[illegible]

Navigation in Grid Space with the NAS Grid Benchmarks (*Extendend Abstract*)

Michael Frumkin, Robert Hood[+]
NASA Advanced Supercomputing Division
[+]Computer Sciences Corporation
M/S T27A-2, NASA Ames Research Center
Moffett Field, CA 94035-1000
{frumkin,rhood}@nas.nasa.gov

Abstract

We present a navigational tool for computational grids. The navigational process is based on measuring the grid characteristics with the NAS Grid Benchmarks (NGB) and using the measurements to assign tasks of a grid application to the grid machines. The tool allows the user to explore the grid space and to navigate the execution of a grid application to minimize its turnaround time. We introduce the notion of gridscape as a user view of the grid and show how it can be measured by NGB. Then we demonstrate how the gridscape can be used with two different schedulers to navigate a grid application through a rudimentary grid.

Keywords: computational grids, navigation, benchmarks

1 Introduction

Computational and data grids [6, 13] provide environments where users can run distributed applications and work with distributed data. Dynamically changing grid resources, a variety of access policies, and interference with other grid users make it difficult to keep track of the tasks of a grid application and to navigate the application through the grid. One approach to navigate[1] on the grid is to rely on a meta-scheduler (or a superscheduler, cf. [20]) which will schedule the tasks of the application. This is not sufficient for a navigation to be successful since resource scheduling has different objective function than the application turnaround time. Our approach to navigation is user centric: obtain a map of the grid and then assign tasks of the application to the appropriate grid machines using a scheduling algorithm.

The problem of navigation of distributed applications has been a subject of scheduling research since the appearance of distributed systems and has significantly intensified with development of computational grids. The scheduling component of the navigation was formulated and partially addressed in early grid works [26, 25], where a scheduling mechanism was embedded into the application. It was realized that "a grid programmer cannot rely on resource schedulers or other system components to promote application performance goals" [26, p. 280]. We will discuss more related projects in the final verion of the paper.

We show that navigation of data flow grid applications can be accomplished with a *gridscape* – a map of the grid resources. The static component of the gridscape can be obtained using grid information services. Then the

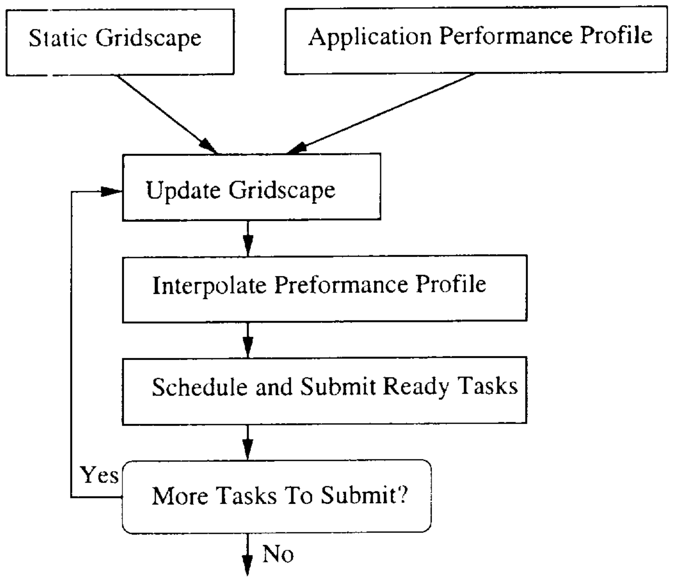
[1]We use term "navigate" according to the definition of Merriam-Webster's Collegiate Dictionary: **navigate** – to steer a course through a medium.

gridscape can be detailed with dynamic, application specific information measured by means of NAS Grid Benchmarks (NGB). To demonstrate the efficiency of this navigation process we implement a Navigation Pad that provides all basic navigational capabilities of observing the gridscape, scheduling and monitoring grid applications. We demonstrate the navigation process in a rudimentary grid using an implementation of NGB in Java and Fortran.

2 The Navigational Cycle

To run, or rather to *navigate*, an application in a grid environment, the users should be able to obtain a gridscape, a map of the grid, and update it at the time they are making decisions about requesting grid resources. The user's ability to make an informed decision on the choice of the appropriate grid resources and to avoid details having insignificant effects on the application turnaround time depends on the *resolution* of the gridscape. Since the grid is comprised of multiple resources whose load and properties are changing dynamically then the gridscape should be *dynamic* so that the user will have a good chance to request and obtain the best available resource.

For a grid application comprised of a number of coarse computationally intensive tasks or components communicating through ports the navigating process can be accomplished by iterating the *navigational cycle* of Figure 1.



Static Gridscape
Application Performance Profile
Update Gridscape
Interpolate Preformance Profile
Schedule and Submit Ready Tasks
Yes
More Tasks To Submit?
No

Figure 1: **Navigational cycle on the grid.** The static gridscape and application performance profile are generated offline by a user. The iterations of the navigational cycle can be executed automatically by a navigational tool.

In this paper we focus on the first step of the navigational cycle: obtaining an up-to-date gridscape. For demonstrating the navigational process we use two basic schedulers and the reporting capabilities of NGB to monitor the tasks.

The iterations of the navigational cycle require updating short-lived gridscape information. The initial iteration requires detailed information about machines, number of processors, memory, disk space, directory structure, OSs, availability of software and libraries used by the tasks, and network latency. This detailed static information may be obtained off-line before the grid application has started. For the subsequent iterations of the navigational cycle, less detailed but dynamic information about actual number of available processors, memory size and network bandwidth is necessary.

There are a number of ways the user can obtain the static component of the gridscape including `ldapsearch`, `grid-info-search`, or recently developed *MDS-2* which implements the GRIP protocol [20]. Architecture specific monitoring commands such as `df`, `ps`, `hinv` (IRIX) or `psrinfo` (Solaris™8) can be used if the user has detailed information about grid machines. The gridscape obtained in this way, however, is insufficient for reliable navigation since it may be stale, inaccurate or nonspecific about some services required by the application. For reliable navigation of a grid application, a current gridscape is necessary. The dynamic component of the gridscape can partially be obtained with another set of tools such as *Network Weather Service* (NWS), *MDS-2* or *LSF*, [23] or machine specific performance monitoring tools such as `top` (IRIX) or `cpustat` (Solaris™8).

In this paper we show how to obtain the most recent gridscape by using the NGB [19]. The NGB tasks are individual instances of the NPB, hence, performance, scalability, and communication requirements of any NGB instance can be easily obtained, making NGB a unique tool for measuring the gridscape. Briefly, our approach is as follows. First we use the parameter study benchmark (ED) to validate the gridscape obtained by `grid-info-search`. To do this we run a serial version of ED an all machines in the gridscape and reduce it to the set of functioning machines where appropriate services are installed. Then we run a multithreaded version of ED (OpenMP in Fortran version [4] or multithreaded Java version [5]) to obtain the available number of processors on the grid machines and then we sort the machines according to the actual performance. Next we thread a chain of sequential communicating processes (HC) across the chosen machines to test network connectivity and bandwidth and update bandwidth information in the gridscape. Finally, the gridscape is used in the navigational cycle to assign application tasks to the grid machines.

3 Observing the Gridscape

Logically computational grid can be veiwed at as a set of basic services necessary and sufficient for running applications on the grid, such as *authenticate, create task, communicate*, see [6, p. 37]. On the physical level this logical view is supported by actual computers, storage devices and networks, see Figure 2. The grid application turnaround time depends on how well the physical layer is able to execute assigned tasks and communicate requests/results between tasks. The user's view of the physical layer of the grid, the gridscape, includes such characteristics as the computer performance, number of processors, disk and memory capacities, network latencies and bandwidth.

As a formalization of gridscape in this paper we use an abstract graph representation of the grid, we call this representation a gridscape as well. Nodes of the graph are either machines or networks and arcs are interconnections between machines or between machines and networks, see Figure 3. Each graph node/arc of the gridscape stores observed information about machine/interconnection, such as name, number of processors, processor speed, latency and bandwidth.

The process of observing the grid and recording the information in the gridscape we call mapping. We map the grid in two steps. First, we use Globus `grid-info-search` to obtain basic information about machines and networks which can be used to run the application. Second, we use NGB to obtain current information about the grid such as machines/services status and actual performance and bandwidth. Results of the observations of both steps are recorded in the gridscape in terms of performance, number of processors, memory size, and disc capacity of the grid machines, and latency and bandwidth of communication links between machines, some of which are shown symbolically at Figure 3.

Observation of the gridscape has to be performed in the presence of the varying grid load, which distorts the observation. Depending on the speed of changing of the gridscape we distinguish between volatile and stable grids. In a stable grid during application run the gridscape varies by a factor which does not exceeds a user defined threshold. This threshold depends on the user requirements of the application turnaround time. If the grid is stable then a

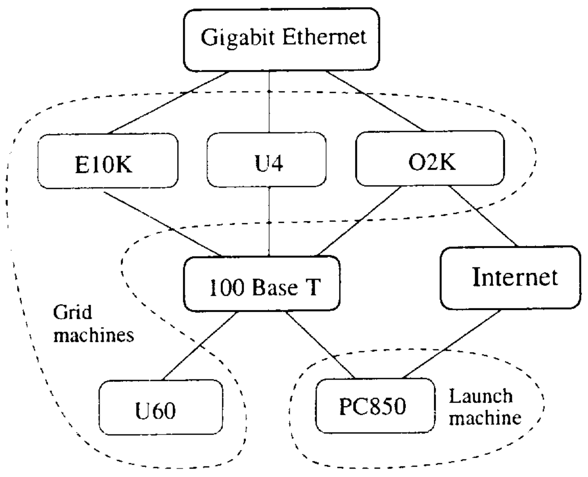


Figure 2: **A physical map of a rudimentary grid.** The network configuration was provided by a network administrator.

gridscape can be obtained once, while in volatile a grid the gridscape must be updated before assignment and launching each task. In Figure 4 we see that if the threshold is 3 then the grid can be considered as stable, otherwise it should be treated as volatile.

4 Navigation on the Grid

With the knowledge of the gridscape, a user can make informed decisions about choosing grid resources and assigning the tasks to minimize the application turnaround time. The gridscape can be used to extrapolate an application performance profile, or model in terms of [26], that is requirements in number of FLOPS and communication volume, to actual time requirements for grid machines and networks. This extrapolated performance in terms of execution time and communication delays then may be used by a scheduling algorithm to schedule the grid application. To

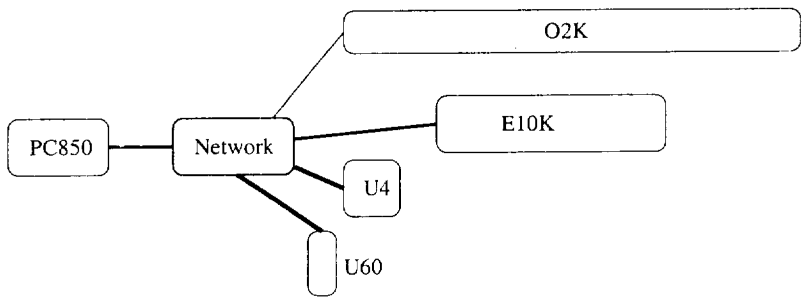


Figure 3: **Gridscape of the rudimentary grid.** The width of the rectangles representing the grid machines are proportional to the number of processors, the height is proportional to the clock rate. Width/length of a network link reflects its bandwidth/latency. The measurement process is described in Section 5.

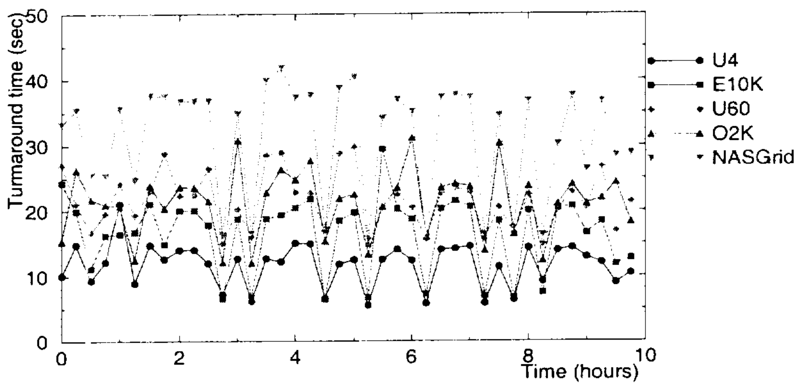


Figure 4: **Monitoring of ED.S (Java).** The benchmark was executed every 15 minutes over a 10 hour period on a rudimentary grid. The turnaround time of ED.S (Fortran) has amplitude smaller by a factor of 5 but exhibits similar volatility.

demonstrate navigation with the gridscape we use two scheduling algorithms: a *greedy scheduler* [18] and a *critical path scheduler* [27, 19]. Results of an experimental navigation are presented in the next section.

As a performance profile of a grid application, basic information about computational demand (say in seconds on a specific machine) for each task and amount of the data sent along each arc can be used [26]. With a gridscape the application performance profile can be extrapolated to all grid resources. The extrapolated performance then can be used to estimate the height of each task. Informally the height of a task is the shortest time from starting the task till completion of the application. The heights of tasks are used by both the critical path and the greedy schedulers described at the end of the section to assign tasks to grid machines giving higher tasks higher priority.

Specifically, a grid application we represent by a data flow graph $G = (V, A)$ which nodes are tasks and arcs are communications between tasks We translate the application performance profile to the weight of nodes and lengths of arcs. For each node we assign weight to be execution time of the appropriate task on the fastest grid machine. For each arc we assign length to be average time of sending data from arc's tail task to arc's head task. Since G is acyclic directed graph we can estimate the heights of the nodes by the algorithm depicted at Figure 5.

Now we briefly describe two scheduling algorithms we use to demonstrate the navigation process. We do not make any conclusions about either the relative efficiency of the schedules obtained with these algorithms or about how close to the optimum these schedules are. Our goal here is to demonstrate that navigation with the gridscape is faster than assignment of tasks to grid machines in a Round Robin fashion.

The Greedy Scheduler assigns for execution the ready node of maximum height. A node is defined to be "ready" if data form all it's predecessors are available. It can be shown that the greedy scheduler gives provably efficient schedules for a number classes of task graphs [18].

The Critical Path Scheduler assigns tasks of a critical path (path of the maximum height) to a fastest grid machine. Then the tasks of the critical path of the graph of remaining tasks are assigned to the second fastest grid machine etc. When the list of free grid machines is exhausted the next critical path is assigned to the first grid machine and so on, cf. [19].

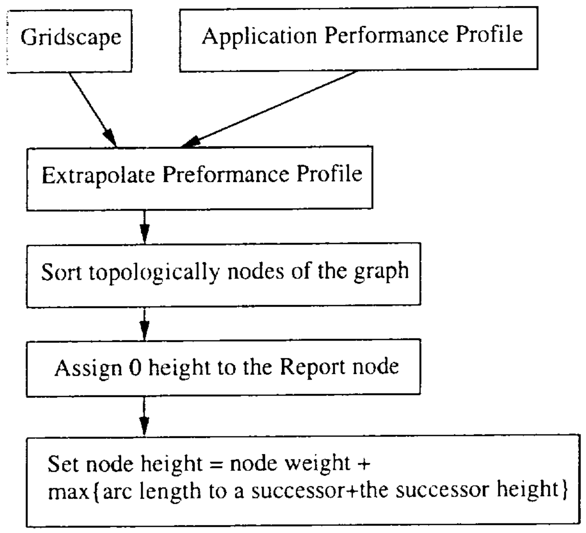

Gridscape
Application Performance Profile
Extrapolate Preformance Profile
Sort topologically nodes of the graph
Assign 0 height to the Report node
Set node height = node weight + max{arc length to a successor+the successor height}

Figure 5: **Algorithm for computing node height.** Informally, height of a node is the shortest time between start of the node task and finish of the whole application.

5 Experimental Navigation

We installed the NGB services on a compact grid comprising 4 machines at NAS: a 32-processor SGI Origin 2000 (O2K), a 16-processor SUN E10000 (E10K), a 4-processor SUN Ultra-4 (U4), and a 2-processor SUN Ultra-60 (U60). The machines were connected by fast, switched Ethernet supplemented with some Gigabit Ethernet connections, see Figure 2. The NGB Pad was launched from on a 850 MHz laptop (PC850) running Windows2000 accessing the grid machines through switched Ethernet and a wireless network connected to an outside router.

First the gridscape of our rudimentary grid was measured by ED.S (Fortran), see top of right panel of Figure 8. Then the bandwidth of the network was estimated via HC.W and it was found that the grid average bandwidth is 2.49 MB/s, see Table 1. Then we used OpenMP version of ED.A (Fortran) to check how many processors are actually available, see Table 2. The the estimations of the number of processors were calculated from ideal speedup curves are shown on Figures 6, 7 and the ED.A (Fortran) turnaround time with requested number of threads equal to the number of machine processors. Then we used the gridscape to schedule VP.W and MB.W both in Java. An application performance profile of these benchmarks was based on the NPB performance model and on the size of data communicated between nodes.

The navigation results of VP.W and MB.W (both in Java) are shown in Figures 8 and 9 respectively. The turnaround time shown at the bottom of the right panel include two sets of runs for each benchmark. Each set includes runs of the benchmarks scheduled with Round Robin, Critical Path and Greedy schedulers. The results imply that the navigation in our rudimentary grid can improve application turnaround time by 20%.

6 Related Work and Conclusions

In the final version of the paper we will discuss the folowing projects: Globus Heart Beat Monitor [12] (part of the Globus [7] metacomputing toolkit), the Network Weather Service [11], the grid simulation projects WARMstones [3],

Table 1: **Measured network bandwidth between the grid machines (MB/s).** The bandwidth was measured with HC.W benchmark with messages in size in range 0.55-1.87 MB. The benchmark uses Java client-server architecture based on HTTP protocol. The number of measurements was in range 2-6 and the standard deviation is shown in parentheses.

From/To	U60	U4	O2K	E10K
U60	2.97 (1.45)	3.48 (0.88)	1.71 (0.13)	3.13 (0.41)
U4	3.64 (0.95)	2.97 (0.70)	2.37 (0.30)	3.24 (0.44)
O2K	0.75 (0.11)	2.20 (0.27)	1.40 (0.08)	2.21 (0.84)
E10K	2.86 (0.46)	2.60 (1.04)	1.67 (0.22)	2.67 (0.34)

Table 2: **The observation of effective number of processors of the grid machines.** The observation was performed with OpenMP version of ED.S and for estimation the number processors we use curve of 7.

Machine	Number of Processors	Clock Rate (MHz)	Observed Speedup	Estimated Number of Processors
U60	2	450	-	-
U4	4	400	-	-
O2K	32	250	7.48	10.47
E10K	16	333	2.03	1.96

MicroGrid [2], and Bricks [10], application steering SCIRun [22], and application scheduling APPLES [25]

The navigation system proposed in this paper allows the user to acquire a map of the grid, which we call a gridscape and to assign tasks of a grid application to the grid machines in order to finish the application in shortest possible time. The advantages of our approach are similar to the advantages of using personal car relative to using a taxicab. It allows the user to keep full control over tasks and to choose the best resources for the tasks based on the current situation on the grid, and as result, to decrease application turnaround time. On the other hand it requires some effort on the user's side. Specifically, it requires an estimation of the application performance profile and the use of off-the-shelf schedulers.

References

[1] D.H. Bailey, J. Barton, T. Lasinski, and H. Simon (Eds.). *The NAS Parallel Benchmarks.* NAS Technical Report RNR-91-002, NASA Ames Research Center, Moffett Field, CA, 1991.

[2] H.J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, A. Chien. *The MicroGrid: a Scientific Tool for Modeling Computational Grids.* Proc. Supercomputing 2000, Dallas, TX, 2000.

[3] S.J. Chapin, *WARMstones: Benchmarking Wide-Area Resource Management Schedulers.* Draft white paper, Syracuse University, http://www.hpdc.syr.edu/~chapin/currentproj.html.

[4] H. Jin, M. Frumkin, J. Yan. *The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance.* NAS Technical Report NAS-99-011, NASA Ames Research Center, Moffett Field, CA, 1999.

[5] M. Frumkin, M. Schultz. H. Jin, J. Yan. *Implementation of NAS Parallel Benchmarks in Java.* Presented at a Poster session at ACM 2000 Java Grande Conference, 2000.

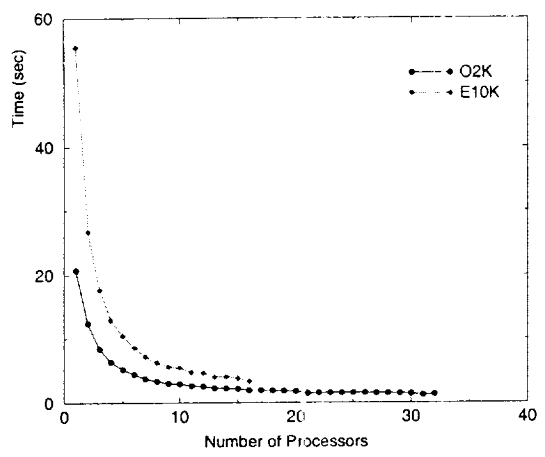


Figure 6: **The ideal time 10 iterations of SP.A (sec) (Fortran, OpenMP).** The measurements were obtained with benchmarks runing under PBS, c.f. [4]. Note that in spite of that O2K has 25% lower clock rate than E10K it executes the same benchmarks almost 2 times faster.

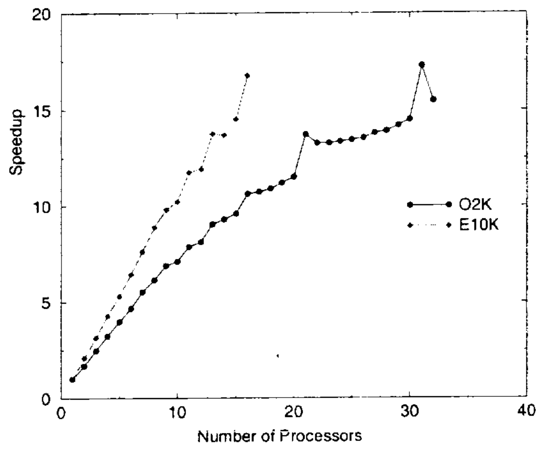


Figure 7: **The ideal speedup of SP.A (Fortran, OpenMP).** Due to faster serial code efficiency of processors are lower on O2K.

[6] *The Grid. Blueprint for a New Computing Infrastructure.* I. Foster, C. Kesselman, Eds., Morgan Kaufmann Publishers Inc., San Francisco, CA, 1999.

[7] I. Foster, C. Kesselman. *Globus: A Metacomputing Infrastructure Toolkit.* Int. J. Supercomputer Applications, 11(2):115-128, 1997, http://www.globus.org.

[8] C.S. Horstmann, G. Cornell. *Core Java 2, Volume 2: Advanced Features.* 4th edition, Prentice Hall, 1999, see also http://java.sun.com/j2se/1.3/docs.

[9] M. Livny, J. Basney, R. Raman, T. Tannenbaum. *Mechanisms for High Throughput Computing.* SPEEDUP Journal, Vol. 11(1), 1997, http://www.cs.wisc.edu/condor/.

[10] A. Takefusa, S. Matsuoka, H. Nakada, K. Aida, U. Nagashima. *Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms.* Proc. High-Performance and Distributed Computing 8, pp. 97-104, 1999.

[11] R. Wolski, N.T. Spring, J. Hayes. *The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing.* J. Future Generation Computing Systems, 1999, also UCSD Technical Report Number TR-CS98-599, 1998, http://nws.npaci.edu/NWS/.

[12] *Globus Heartbeat Monitor.* http://www.globus.org/hbm/.

[13] *NASA Information Power Grid.* http://www.nas.nasa.gov/IPG.

[14] *Codine 5.2 Manual, Revision A.* Sun Microsystems, Inc., Palo Alto, CA, September 2000, http://www.sun.com/gridware.

[15] *Legion 1.6, Developer Manual.* The Legion Research Group, Dept. Computer Science, U. Virginia, Charlottesville, VA, 1999, http://legion.virginia.edu.

[16] R.D. Blumofe, P.A. Lisiecki. *Adaptive and Reliable Parallel Computing on Networks of Workstations.* Proceedings pf USENIX 1997 Annual Technical Conference on UNIX and Advanced Computing Systems, Anaheim, CA, Jan. 6-10, 1997, 15 pp.

[17] D. Feitelson, A. Weil. *Utilization and Predictability in Scheduling the IBM SP2 with Backfilling.* 12th International Parallel Processing Symposium and 9th Symposium on Parallel and Distributed Processing, 1998, p. 542-546.

[18] G.E. Blelloch, P.B. Gibbons, Y. Matias. *Provably Efficient Scheduling for Languages with Fine-Grained Parallelism.* Journal of ACM, Vol. 46, No. 2, March 1999, pp 281-321.

[19] M. Frumkin, Rob F. Van der Wijngaart. *NAS Grid Benchmarks: A Tool for Grid Space Exploration.* Submitted to Journal of Cluster Computing. Preliminary version published in HPDC-10, 7-9 August, 2001, San-Francisco, CA, pp. 315-322.

[20] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. *Grid Information Services for Distributed Resource Sharing.* HPDC-10, 7-9 August, 2001, San-Francisco, CA, pp. 181-194.

[21] M. Miller, C.D. Hansen, S.G. Parker, C.R. Johnson. *Simulation Steering with SCIRun in a Distributed Memory Environment.* http://www.cs.utah.edu/pubs/scirun_pubs.html, HPDC-7.

[22] C.R. Johnson, S.G. Parker, D. Weinstein. *Large Scale Computational Science Applications using the SCIRun Problem Solving Environment.* Supercomputing 2000.

[23] Platform Computing. *Building Production Grids with Platform Computing.* http://www.platform.com/industry/whitepapers.

[24] D. Gannon at al. *Programming the Grid: Distributed Software Components, P2P and Grid Web services for Scientific Applications.* http://www.extreme.indiana.edu/ gannon.

[25] F. Berman, R. Wolski, S. Figueira, J. Schopf, S. Shao. *Application Level Scheduling on Distributed Heterogeneous Networks.* In Proceedings of Supercomputing 1996, 1996.

[26] F. Berman. *High-Performance Schedulers.* In: The Grid. Blueprint for a New Computing Infrastructure. I. Foster, C. Kesselman, Eds., Morgan Kaufmann Publishers Inc., San Francisco, CA, 1999.

[27] J.K. Hollingsworth. *Critical Path Profiling of Message Passing and Shared-memory Programs.* IEEE Transactions on Parallel and Distributed Systems, 1998, pp. 1029-1040.

[28] D. Royo, N. Kapadia, J. Fortes, L. Diaz de Cerio. *Active Yellow Pages: A Pipelined Resource Management Architecture for Wide-Area Network Computing.* HPDC-10, 7-9 August, 2001, San-Francisco, CA, pp. 147-157.

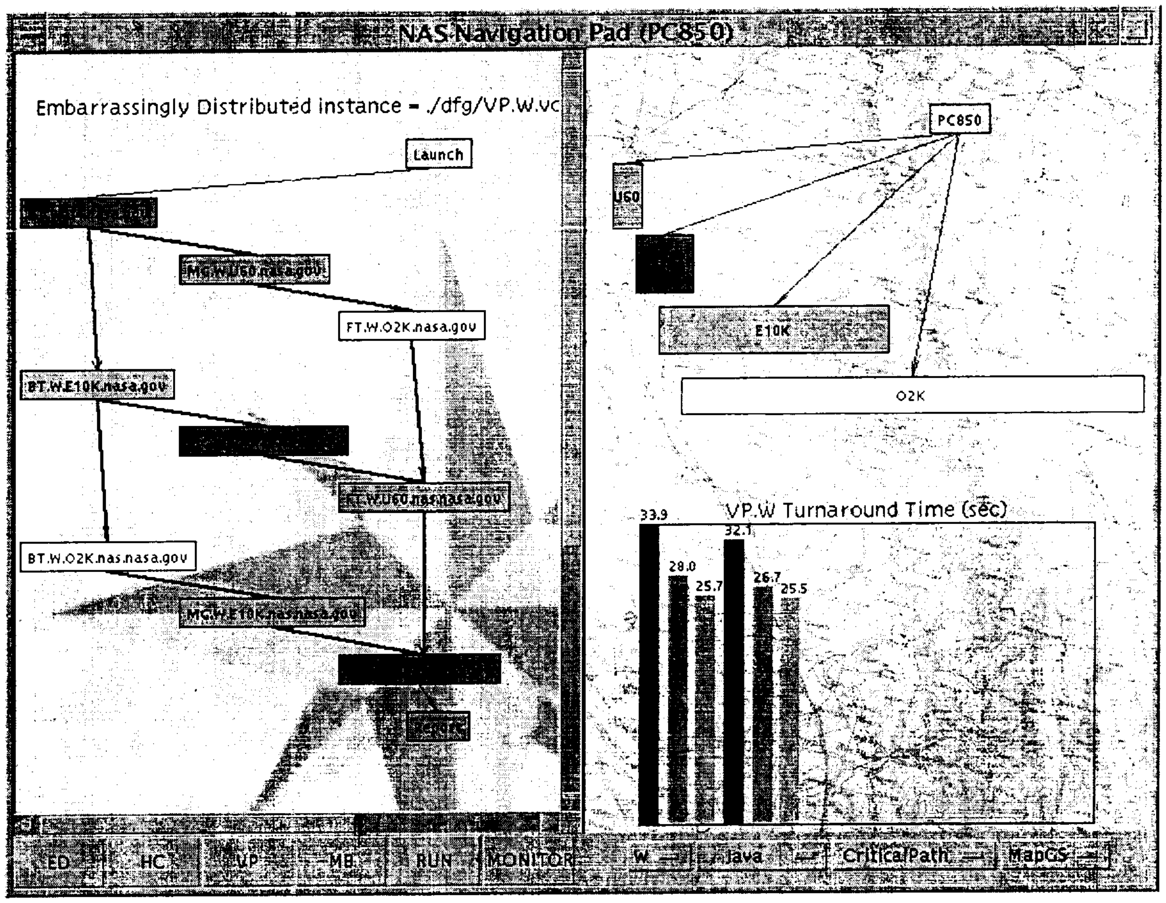


Figure 8: **The turnaround time of VP.W (Java).** The benchmark was scheduled with the Round Robin task assignment (shown), with the Critical Path and Greedy schedulers. The two sets of vertical bars on the right panel show turnaround time for two sets of runs of VP.W scheduled with these three schedulers. For example, the two Round Robin task assignment runs took 33.9 and 32.1 seconds.

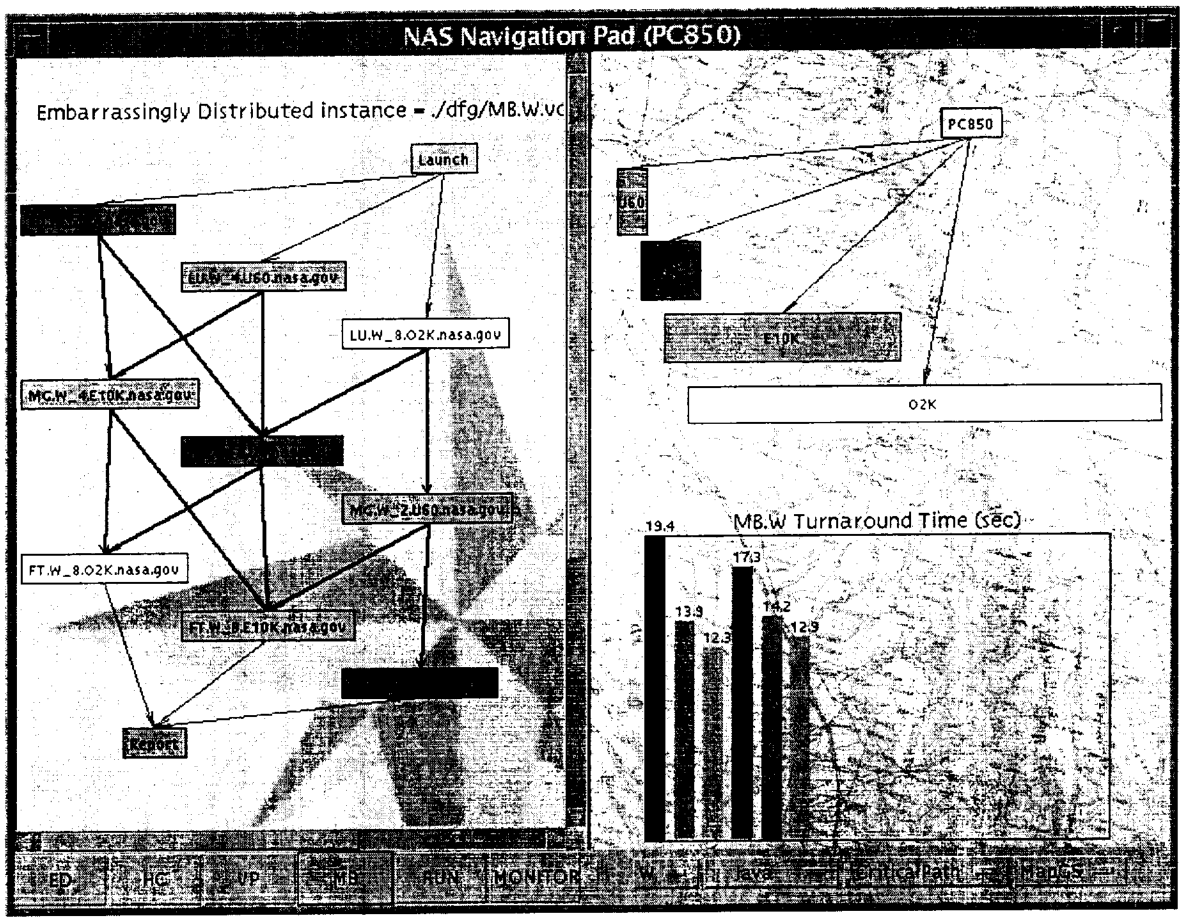


Figure 9: **Navigation results of MB.W (Java).**