Survey of TeraGrid Job Distribution: Toward Specialized Serial Machines as TeraGrid Resources

Arvind Gopu, Richard Repasky, and Scott McCaulay

Abstract— As we proceed towards the age of petascale computing, it is important to be aware that even today more than half of national cyberinfrastructure users are serial users who run single processor code; more over, coarse-grained parallel application do not necessarily benefit from a high-speed low-latency interconnect. While a majority of compute resources on the TeraGrid today are massive parallel machines with high-speed low-latency interconnects like Myrinet or Infiniband, optimized to run large fine-grained parallel applications that use hundreds of processors/cores in parallel, usage patterns indicate that there is still considerable demand that could be just as effectively met by large computational resources with no special high-speed or low-latency interconnects. Research allocations involving large serial applications or coarse-grained parallel applications could be allocated to these machines, thus possibly leading to decreased wait times for massive parallel and large serial jobs. This change in focus would also lower the financial barrier for potential new resource providers to the national cyberinfrastructure, by allowing them to reallocate funds from the interconnect to additional computational capacity.

Index Terms—Serial, parallel, coarse-grained, fine-grained, MPI, interconnect, Myrinet, Infiniband.

1 Introduction

Petascale computing is one of the objectives of the National Science Foundation (NSF) in the near future [1]. The TeraGrid [5] is NSF's flagship effort to create a national cyberinfrastrucutre and to move forward toward achieving the petascale computing goal. Some of the biggest research projects that use TeraGrid resources run massive parallel applications [6, 7]. Consequently, a majority of the current computational resources on the TeraGrid are massive parallel machines with specialized high-speed low-latency interconnects like Myrinet [3] or Infiniband [4]; these machines are primarily designed to run fine-grained parallel applications that use hundreds of processors/cores in parallel and also have a significant communication element.

But is there a significant component of research that mainly uses serial applications on TeraGrid resources? Also, do users who run coarse-grained parallel applications need a high-speed low-latency interconnect to run optimally? We believe we have an answer to the first question based on usage characteristics of TeraGrid users over a two-year period. We have discovered that more than 50% of the jobs run on the TeraGrid in the October 2004-06 time period were single processor jobs; jobs that we will refer to as *serial jobs* in the rest of this paper. While there is a valid utility associated with allowing shorter serial jobs to run on parallel machines – they increase overall resource utilization by allowing backfill via. the scheduler [8] – larger serial jobs most often lead to longer wait times for parallel jobs, or endure long wait times themselves (for reasons explained later). It is also well documented that coarse-grained parallel applications, with negligible amounts of communication involved, might not necessarily need an expensive high-speed low-latency interconnect [9, 10].

Given that there is still significant demand (of compute cycles) that could be just as effectively met by a large computational resource with no special high-speed or low-latency interconnect, TeraGrid research allocation requests involving large serial or coarse-grained parallel applications could be allocated to such machines at the POPS allocation stage [2]; doing so might not only lead to decreased wait times for both parallel and large serial jobs but also will allow new resource providers to get their feet wet as a Grid resource provider – get used to the TeraGrid accounting process, installing grid software, etc. without the additional responsibility of having to maintain parallel hardware and software (which can be a rather substantial task). More over, high-speed low-latency interconnects have historically constituted 20-30% of total system cost. Focusing on acquiring serial machines without such interconnects will lower the financial barrier for potential new resource providers to the national cyberinfrastructure, and will also enable any resource provider to get more computational power (in terms of TeraFLOPS) by reallocating funds from the interconnect to computational capacity.

2 TeraGrid Job Distribution Based on Processor Count

The TeraGrid central accounting database maintains usage records for all jobs run on TeraGrid resources. We queried that database to retrieve usage on all systems of capacity 1 TeraFLOP or more, over a two year period (October 2004-06). All data collected was completely anonymous: no information about the users or their affiliation was collected.

Over 1.8 million jobs had been submitted during that time period, out of which almost a million jobs were serial (single processor) jobs. The distribution based on number of processors used per job, for all jobs submitted to TeraGrid resources 1 TF or more, between October 2004-06 is shown in fig.1. As is evident by looking at the graph, all multi-processor jobs add up to about 800,000 jobs. This indicates that a lot of computational research still happens with the use of serial applications that don't scale over multiple processors/cores. Researching into the category of parallel jobs – whether they were fine or coarse-grained – was beyond the scope of our analysis in this paper.

- *Arvind Gopu, Indiana University, agopu@indiana.edu*
- *Richard Repasky, Indiana University, rrepasky@indiana.edu*
- *D Scott McCaulay, Indiana University, smccaula@indiana.edu*

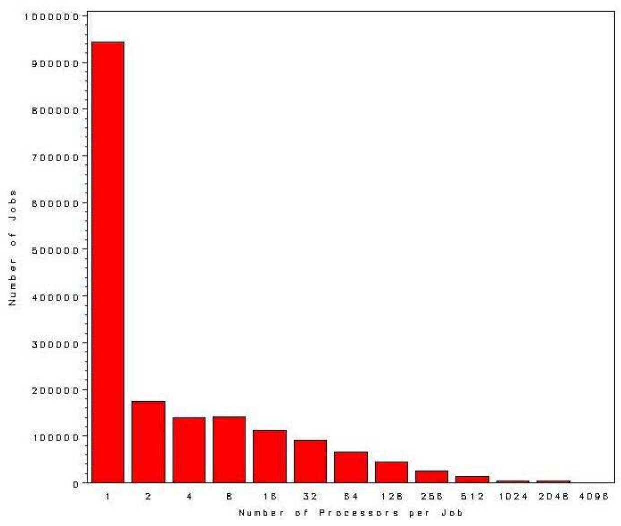

Fig.1. Plot showing Number of jobs vs. Processor Count per Job

3 Analysis of serial job characteristics on the TeraGrid

Given that more than half of the jobs submitted to TeraGrid resources are serial jobs, we will analyze their characteristics in more detail. In particular we will delve into how short serial jobs are used for backfill (defined in the subsection below) and how larger serial jobs can endure long queue wait times themselves and/or can lead to parallel jobs enduring long queue wait times.

3.1 Scheduler Backfill on Parallel Systems

To maximize overall resource utilization while also preventing excessive delays in starting large parallel jobs, most massive parallel machines allow for scheduler backfill using short single processor and/or small multi-processor jobs. A backfill capable scheduler allows jobs with smaller resource requirements to continuously use up resources before it can accumulate enough resources to run a larger job [8, 11]. To illustrate the concept of backfill, let us consider the following scenario involving two users: one user submits a large parallel job, the other submits a set of short serial jobs; while the resource manager is collecting resources – compute nodes – for the parallel job, it might allow several of the serial jobs to run on the compute nodes it has collected so far, provided those serial jobs will finish before the time it expects to have all the nodes required for the parallel job.That is essentially what backfill capable schedulers do.

3.2 Backfill vs. Increased Queue Wait Times

Many parallel machines are configured to prefer massive parallel jobs over smaller parallel jobs over single processor jobs in that order, to ensure optimal utilization of their expensive interconnect. Yet, as mentioned earlier, from a resource provider perspective, smaller jobs, especially short serial jobs, are excellent candidates to provide backfill [11]. The backfill scheme explained in section 3.1 works very well provided:

- The serial jobs under discussion are submitted *after* the parallel job has been submitted, and
- The wall time requirement for the serial jobs is such that they will *finish before* the scheduler can accumulate all the required nodes for the parallel job.

The scheme does not work very well when one or more serial jobs are submitted *before* the parallel job is submitted; in that case, the parallel job might end up waiting till some of the (already running) serial jobs complete because the resource manager might not have enough free resources to schedule the parallel job to start.

The scheme also does not work well when users submit serial jobs with really large wall-clock time requirement, jobs that cannot finish in time for the scheduler to allow them to start via backfill; in this case, the serial jobs may end up waiting much longer on the queue.

3.3 Wall-clock time Distribution of Serial Jobs on TeraGrid

We analyzed the usage data we retrieved from the TeraGrid central usage database, and tabulated job wall-clock times for all the serial jobs that were submitted in the October 2004-06 time period on resources bigger than 1 TF. The results are shown below in table 1.

Table 1: Distribution of Serial Jobs on TeraGrid

Job Wall-clock Time (hours)	Number of Jobs	Percentage of Jobs
0 – 1	597699	63.67%
1 – 2	157170	16.74%
2 – 5	136803	14.57%
5 – 10	32308	3.44%
10 – 20	9146	0.97%
20 – 50	5262	0.50%
50+	428	0.05%

The wall-clock time distribution for serial jobs tabulated in table 1 is plotted in fig.2. As is evident, over 60% of the serial jobs take one hour or less to complete. These make perfect candidates for backfill on large parallel machines, and are already increasing system utilization on TeraGrid resources.

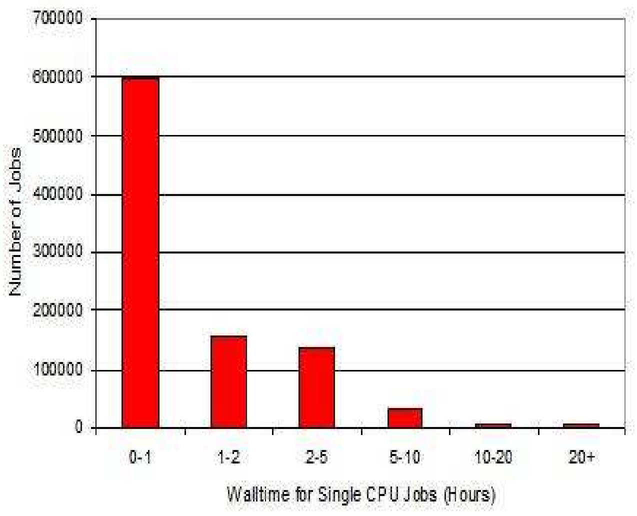

Fig.2. Plot showing Number of jobs vs. Serial job wall-clock

In the above graph, the jobs that take more than an hour of wall-clock time constitute approximately 37% of all serial jobs. Considering the scenario explained in section 3.1, it is conceivable that they could have waited for a longer time in their queue; or they could have caused parallel jobs submitted after them to wait for a longer time in the queue (because the scheduler was waiting for them to finish and free up compute nodes for the larger parallel job). Short parallel jobs running coarse-grained parallel codes also fall under the same category as the above larger serial jobs – quantifying those jobs would be very interesting but is out of the scope of our analysis in this paper.

4 POSSIBLE IMPROVEMENT: ALLOCATION OF SERIAL AND COARSE-GRAINED PARALLE JOBS TO RESOURCES WITH NO HIGH-SPEED INTERCONNECT

We believe it is worth considering categorization of research allocation requests that use serial applications into two categories – requests that use:

1. **Short Serial Jobs:** Serial jobs that have short wall-clock times, say, maximum wall-clock time of one hour; Monte Carlo simulations, or experiments where a short application is repeatedly run several hundred or thousand times, fall into this category. As explained before, these jobs would make for excellent cases for back-fill because the scheduler knows the job's wall-clock time requirement is short and can afford to schedule them while it is collecting resources for a parallel job.
2. **Large Serial Jobs:** We had already shown about 37% of serial jobs on the Teragrid require more than an hour of wall-clock time. We also researched locally at Indiana University, and found that more than half of our HPC/Grid computing users run serial applications that need several days' worth of wall-time. While it might seem like there is a need for these codes to be optimized and/or parallelized, it is also necessary to remember that old legacy applications are not always straight-forward to optimize or parallelize. These users, when they submit their jobs on large parallel resources, may end up facing one of two situations explained before in section 3.1 – they either start running before a massive parallel job is submitted thus clogging up the scheduler's capabilities to schedule the latter (till the serial job finishes); or they get stuck in the queue for long periods of time, because the scheduler does not let them run given their long wall time requirement.

It might also be worth considering the classification of allocation requests that use parallel applications into two categories – requests that use:

1. **Coarse-grained (Embarrassingly) Parallel Applications:** Coarse-grained parallel applications that do not have a significant amount of communication (message passing) do not require high-speed low-latency interconnect [9, 10], and can run on a serial machine (without any high-speed or low-latency interconnect) with almost no or very minimal performance hit.
2. **Fine-grained Parallel Applications:** Parallel applications that have a significant communication element (message passing), for example, codes that process grids of data and so forth, fall under this category. These applications are perfect for massive parallel compute systems with a high-speed low-latency interconnect like Myrinet [3] or Infiniband [4], and would optimally use such systems *and* their expensive interconnect.

If the above categorizations of serial and parallel applications are made, we believe that there is a good case to be made for specialized serial resources – ones with no high-speed low-latency interconnect. TeraGrid resource providers, possibly existing ones but especially potential new resource providers, could consider providing such serial resources (at least, to start with). Then research allocations involving either large serial applications or coarse-grained parallel applications could be allocated to these machines, possibly leading to decreased wait times for both large parallel and large serial jobs. Allocations that use small serial applications (say, with a wall-clock time requirement of one hour or less) could still be allocated to larger parallel systems with expensive interconnects, and these jobs could be used for backfill thus retaining the advantage of increased utilization of those resources. This change in focus (i.e. offering serial machines with no expensive intercomnect) would also lower the financial barrier for potential new resource providers to the national cyberinfrastructure, by allowing them to reallocate funds from the interconnect to additional computational capacity. And this change would enhance the user experience for both parallel and serial users, with likely earlier job start times. It also might provide an opportunity for new resource providers to get their feet wet in becoming part of the grid without having to worry about maintaining specialized parallel hardware and software, a task that can be very involved.

5 SUMMARY

In this paper, we presented job distribution characteristics derived from usage data collected off the TeraGrid central usage database. We showed the distribution of single processor (serial) jobs versus multi-processor (parallel) jobs. We explained the utility of short serial jobs for scheduler backfill. We explained how TeraGrid resource providers might consider offering serial resources with no high-speed or low-latency interconnect, and possibly reduce queue wait times for both massive parallel and large serial jobs. We put forth a case for considering allocation of requests that involve large serial or coarse-grained parallel applications to such specialized serial resources. We also explained the added benefits to resource providers in offering such serial resources, without expensive interconnects, especially for new resource providers.

ACKNOWLEDGMENT

We wish to thank Larry Simms for his assistance with analysis and visualization of the TeraGrid usage data using SAS. We also wish to thank David Hart and Chris Baumbauer of the TeraGrid Accounting working group for permitting us to query the TeraGrid central database to collect usage data used on this paper.

REFERENCES

[1] NSF High Performance Computing System Acquisition: Towards a Petascale Computing Environment for Science and Engineering: http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=13649
[2] Partnership Online Proposal System (POPS): https://pops-submit.ci-partnership.org/
[3] Myricom Home Page: http://myri.com/
[4] Infiniband Trade Associated Home: http://www.infinibandta.org/home
[5] TeraGrid Home Page: http://teragrid.org/
[6] Dong, S., Karniadakis, G. E., and Karonis, N. T. 2005. Cross-Site Computations on the TeraGrid. Computing in Science and Engg. 7, 5 (Sep. 2005), 14-23.
[7] Theoretical and Computational Biophysics Group at UIUC – Software Page: http://www.ks.uiuc.edu/Development/
[8] MOAB Scheduler Backfill function: http://www.clusterresources.com/products/mwm/docs/8.2backfill.shtml
[9] Ron Pepper and Rinku Gupta, "Designing High Performance Clusters", http://www.dell.com/downloads/global/power/ps1q05-20040175-Pepper.pdf
[10] How to Determine the Correct Interconnect Technology for an HPC Cluster: http://shareit.jotxpert.net/WikiHome/Articles/88671
[11] IBM Cluster Information Center – Using the Backfill Scheduler: http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.loadl.doc/loadl34/am2ug30514.html