RESOURCE PREDICTORS IN HEP APPLICATIONS

J. Huth*, S. Grinstein, P. Hurst, Harvard University, USA
J. M. Schopf, Argonne National Laboratory, USA

Abstract

The estimation of resource needs for data manipulation is fundamental to the operation of the Grid. Situations will arise when it will be necessary to determine which is more expedient, transferring a replica from a remote site or recreating the data from scratch. This paper explores the possibility of predicting end-to-end file transfer times, and then using these file transfer predictions along with file recreation predictions in a model framework to expedite data instantiation.

INTRODUCTION

The ATLAS (A Toroidal LHC ApparatuS) experiment [1] will use a data distribution system in which datasets are replicated across collaboration sites. In this scheme no single site is likely to be able to store all the datasets; however, a single dataset may be available at many locations. Many of these datasets can also be recreated locally based on a limited number of inputs. Users wishing to instantiate a dataset might choose to download a replica from a remote site or recreate it from scratch locally. Making an optimal choice will require estimates of both the time necessary to download the file and the time necessary to recreate it.

In previous work [2] we have demonstrated estimates of the execution times of ATLAS applications that could be used to recreate datasets. This paper describes our efforts first to estimate file transfer times and then to implement a software tool which uses file transfer estimates and recreation time estimates to instantiate a dataset in the shortest time possible. Our results show that transfer times for typical ATLAS data files over quiet network channels can be estimated to approximately 5%, though our estimate increases significantly when the network is busy. We have tested our software tool on a number of Chimera [3] job files which implement a simple strawman analysis chain and have realized time savings consistent with the assumptions of the strawman model.

PREDICTING FILE TRANSFER TIMES

Many different groups are investigating sophisticated file transfer predictions and mechanisms for dynamically optimizing transfer times[4, 5]. Our goal is to investigate a framework which uses transfer time predictions along with recreation time predictions to optimize dataset instantiation. We can test the operation of the framework using relatively unsophisticated file transfer predictions.

We use a simple model based on end-to-end historical data from GridFTP [6] logs to predict file transfer times. We use the average observed bandwidth with no file-size filtering. This simple model works well on the machines and networks we used during our tests.

We transferred files stored on disks at Brookhaven National Laboratory (BNL) and CERN to machines at Harvard. The bulk of the transfers originated on a particular BNL machine, aftpexp01.bnl.gov, with 4 3GHz Xeon processors and 2GB of RAM, running Linux 2.4.21-37.ELsmp, and having a 1.0GBit/s connection to the BNL network. The target Harvard machine has 2 3.4GHz P4 processors and 1.5GB of RAM, runs Linux 2.4.20-21.EL.cernsmp, and has a 1.0 GBit/s connection to Harvard's network. A smaller number of files were transferred from the Linux-based CASTOR servers at CERN. The local BNL, Harvard, and CERN networks are connected by high speed backbones. Typical routes are from BNL over ESnet to the Manhattan Landing switch, then via the Northern Exchange network to Harvard. Average latency from BNL is approximately 7.8 ms. Files originating at CERN arrive in Chicago via LHCnet, travel over Abilene to Manhattan, and then arrive at Harvard via the Northern Exchange. Average latency from CERN is approximately 148 ms.

To establish an initial benchmark 20 files each of 25MB, 50MB, 100MB, 250MB, 500MB and 1GB were transferred from BNL to Harvard. A plot of the transfer time in seconds versus file size in MB is shown in Figure 1.

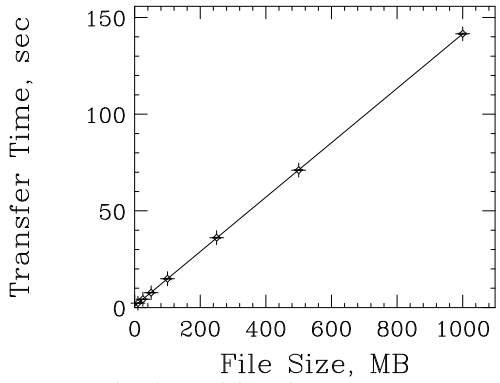

Figure 1: Transfer time of files from BNL to Harvard versus file size.

The average file transfer times are linear with file size.

* huth@physics.harvard.edu

This initial benchmarking was done when the machines and networks were quiet. Transfer times for 100MB files (typical for recent ATLAS data files) copied from BNL to Harvard have a variance less than 5%.

Tests of our optimization application occurred during the same period that "Service Challenge 3" was transferring large numbers of files. Transfer times were still linear with file size during this period, but the variance in transfer times increased; we observed a variance of approximately 90% for 100MB files, though the variances for larger files and files transferred from CERN were approximately 55% and 10%, respectively. A plot of transfer times versus file size for transfers from BNL to Harvard during this period of high network traffic is shown in Figure 2.

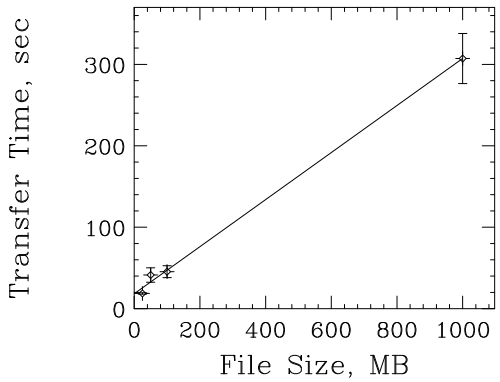

Figure 2: Transfer time of files from BNL to Harvard versus file size for transfers taking place during times of high network traffic.

OPTIMIZATION APPLICATION

Having predictions for the execution time necessary to recreate a dataset and for the time necessary to transfer a dataset, we have developed an optimization application to instantiate the dataset in the least amount of time. The algorithm and much of the implementation are general, but for demonstration purposes we have designed the optimization application to be used as a post-processor to optimize the directed acyclic graphs (DAG's) produced by Chimera. Our implementation also uses the Globus Replica Location Service (RLS) [7] to index input and output files, and stores historical execution and bandwidth data in a MySQL database [8].

A block diagram of the optimization workflow is shown in Figure 3. A shell script parses the DAG (and the underlying job submission files) looking for output file names, binary names, and execution parameters (especially the number of events to be processed). The script searches for the output files indexed in the RLS, and their host locations and file sizes are determined. A client then queries the database for the performance parameters for the particular binaries and for the historical bandwidths between the files' host locations and the local site. The script uses the information about the file locations and sizes along with the bandwidth information to predict the file transfer times.

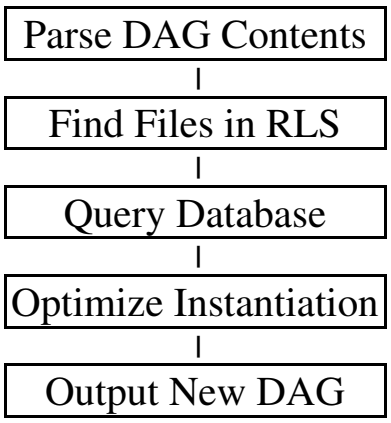

Figure 3: Block diagram showing the workflow of the optimization application.

The script also uses the performance parameters from the database with the execution parameters from the DAG to predict recreation times. The job submission files are then rewritten to either transfer the output file from the remote host or to recreate it locally, whichever is faster.

APPLICATION TESTS

We tested our optimization application on a number of simple DAG's. Our strategy was first to generate a "non-optimized" DAG which used a random mix of recreations and transfers to instantiate a series of test datasets. We ran our application on this DAG to produce an "optimized" DAG which minimized the time taken to instantiate each of the datasets. We then submitted both DAG's to our local queue and compared the total times taken by the DAG's to instantiate the full series of datasets. We expected the "optimized" DAG to take less total time.

We currently find that typical ATLAS execution times are long compared with typical transfer times. For example, the current ATLAS full event reconstruction takes approximately 20 minutes on our local Harvard machine to process 50 events and produce a 100MB output file, while this same 100MB file can be transferred from BNL to Harvard in approximately 15 seconds. Currently the choice between transferring files and executing ATLAS code locally is clear. However, we envision a time of faster binaries and more contentious networks in which transfer and recreation times are more nearly equal, and the choice is not so clear at the outset. We have chosen to test this equal time scenario by using a simplified binary (keg, the Kanonical Executable for Grids) [9] which has an adjustable execution time which can be matched to the transfer time.

Strawman Model

We use a Monte Carlo technique to generate non-optimized DAG's which instantiate a series of datasets. We begin by randomly choosing to instantiate one of several remote files (4 at BNL, 2 at CERN) indexed in our local RLS. One we have selected a file we know the time necessary to transfer it and can assign it a recreation time comparable to the transfer time. We select this recreation time from a uniform distribution extending from 0 seconds to twice the transfer time for the file. The average recreation time, therefore, is equal to the transfer time, but with substantial variance. Finally, we choose randomly either to instantiate by transferring or to instantiate by recreation, and then write this choice into the job submission scripts of the non-optimized DAG. On average, then, these non-optimized DAG's should have a random mix of transfers and recreations and have a total processing time equal to the sum of the transfer times of the series of files it instantiates.

When we operate on this non-optimized DAG with our optimization application it will select the quickest means of instantiating a particular dataset and rewrite the job submission scripts. From the details of the model we can predict *a priori* that the optimized DAG's should take 25% less time on average than the non-optimized DAG's.

RESULTS

We generated 23 non-optimized DAG's each of which instantiated either 10, 20, or 40 datasets. We processed these non-optimized DAG's and measured their total running time. We operated on these non-optimized DAG's with out optimizing application and produced optimized DAG's. We then processed these optimized DAG's and measured their total running time. The results are shown in Figure 4. The circles show the total running time of the non-optimized DAG's versus the predicted time for these non-optimized DAG's, while the diagonal crosses show the total running time for the optimized DAG's.

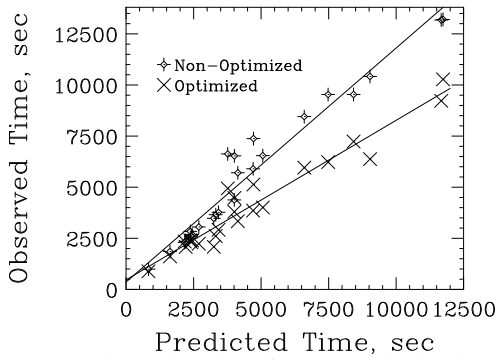

Figure 4: Total running time for non-optimized and optimized DAG's versus predicted time for the non-optimized DAG's.

The total running time for the non-optimized DAG's agrees reasonably well with the prediction. The running time for the optimized DAG's is generally less than that of the optimized DAG's, as expected. Figure 5 shows a plot of the ratio (optimized run time)/(non-optimized run time) versus run time for the non-optimized DAG. We find that on average the optimized run times are 28% less than the non-optimized run times, as expected from the parameters of our strawman model.

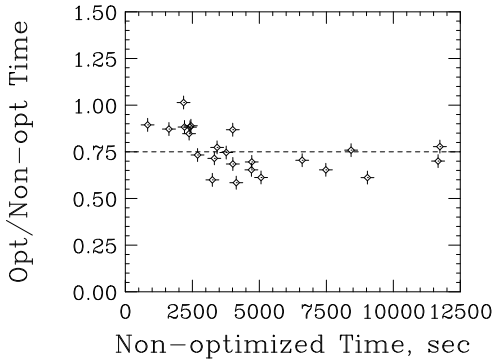

Figure 5: Ratio of run times of optimized and non-optimized DAG's versus run time for the non-optimized DAG's.

SUMMARY

We have developed an application which successfully uses simple predictions of file transfer times and dataset recreation times to optimize dataset instantiation. In many current instances it may be clear that one or the other method is fastest. We envision scenarios in which file transfer and recreation times are nearly equal and there is no clear *a priori* choice. In these equal–time scenarios our application can be used to dynamically optimize instantiation times. Tests on an equal–time strawman model demonstrate a significant time savings.

ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant Number PHY-0218987.

REFERENCES

[1] ATLAS Collaboration, "ATLAS Detector and Physics Performance Technical Design Report, CERN-LHCC-99-14-15" (1999).

[2] S. Grinstein, J. Huth, and J. Schopf, "Resource Predictors in HEP Applications" CHEP 2004, Interlaken, Switzerland.

[3] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation", 14th International Conference on Scientific and Statistical Database Management (SSDBM'02)

[4] S. Vazhkudai, J. Schopf, "Using Regression Techniques to Predict Large Data Transfers", The International Journal of High Performance Computing Applications (IJHPCA), special issue on Grid Computing: Infrastructure and Applications, Vol 17, No. 3, August 2003.

[5] T. Kosar, G. Kola and M. Livny, “A Framework for Self-optimizing, Fault-tolerant, High Performance Bulk Data Transfers in a Heterogeneous Grid Environment”, Proceedings of 2nd Int. Symposium on Parallel and Distributed Computing (ISPDC2003), Ljubljana, Slovenia, October 2003.

[6] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, “Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing”, IEEE Mass Storage Conference, April 2001, San Diego, California.

[7] A. L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, and R. Schwartzkopf, “Performance and Scalability of a Replica Location Service”, Proceedings of the International Symposium on High Performance Distributed Computing Conference (HPDC-13), June 2004.

[8] http://www.mysql.com

[9] GriPhyN Collaboration, GriPhyN Technical Report 2002-10.