all that is needed for the purpose of simulation in the context of Grid application scheduling research. For instance,

they do not contain information concerning the network traffic (e.g., the available bandwidth throughout time). Common techniques involve the use of simulated sources of traffic using random laws, which are seldom validated against real traffic. Also, there is no information on the characteristics of network links. As a result, several parameters have to be hand-tuned to obtain realistic platforms, and the "art" of building a realistic platform model requires both significant experience and good intuition. While this can be expected from researchers that specialize in networks, it is seldom the case for researchers in the area of scheduling.

Beyond the above limitations, perhaps the most significant problem faced by scheduling researchers wanting to use these generated topologies as a basis for their simulations is as follows. Since Grid algorithms are to be implemented at the application level, only very coarse information about topology and routing is available to them in practice. So while using complex topologies may be appropriate for simulation, more synthetic models that only use information available at the application level are needed for designing practical scheduling algorithms.

One possibility is to abstract the wide-area networking infrastructure as a fully connected network among the "sites" that form the Grid platform, where a site typically corresponds to an institution in a single geographical location. This simplifying model implies that there is no contention between application transfers on the Internet when the source and/or destination sites of these transfers are different. For instance, this model assumes that transfers from UCSD to UVA do not interfere with transfers from UCSD to UW on the Internet, but of course they can interfere with each other within the UCSD institution, i.e., before the connection to an Internet backbone. Based on our discussion of bandwidth sharing in Section 4, this Grid model differentiates between local-area links and wide-area links. For instance, bandwidth sharing on the wide-area links could be modeled by the empirical model developed in that section, or with the simple Ideal model. Within each institution it is reasonable to think that some knowledge of the network topology is directly available or that it can be easily discovered. For instance, the work in [32] describes an extension to the Effective Network View (ENV) software [51] by which effective layer-3 topologies can be discovered with application-level measurements and contention tests (we focus on layer-3 topology information as it can be obtained at the application level [17, 50, 23, 43, 26]). See [34] for an example of the type of topology that can be discovered with this tool. Note that ENV also annotates the (logical) network links with effective bandwidths. The bandwidth and bandwidth-sharing behavior of wide-area links can be identified via experiments similar to the one used to compute the data in Figure 1.

Figure 2 depicts a sample such Grid model with four sites. Wide-area links are shown as grey lines, while local-area links are shown in black. Even with such a seemingly complex topology, it is straightforward to exploit the bandwidth-sharing model of Section 4 to develop new scheduling algorithms analytically. This is demonstrated in our recent work on divisible load scheduling over wide-area network [37]. Note that the ENV approach could always be used to try to infer link sharing over the wide area and move beyond the fully connected assumption if needed. Finally,

end-to-end RTTs can be easily measured at the application level.

We claim that the above Grid model together with our bandwidth-sharing model is a good example of a model that strikes a good balance between being simple enough to be amenable to the analysis and the development of novel scheduling algorithms and yet significantly more realistic than models traditionally used in the scheduling literature. We have performed component-wise validation of this model (see [10, 58]).

5. The SimGrid Framework

While it is important to precisely define platform models, and in particular network models, as a basis for developing scheduling algorithms, it is often impossible to quantify the effectiveness of these algorithms and to compare them via direct analysis. Instead, scheduling researchers must perform experimental application executions over large numbers of application and platform scenarios in order to obtain statistically significant quantitative results. Given that experiments on real-world Grid platform are labor-intensive, necessarily constrained by the available infrastructure, and unreproducible, scheduling researchers typically resort to simulation. Examining the state-of-the-art of the simulation technology used in scheduling research we found two main problems. First, there is no simulation standard in this area, and researchers often build "throw-away" simulators using a variety of programming languages and tools (e.g., generic discrete-event simulation packages such as [49, 57, 42, 28]). This not only forces researchers to build simulations from the ground up, but also virtually precludes others from reproducing and extending results presented in research articles, which is a clear impediment to scientific advance. Second, while throw-away simulators were perhaps acceptable when platform simulation models were simple, we have seen in all the discussions above that current platforms such as Grids require more sophisticated models. Therefore, there is a clear need for a standard simulation framework whose Application Programming Interface (API) is specifically designed for scheduling research and for a simulation framework underneath that API that implements realistic platform models relevant to today's platforms. These considerations have motivated the development of the SimGrid simulation framework [32, 34].

The main principles of SimGrid are presented in [34]; the API documentation as well as several examples can be found on the project's Web page [52]. What is most important for the discussion in this paper is the network modeling capabilities of SimGrid: we have implemented most of the models discussed in Sections 2, 3, and 4 as part of the software. The SimGrid user can define network links with given latencies and bandwidths, and with either a simplistic or an ideal bandwidth-sharing behavior as described in Section 3.2. At the moment SimGrid does not support an arbitrary bandwidth-sharing behavior such as the empirical model present in that section. With such network links, it is possible to set up a network topology of links, routers, and switches, that interconnect compute resources. This topology can be completely arbitrary and built from the ground up by the SimGrid user using the API. It is also possible to import and annotate topologies generated by

the BRITE [40] generator. Finally, SimGrid can automatically import topologies discovered by the ENV tool [51] as described in [32]. No matter how the topology is created, SimGrid implements the macroscopic TCP model described in Section 4.1 via the algorithm developed in [10]. Therefore, SimGrid makes it straightforward to simulate an application execution on a platform such as the one depicted in Figure 1. Finally, one issue that we have not discussed is that of traffic on the network infrastructure due to applications and users beyond the ones simulated explicitly by the SimGrid user. Given that there is no widely accepted model for network traffic, SimGrid just allows for the bandwidth and latencies of network links to fluctuate according to arbitrary traces (i.e., time-stamped series of values).

6. Conclusion

In this paper we have discussed a number of issues pertaining to the modeling of large-scale computational platforms for the specific purpose of scheduling research. In particular we have focused on network models and have highlighted the shortcomings of modeling approaches used traditionally in the parallel computing area when applied to wide-area networks that are found in Grid platforms. We have made a case for the modeling of network latencies, presented empirical data and made a case for bandwidth-sharing models that reflect the behavior of TCP on real-world networks, and proposed a streamlined model for Grid computing platforms that is more realistic that used previously and yet amenable to analysis. We have also highlighted those models that we have implemented as part of our own simulation framework specifically designed for the evaluation of scheduling algorithms, SimGrid.

Although SimGrid has already been used successfully in over a dozen research projects as seen in articles published in the scheduling literature over the last couple of years, we plan to extend the simulation framework in several directions. We will investigate ways in which an arbitrary bandwidth-sharing model such as the empirical model derived in Section 3.2 can be integrated with our macroscopic TCP model. Also, we are currently extending the SimGrid API to take into account failures of resources, including network links, in order to provide a simulation environment in which scheduling algorithms that account for failures can be developed. Another interesting direction is that pointed out by a recently published article [31] in which it is shown that the traditional assumption of full overlap between communication and communication is not realistic in practice. In fact, and not surprisingly, a processor engaged in communication sees its processing speed decrease. The results in [31] provide the basis for modeling the effects of this interference. It would be interesting to extend SimGrid with such a model in order to allows users to study the impact of the actual overhead of overlapping communication with computation when scheduling parallel and distributed applications.

Acknowledgments

The author wishes to thank Arnaud Legrand and Loris Marchal whose work on

the SimGrid project as well as many fruitful discussions on the topic of modeling and simulation have helped generate some of the content of this paper, as well as Charles Wurster who performed the bandwidth sharing experiments presented in Section 3.1.

References

1. Adler, M., Gong, Y., and Rosenberg, A. Optimal sharing of bags of tasks in heterogeneous clusters. In *Proceedings of the 15th ACM Symposium on PArallelism in Algorithms and Architectures (SPAA'03)* (2003), pp. 1–10.

2. Aiello, V., Chung, F., and Lu, L. A randdom graph model for massive graphs. In *Proceedings of the 32nd Annual Symposium on the Theory of Computing* (2000).

3. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnal, D., and Tuecke, S. Data Management and Transfer in High Performance Computational Grid Environments. *Parallel Computing Journal 28(5)* (2002), 749–771.

4. Barabásis, A., and Albert, R. Emergence of scaling in random networks. *Science 286* (1999), 509–512.

5. Bertsekas, D., and Gallager, R. *Data Networks*. Prenctice Hall, 1987.

6. Bharadwaj, V., Ghose, D., and Mani, V. Multi-Installment Load Distribution in Tree Networks With Delays. *IEEE Trans. on Aerospace and Electronc Systems 31*, 2 (1995), 555–567.

7. Bharadwaj, V., Li, X., and Ko, C. On the Influence of Start-Up Costs in Scheduling Divisible Loads on Bus Networks. *IEEE Transactions on Parallel and Distributed Systems 11*, 12 (2000), 1288–1305.

8. Bu, T., and Towsley, D. On distinguishing between internet power-law generators. In *Proceedings of IEEE Infocom* (2002).

9. Calvert, K. L., Doar, M. B., and Zegura, E. W. Modeling Internet Topology. *IEEE Communications Magazine 35*, 6 (June 1997), 160–163.

10. Casanova, H., and Marchal, L. A Network Model for Simulation of Grid Application. Tech. Rep. 2002-40, Laboratoire d'Informatique du Parallélisme, ENS-Lyon, France, October 2002.

11. Cheng, Y.-C., and Robertazzi, T. Distributed Computation for a Tree-Network With Communication Delay. *IEEE transactions on aerospace and electronic systems 26*, 3 (1990).

12. Chiu, D. Some Observations on Fairness of Bandwidth Sharing. Tech. rep., Sun Microsystems, 1999.

13. Chun, G., Dail, H., Casanova, H., and Snavely, A. Benchmark Probes for Grid Assessment. In *Proceedings of the High-Performance Grid Computing Workshop* (April 2004).

14. Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K., Santos, E., Subramonian, R., and Von Eicken, T. LogP: Towards a Realistic Model of Parallel Computation. In *Proceedings of the ACM SIGPLAN Symposium on*

Principles and Practice of Parallel Programming (1993).

15. Doar, M. A Better Model for Generating Test Networks. In *Proceedings of Globecom '96* (Nov. 1996). Available at `http://citeseer.nj.nec.com/doar96better.html`.

16. DOE Science Grid. `http://www.doesciencegrid.org/`.

17. Downey, A. Using Pathchar to Estimate Internet Link Characteristics. *Measurement and Modeling of Computer Systems* (1999), 222–223.

18. Faloutsos, M., Faloutsos, P., and Faloutsos, C. On Power-law Relationships of the Internet Topology. In *Proceedings of SIGCOMM'99* (1999), pp. 251–262.

19. Floyd, S., and Fall, K. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking 7*, 4 (1999), 458–472.

20. Foster, I., and Kesselman, C., Eds. *Computational Grids: Blueprint for a New Computing Infrastructure*, 2nd ed. M. Kaufman Publishers, Inc., 2003.

21. Foster, I., Kesselman, C., Nick, J., and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG document, Global Grid Forum, June 2002.

22. Foster, I., Kesselman, C., and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications 15*, 3 (2001).

23. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking* (October 2001).

24. The Globus Toolkit. `http://www.globus.org`.

25. Govindan, R., and Reddy, A. An analysis of inter-domain topology and route stability. In *Proceedings of IEEE Infocom* (1997).

26. Govindan, R., and Tangmunarunkit, H. Heuristics for Internet Map Discovery. In *Proceedings of INFOCOM* (March 2000), pp. 1371–1380.

27. GT-ITM: Georgia Tech Internetwork Topology Models. `http://www.cc.gatech.edu/projects/gtitm/`.

28. Howell, F., and R., M. SimJava: A Discrete Event Simulation Package for Java with Applications in Computer Systems Modelling. In *Proceedings of the First International Conference on Web-based Modelling and Simulation* (Jan 1998).

29. Jaiswal, S., Rosenberg, A., and Towsley, D. Comparing the structure of power-law graphs and the Internet AS graph. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)* (2004), pp. 294–303.

30. Kelly, F. Charging and rate control for elastic traffic. *European Transactions on Telecommunications 8* (1997), 33–37.

31. Kreaseck, B., Carter, L., Casanova, H., and Ferrante, J. On the Interference of Communication on Computation. In *Proceedings of the workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems, Santa Fe, New Mexico* (April 2004).

32. Legrand, A., and Lerouge, J. MetaSimGrid : Towards realistic scheduling simulation of distributed applications. Tech. Rep. 2002-28, Laboratoire d'Informatique du Parallélisme, ENS-Lyon, France, July 2002.

33. Legrand, A., Marchal, L., and Casanova, H. Scheduling Distributed Applications: The SimGrid Simulation Framework. In *Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)* (May 2003).

34. Legrand, A., Marchal, L., and Casanova, H. Scheduling Distributed Applications: The SimGrid Simulation Framework. In *Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)* (May 2003).

35. Li, K. Parallel Processing of Divisible Loads on Partitionable Static Interconnection Networks. *Cluster Computing 6*, 1 (2003), 47–55.

36. Liu, J., and Nicol, D. M. *DaSSF 3.1 User's Manual*, pr 2001.

37. Marchal, L., Yang, Y., Casanova, H., and Robert, Y. A realistic network/application model for scheduling divisible loads on large-scale platforms. Tech. Rep. RR-2004-21, LIP, ENS-Lyon, April 2004.

38. Massoulié, L., and Roberts, J. Bandwidth sharing: Objectives and algorithms. In *Proceedins of INFOCOM* (1999), pp. 1395–1403.

39. Mathis, M., Semke, J., and Mahdavi, J. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communications Review 27*, 3 (1997).

40. Medina, A., Lakhina, A., Matta, I., and Byers, J. BRITE: Universal Topology Generation from a User's Perspective. Tech. Rep. 2001-003, Computer Science Department, Boston University, 1 2001.

41. Medina, A., Matta, I., and Byers, J. On the Origin of Power Laws in Internet Topologies. *ACM Computer Communication Review 30*, 2 (2001).

42. Miller, A., Nair, R., and Zhang, Z. JSIM: A Java-Based Simulation and Animation Environment. In *Proceedings of the 30th Annual Simulation Symposium (ANSS'97)* (April 1997), pp. 31–42.

43. Ng, T. E., and Zhang, H. Predicting Internet Network Distance with Coordinates-based Approaches. In *Proceedings of IEEE INFOCOM'02, New York, NY* (June 2002).

44. The NPACI Grid. `http://npacigrid.npaci.edu/`.

45. The network simulator - ns-2. `http://www.isi.edu/nsnam/ns`.

46. The OMNeT++ Project. `http://www.omnetpp.org/`.

47. Roberts, J., and Massoulié, L. Bandwidth sharing and admission control for elastic traffic. In *Proceedings of the ITC Specialist Seminar, Yokohama* (October 1998).

48. Rosenberg, A. Sharing Partitionable Workloads in Heterogeneous NOWs: Greedier Is Not Better. In *Proceedings of the 3rd IEEE International Conference on Cluster Computing (Cluster 2001)* (2001), pp. 124–131.

49. Schwetman, H. CSIM: A C-based, process oriented simulation language. In

Proceedings of the 1986 Winter Simulation Conference (December 1986), pp. 387–396.

50. Seshan, S., Stemm, M., and Katz, R. SPAND: Shared Passive Network Performance Discovery. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems* (1997).

51. Shao, G., Berman, F., and Wolski, R. Using effective network views to promote distributed application performance. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications* (June 1999).

52. The SimGrid Project. `http://grail.sdsc.edu/projects/simgrid`.

53. Smarr, L., Chien, A., DeFanti, T., Leigh, J., and Papadopoulos, P. The OptIPuter. *Communications of the ACM 46*, 11 (Nov 2003), 58–67.

54. Sugih, J. Inet-3.0: Internet Topology Generator. Tech. Rep. 456-02, Computer Science Department, University of Michigan, 2002.

55. Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S., and Willinger, W. Network topology generators: Degree-based vs. structural. In *Proceedings of ACM SIGCOMM, 2002* (2002).

56. TeraGrid. `http://www.teragrid.org/`.

57. Toh, S. SimC: A C Function Library for Discrete Simulation. In *Proceedings of the 11th Workshop in Parallel and Distributed Simulation* (January 1993), pp. 18–20.

58. Wurster, C. Accounting for Concurrent Connections when Scheduling Parallel Applications . Master's thesis, Dept. of Computer Science and Engineering, University of California, San Diego, June 2004.

59. Yang, Y., and Casanova, H. Extensions to the Multi-Installment Algorithm: Affine Costs and Output Data Transfers. Tech. Rep. CS2003-0754, Dept. of Computer Science and Engineering, University of California, San Diego, July 2003.

60. Yang, Y., and Casanova, H. UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)* (April 2003).

61. Zegura, E., Calvert, K., and Donohoo, M. A quantitative comparison of graph-based models for internetworks. *IEEE/ACM Transactions on Networking 5*, 6 (1997), 770–783.