A Look at Some Ideas and Experiments

Jack Dongarra
University of Tennessee
and
Oak Ridge National Laboratory

Orientation

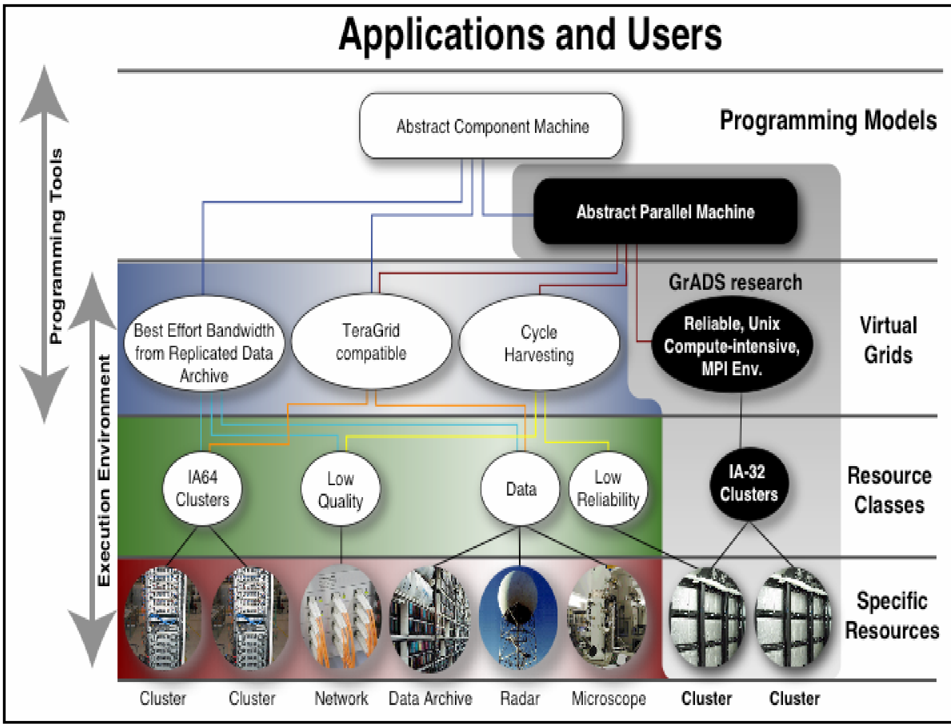
- The design of smart numerical libraries; libraries that can use the “best” available resources, analyze the data, and search the space of solution strategies to make optimal choices
- The development of “agent-based” methods for solving large numerical problems on both local and distant grids
- Development of a prototype framework based on standard components for building and executing composite applications

The Grid: Abstraction

- Semantically: the grid is nothing but abstraction
  - Resource abstraction
    - Physical resources can be assigned to virtual resource needs (matched by properties)
    - Grid provides a mapping between virtual and physical resources
  - User abstraction
    - Grid provides a temporal mapping between virtual and physical users

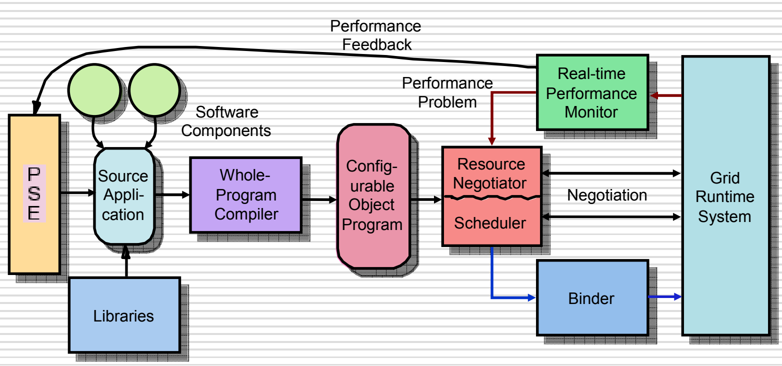
With The Grid…

- What performance are we evaluating?
  - Algorithms
  - Software
  - Systems
- What are we interested in?
  - Fastest time to solution?
  - Best resource utilization?
  - Lowest "cost" to solution?
  - Reliability of solution?
  - …



NSF/NGS
GrADS - GrADSoft Architecture

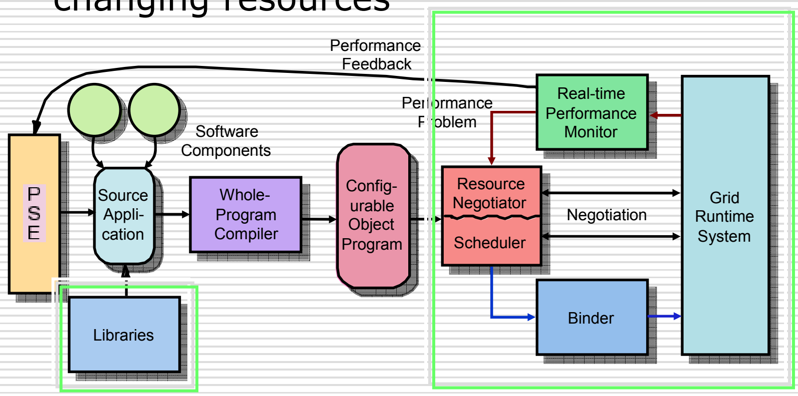
- Goal: reliable performance on dynamically changing resources



PIs: Ken Kennedy, Fran Berman, Andrew Chein, Keith Cooper, JD, Ian Foster, Lennart Johnsson, Dan Reed, Carl Kesselman, John Mellor-Crummey, Linda Torczon & Rich Wolski

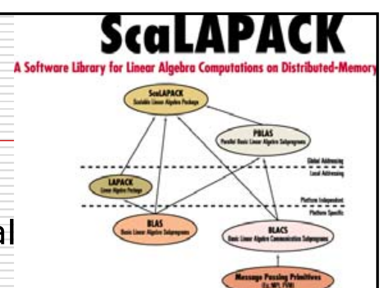
NSF/NGS
GrADS - GrADSoft Architecture

- Goal: reliable performance on dynamically changing resources



PIs: Ken Kennedy, Fran Berman, Andrew Chein, Keith Cooper, JD, Ian Foster, Lennart Johnsson, Dan Reed, Carl Kesselman, John Mellor-Crummey, Linda Torczon & Rich Wolski

ScaLAPACK



- ScaLAPACK is a portable distributed memory numerical library
- Complete numerical library for dense matrix computations
- Designed for distributed parallel computing (MPP & Clusters) using MPI
- One of the first math software packages to do this
- Numerical software that will work on a heterogeneous platform
- Funding from DOE, NSF, and DARPA
- In use today by IBM, HP-Convex, Fujitsu, NEC, Sun, SGI, Cray, NAG, IMSL, …
  - Tailor performance & provide support

To Use ScaLAPACK a User Must:

- Download the package and auxiliary packages (like PBLAS, BLAS, BLACS, & MPI) to the machines.
- Write a SPMD program which
  - Sets up the logical 2-D process grid
  - Places the data on the logical process grid
  - Calls the numerical library routine in a SPMD fashion
  - Collects the solution after the library routine finishes
- The user must allocate the processors and decide the number of processes the application will run on
- The user must start the application
  - "mpirun –np *N* user_app"
    - Note: the number of processors is fixed by the user before the run, if problem size changes dynamically …
- Upon completion, return the processors to the pool of resources

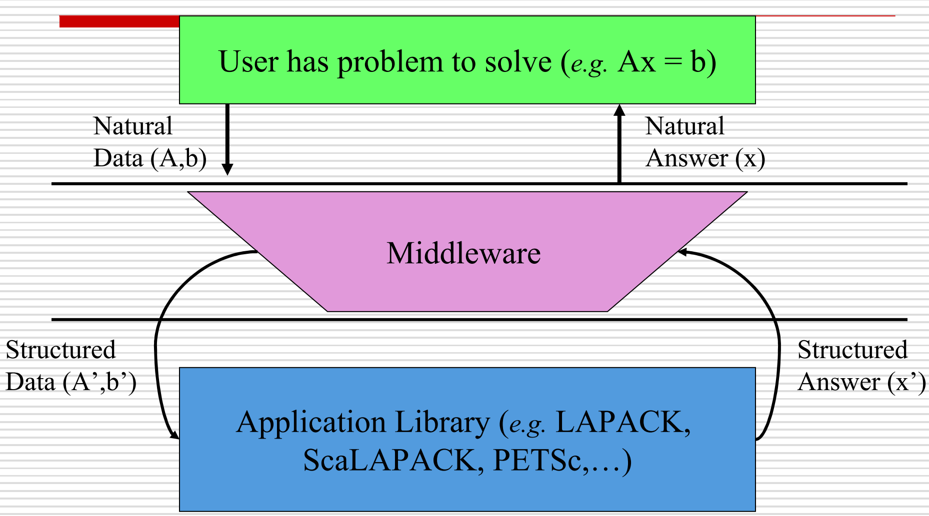
ScaLAPACK Grid Enabled

- Implement a version of a ScaLAPACK library routine that runs on the Grid.
  - Make use of resources at the user's disposal
  - Provide the best time to solution
  - Proceed without the user's involvement
- Make as few changes as possible to the numerical software.
- Assumption is that the user is already "Grid enabled" and runs a program that contacts the execution environment to determine where the execution should take place.
- Best time to solution

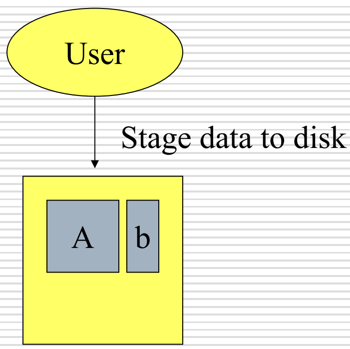
GrADS Numerical Library

- Want to relieve the user of some of the tasks
- Make decisions on which machines to use based on the user's problem and the state of the system
  - Determinate machines that can be used
  - Optimize for the best time to solution
  - Distribute the data on the processors and collections of results
  - Start the SPMD library routine on all the platforms
  - Check to see if the computation is proceeding as planned
    - If not perhaps migrate application

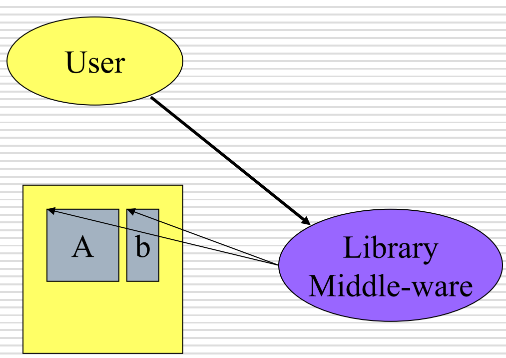
Big Picture...



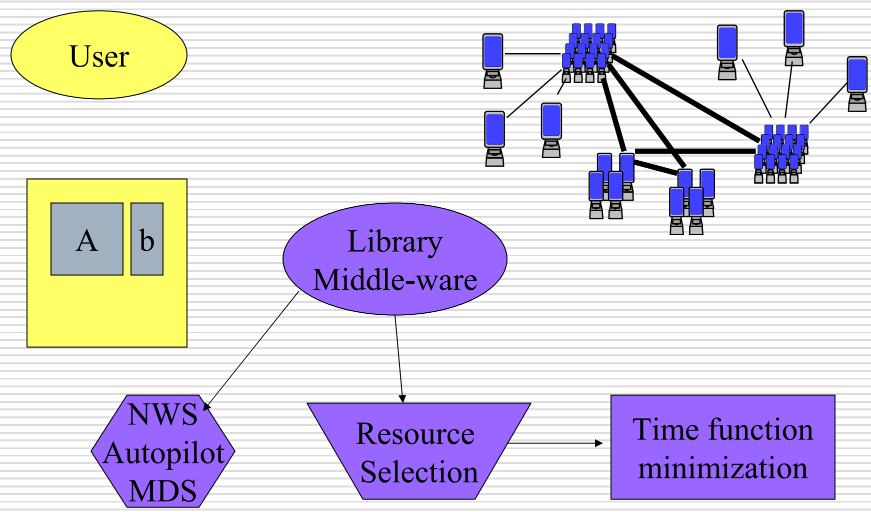
Numerical Libraries for Grids



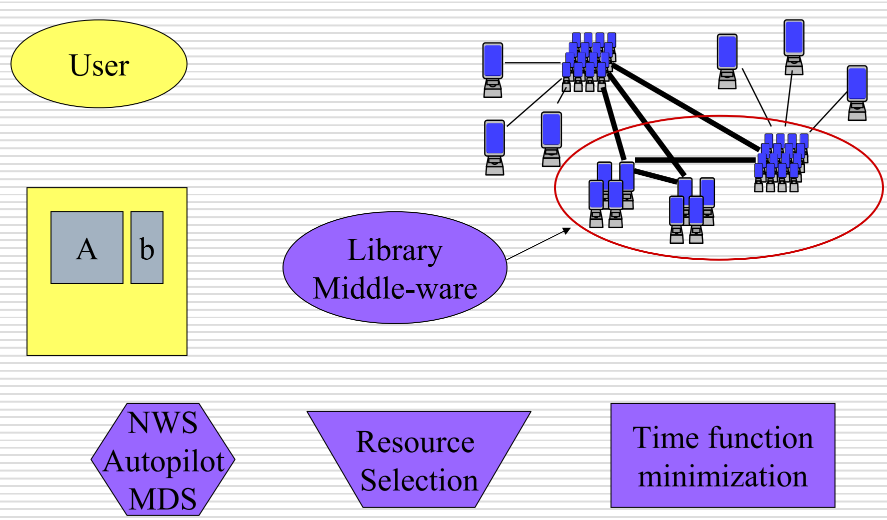
Numerical Libraries for Grids



Numerical Libraries for Grids

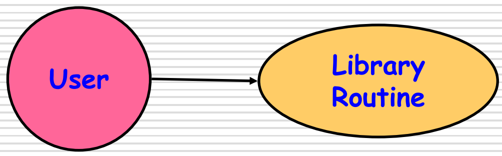


Numerical Libraries for Grids



Uses Grid infrastructure, i.e.Globus/NWS.

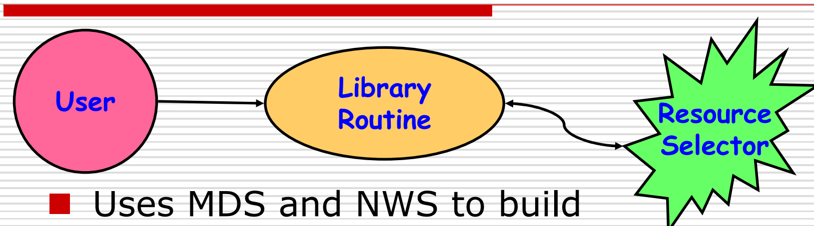
GrADS Library Sequence



- Has "crafted code" to make things work correctly and together.

Assumptions:
Autopilot Manager has been started and
Globus is there.

Resource Selector



- Uses MDS and NWS to build an array of values
  - 2 matrices (bw,lat) 2 arrays (cpu, memory available)
  - Matrix information is clique based
- On return from RS, Crafted Code filters information to use only machines that have the necessary software and are really eligible to be used.

Resource Selector Input

- Clique based
  - 2 @ UT, UCSD, UIUC
    - Part of the MacroGrid
  - Full at the cluster level and the connections (clique leaders)
  - Bandwidth and Latency information looks like this.
  - Linear arrays for CPU and Memory
- Matrix of values are filled out to generate a complete, dense, matrix of values.
- At this point have a workable coarse grid.
  - Know what is available, the connections, and the power of the machines

x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x	x	x	x
x	x x x x x x x x x x x x x x x x	x	x
x	x	x x x x x x x x x x x x x x x x x x x x x x x x x	x
x	x	x	x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x

Uses NWS to collect information

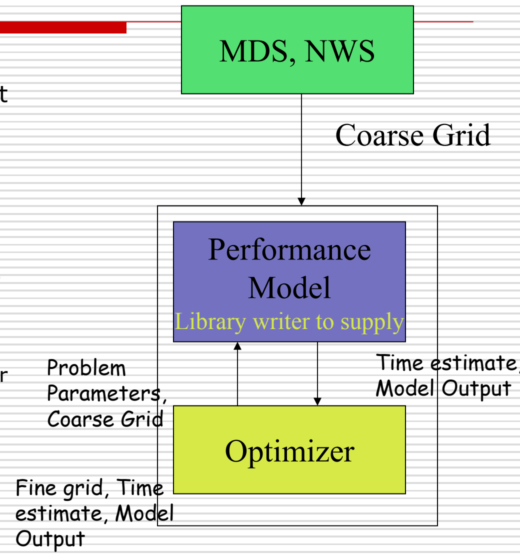
ScaLAPACK Performance Model

$$T(n,p) = C_f t_f + C_v t_v + C_m t_m$$

- $C_f = \frac{2n^3}{3p}$ Total number of floating-point operations per processor
- $C_v = (3 + \frac{1}{4}\log_2 p)\frac{n^2}{\sqrt{p}}$ Total number of data items communicated per processor
- $C_m = n(6 + \log_2 p)$ Total number of messages
- t_f Time per floating point operation
- t_v Time per data item communicated
- t_m Time per message

Resource Selector/Performance Modeler

- Refines the course grid by determining the process set that will provide the best time to solution.
- This is based on dynamic information from the grid and the routines performance model.
- The PM does a simulation of the actual application using the information from the RS.
  - It literally runs the program without doing the computation or data movement.
- There is no backtracking in the Optimizer.
  - This is an area for enhancement and experimentation.
- Simulated annealing used as well



Performance Model Validation

	Opus14	Opus13	Opus16	Opus15	Torc4	Torc6	Torc7
mem(MB)	215	214	227	215	233	479	479
speed	270	270	270	270	330	330	330
load	1	0.99	1	0.99	1	1.04	0.87

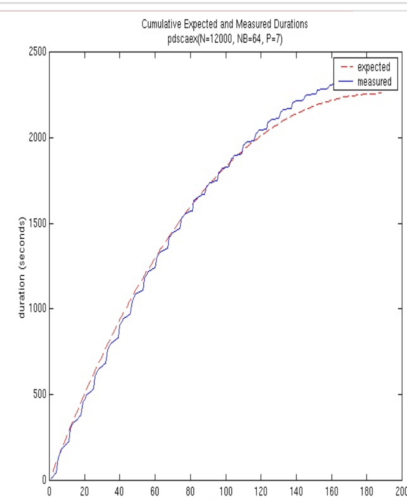
Speed = performance of DGEMM (ATLAS)

Latency	Opus14	Opus13	Opus16	Opus15	Torc4	Torc6	Torc7
Opus14	-1	0.24	0.29	0.26	83.78	83.78	83.78
Opus13	0.24	-1	0.24	0.23	83.78	83.78	83.78
Opus16	0.29	0.24	-1	0.23	83.78	83.78	83.78
Opus15	0.26	0.23	0.23	-1	83.78	83.78	83.78
Torc4	83.78	83.78	83.78	83.78	-1	0.31	0.31
Torc6	83.78	83.78	83.78	83.78	0.31	-1	0.31
Torc7	83.78	83.78	83.78	83.78	0.31	0.31	-1

Latency in msec

Bandwidth	Opus14	Opus13	Opus16	Opus15	Torc4	Torc6	Torc7
Opus14	-1	248.83	247.31	246.38	2.83	2.83	2.83
Opus13	248.83	-1	244.54	240.94	2.83	2.83	2.83
Opus16	247.31	244.54	-1	247.54	2.83	2.83	2.83
Opus15	246.38	240.94	247.54	-1	2.83	2.83	2.83
Torc4	2.83	2.83	2.83	2.83	-1	81.96	56.47
Torc6	2.83	2.83	2.83	2.83	81.96	-1	50.9
Torc7	2.83	2.83	2.83	2.83	56.47	50.9	-1

Bandwidth in Mb/s



This is for a refined grid

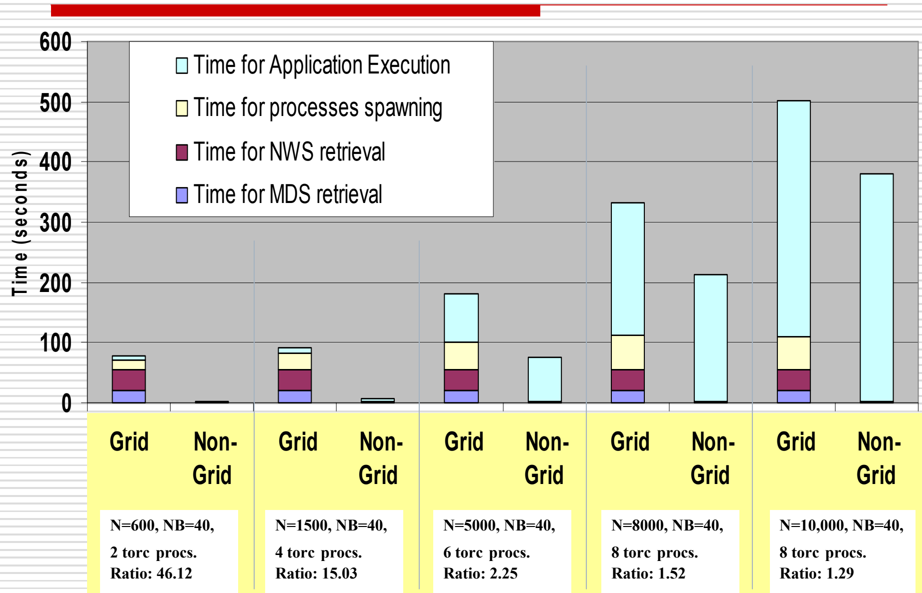
Experimental Hardware / Software Grid

MacroGrid Testbed	TORC	CYPHER	OPUS
Type	Cluster 8 Dual Pentium III	Cluster 16 Dual Pentium III	Cluster 8 Pentium II
OS	Red Hat Linux 2.2.15 SMP	Debian Linux 2.2.17 SMP	Red Hat Linux 2.2.16
Memory	512 MB	512 MB	128 or 256 MB
CPU speed	550 MHz	500 MHz	265 – 448 MHz
Network	Fast Ethernet (100 Mbit/s) (3Com 3C905B) and switch (BayStack 350T) with 16 ports	Gigabit Ethernet (SK-9843) and switch (Foundry FastIron II) with 24 ports	Myrinet (LANai 4.3) with 16 ports each

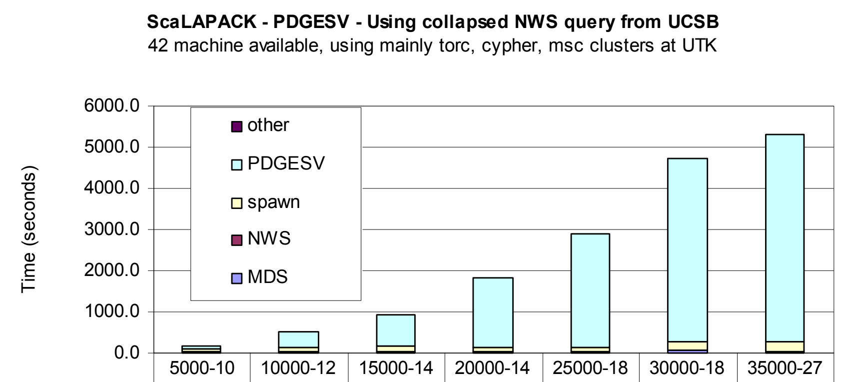
- Globus version 1.1.3
- Autopilot version 2.3
- NWS version 2.0.pre2
- MPICH-G version 1.1.2
- ScaLAPACK version 1.6
- ATLAS/BLAS version 3.0.2
- BLACS version 1.1
- PAPI version 1.1.5
- GrADS' "Crafted code"

Independent components being put together and interacting

Grid ScaLAPACK vs Non-Grid ScaLAPACK, Dedicated Torc machines



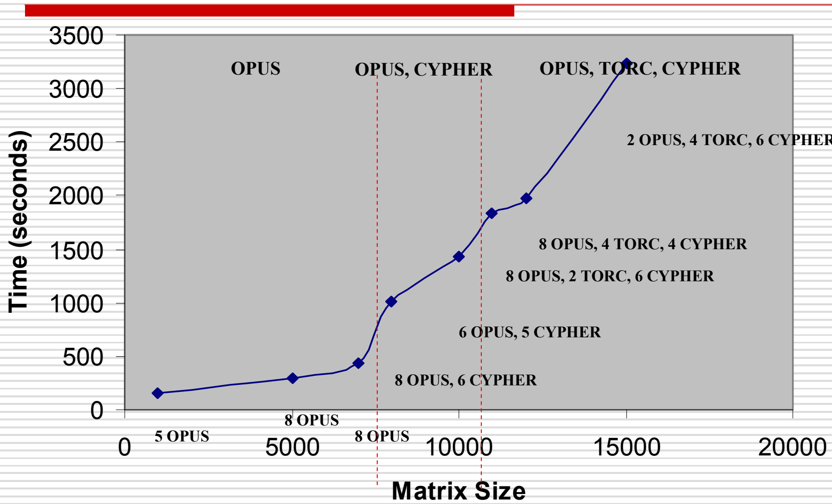
PDGESV Time Breakdown



	5000-10	10000-12	15000-14	20000-14	25000-18	30000-18	35000-27
other	4.7	5.3	1.0	2.3	7.6	1.1	4.7
PDGESV	58.7	394.7	749.4	1686.7	2747.1	4472.7	5020.4
spawn	92.2	105.9	154.1	124.7	135.6	181.0	264.5
NWS	5.5	7.4	12.3	12.0	4.2	10.2	8.5
MDS	13.0	11.0	10.0	11.0	14.0	73.0	12.0

Matrix Size - Nproc

ScaLAPACK across 3 Clusters

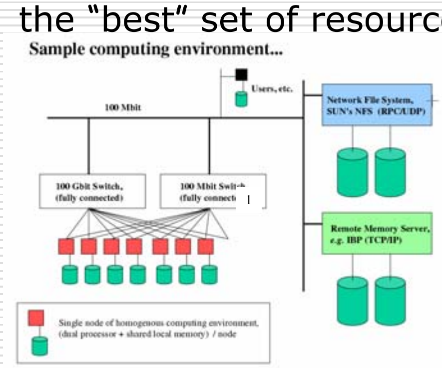


Largest Problem Solved

- Matrix of size 35,000
  - 7.2 GB for the data
  - 32 processors to choose from UIUC and UT
    - Not all machines have 512 MBs, some little as 128 MBs
  - PM chose 27 machines in 3 clusters from UT
  - Computation took 87 minutes
    - 5.5 Gflop/s total
    - 205 Mflop/s per processor
    - Rule of thumb for ScaLAPACK is about 50% of theoretical peak
    - Processors are 500 MHz or 500 Mflop/s peak
    - For this grid computation 6% less than ScaLAPACK

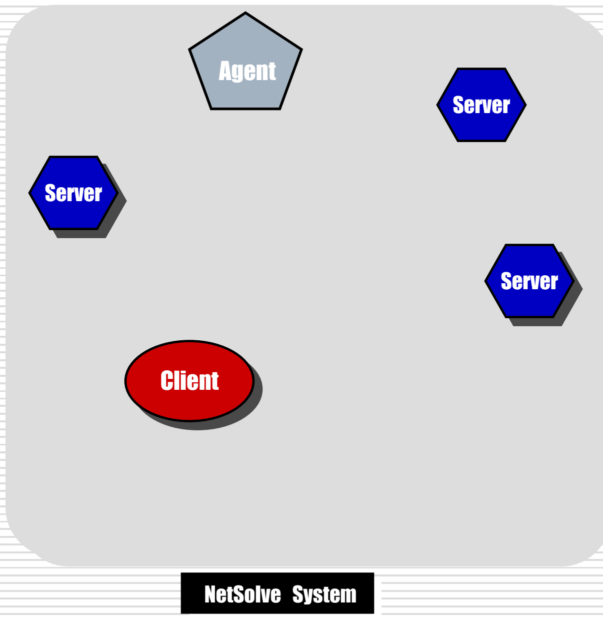
LAPACK For Clusters

- Developing middleware which couples cluster system information with the specifics of a user problem to launch cluster based applications on the “best” set of resource available.

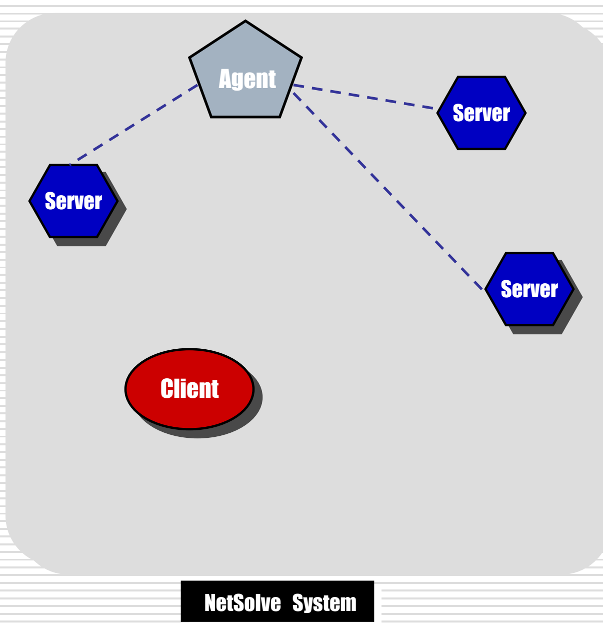


- Using ScaLAPACK as the prototype software

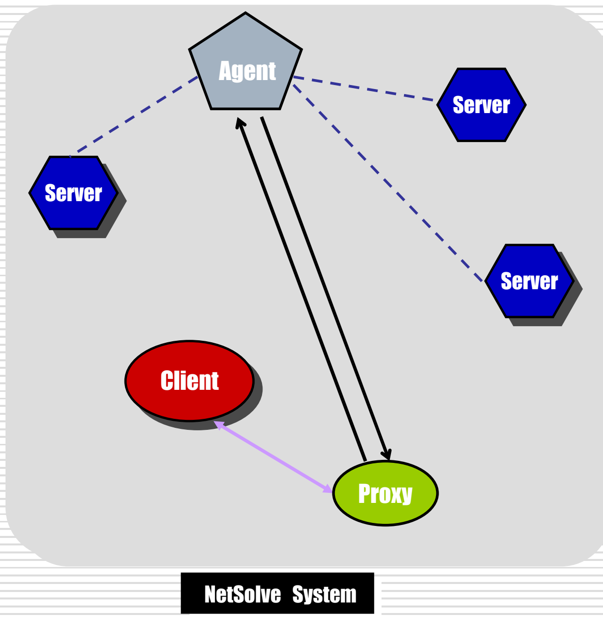
NetSolve Timing



NetSolve Timing

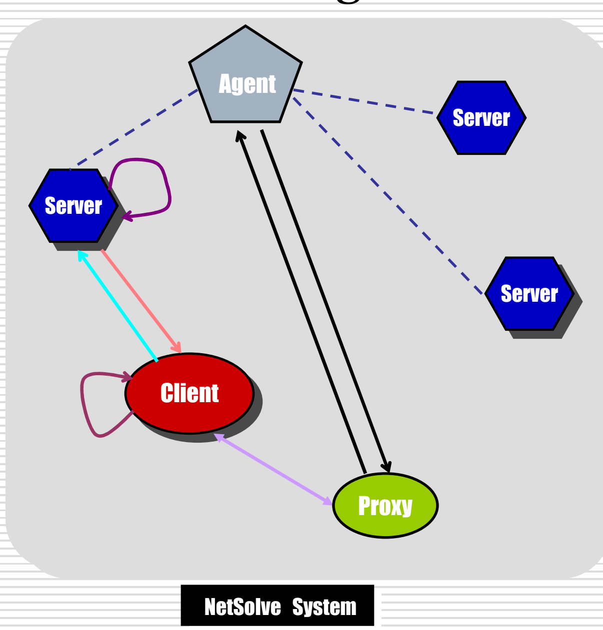


NetSolve Timing



Start proxy
Initialize objects
Receive server info
Send input to server
Job complete
Download output

NetSolve Timing



Start proxy
Initialize objects
Receive server info
Send input to server
Job complete
Download output

Grid Data Movement

- Experiments
  - Comparing BW and times for moving data over a grid/WAN/LAN using 4 protocols
  - NetSolve, scp, Globus gridftp using 1 stream, gridftp using 16 streams
- Data
  - Floats [0-1) or ints (for NetSolve) in binary format, can be compressed about 10%
- Grid Sites
  - UTK msc (dual 933 MHz PIII, Linux)
  - UTK torc (dual 550MHz PIII, Linux)
  - UCSD mystere (1733 MHz Athlon XP, Linux)
  - UH mckinley (900 MHz Itanium IA64, Linux)

Details on the Protocols

- NetSolve
  - Uses untuned TCP sockets to move data
- Scp
  - Encrypts data, and compresses by default (gzip –6)
- Globus-url-copy
  - Provides data authentication, GSI security, tcpip buffer and socket tuning, no compression/encryption
- Globus-16
  - Uses 16 parallel streams of globus-url-copy

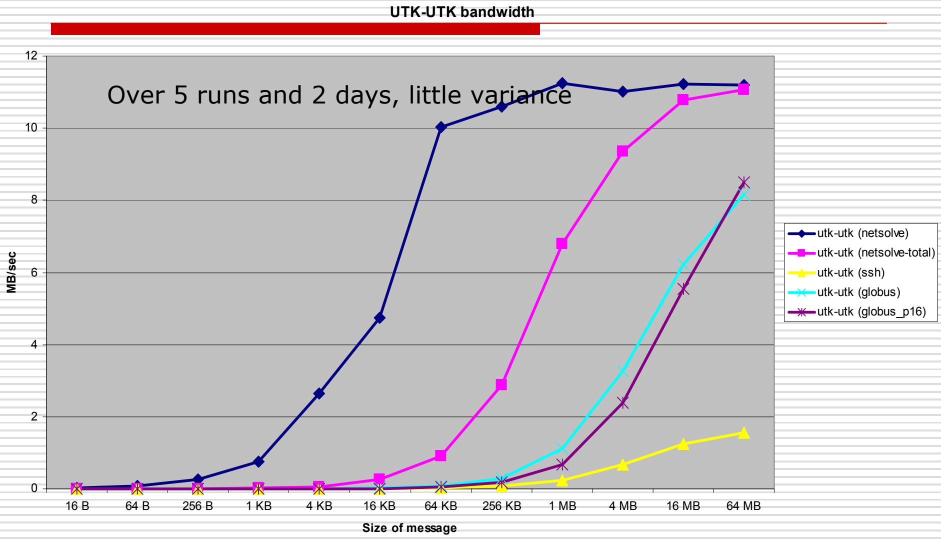
Transfer maximums

- Measured using iperf w defaults
- UTK (msc) –UTK (torc)
  - BW to 89.4Mb/sec = 11 MB/sec
  - BW back 93.6 Mb/sec
- UTK (msc) –UCSD (mystere)
  - BW to 7.96Mb/sec = 1MB/sec
  - BW back 8.00 Mb/sec
- UTK – UH (asymmetric)
  - BW to 9.4 Mb/sec = 1.2 MB/sec
  - BW back 16.4 Mb/sec = 2.0 MB/sec
- Globus-16 stream may exceed iperf 1-stream
  - UTK-UH BW back using 16 parallel threads 59 Mb/sec = 7.3 MB/sec

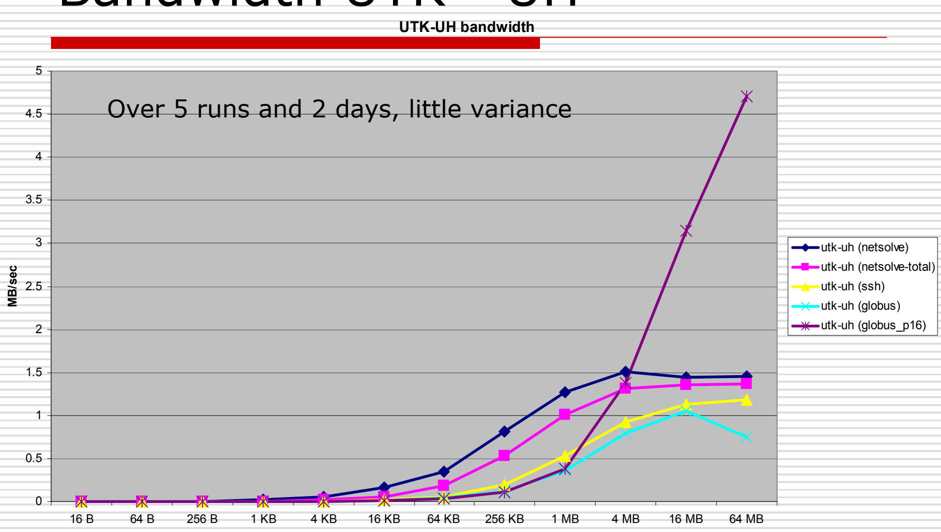
Details on the Protocols

- NetSolve
  - Only includes time to move data, no other overhead is included
- NetSolve-total
  - Includes all the parts of the NetSolve process, including overhead of contacting agent, etc
- Scp
  - Encrypts data, and compresses by default (gzip –6)
- Globus-url-copy
  - Provides data authentication, GSI security, tcpip buffer and socket tuning, no compression/encryption
- Globus-16
  - Uses 16 parallel streams of globus-url-copy

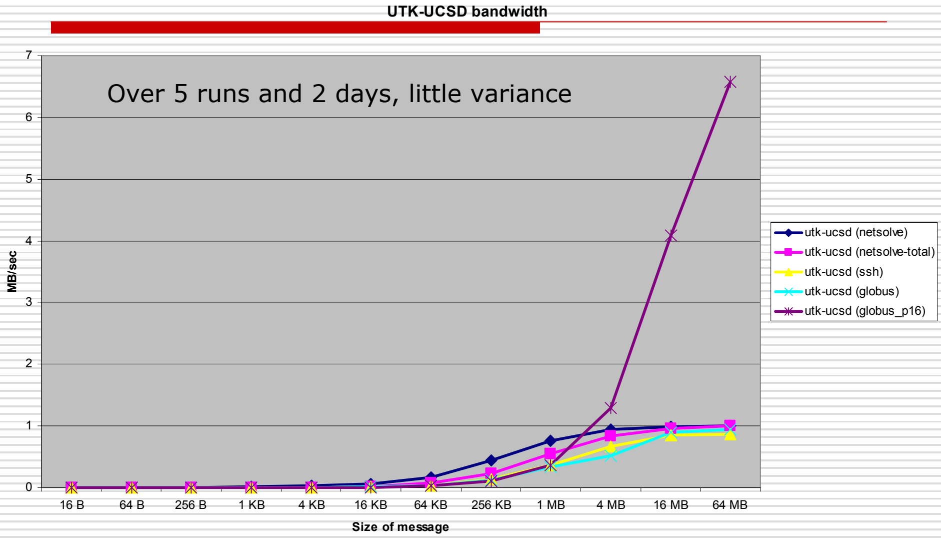
Bandwidth UTK – UTK



Bandwidth UTK - UH



Bandwidth UTK - UCSD



Discussion

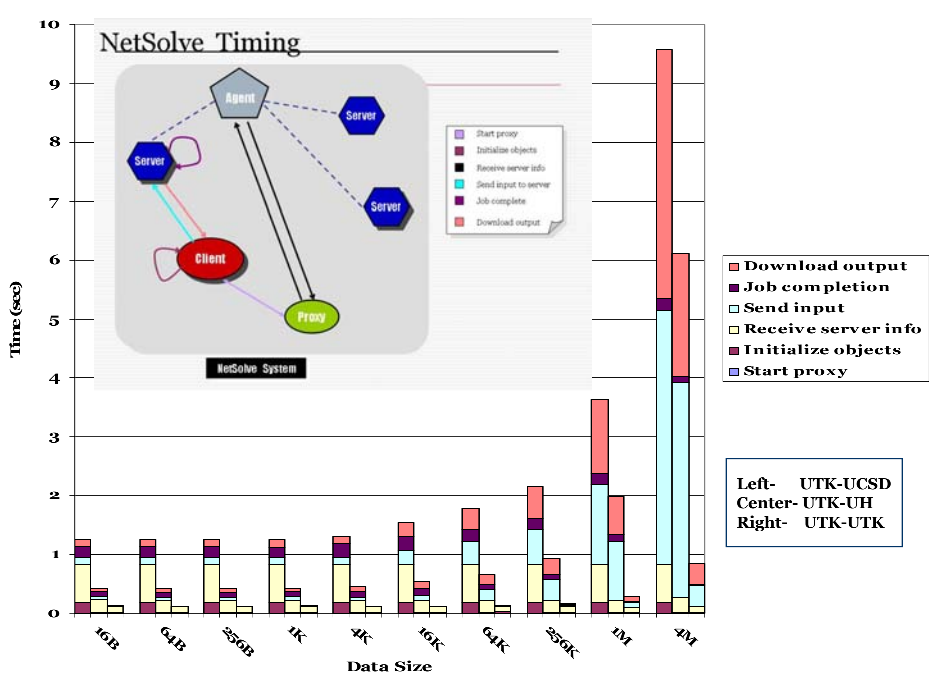
- NetSolve is good for all tested message sizes
  - This is unexpected, since NetSolve does no TCP tuning/etc to improve
  - Linux 2.4 has auto-tuning built into the kernel, which may explain the excellent results
  - We may get different results on Solaris, etc, but we could not compare with globus data transfer protocols, because GrADS does not have any other machine architectures currently
  - Multi-stream globus-16 can do better over WAN

Details on Timing

- Timing for NetSolve
  - Source code was instrumented and precise timings for each operation were obtained
- Timing for globus and scp
  - Perl script was used with a HiRes timer

```
$start = time()
open(CMD, "globus-url-copy file:/file
    gsiftp://remotesite/file" |)
while (CMD) {}
close(CMD)
$end = time()
return $end -$start
```

  - Same for scp command



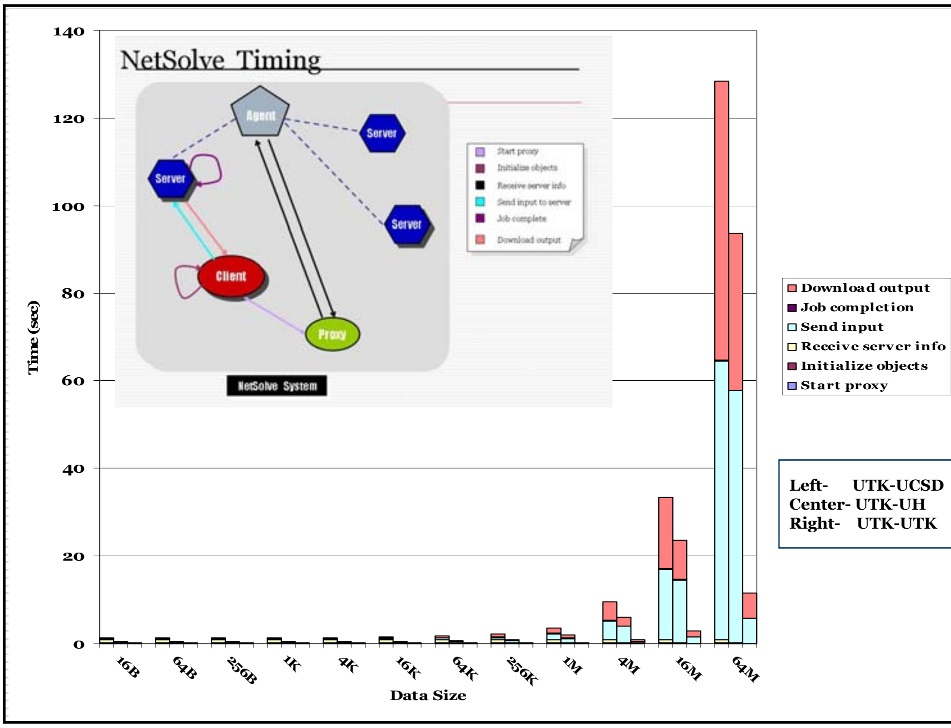
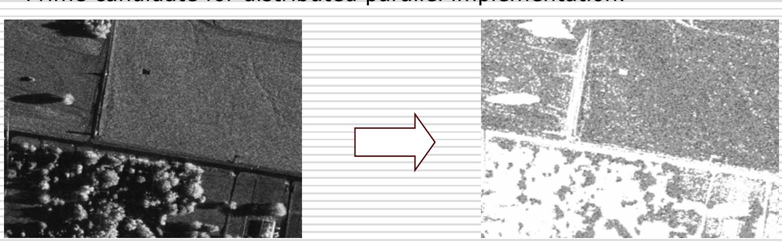


Image Processing

- Image Segmentation
- Goal is to classify pixels into predefined classes which represent different features of the image. (e.g. Trees, Shadows, Roads, etc.)
- CPU intensive application
- Computations for each pixel class are independent of other classes.
- Prime candidate for distributed parallel implementation.



Parallel Image Segmentation

- MPI
- One node per pixel class
- Drastically reduces computation time since the pixel class computations are independent of each other.
- Final image is produced at master node by combining results from each worker node.

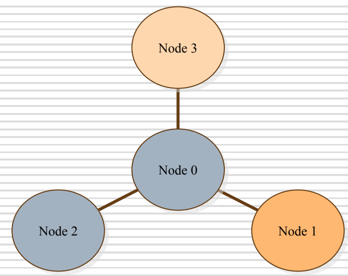


Image Segmentation as Remote Job

- **NetSolve Service**
- MPI code is wrapped as a NetSolve service.
- Clients pass input image and pixel class statistics to service.
- Service launches MPI code to produce the output image.
- Output image returned to the client as result.

- **Globus Job**
- Client transfers input image to remote site (e.g. globus-url-copy file:/inputfile.png https://server/inputfile.png).
- Job and input parameters submitted to remote site using globus-job-run.
- Resulting image is retrieved by user with globus-url-copy.

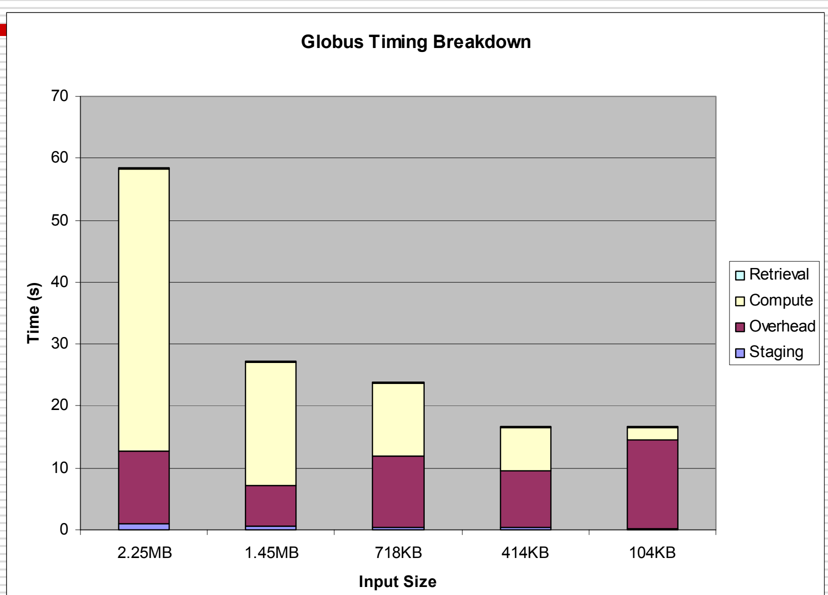
Remote Processing Benchmarks

- Input data staging
- Job submission / Service request overhead
- Job compute time
- Result data retrieval

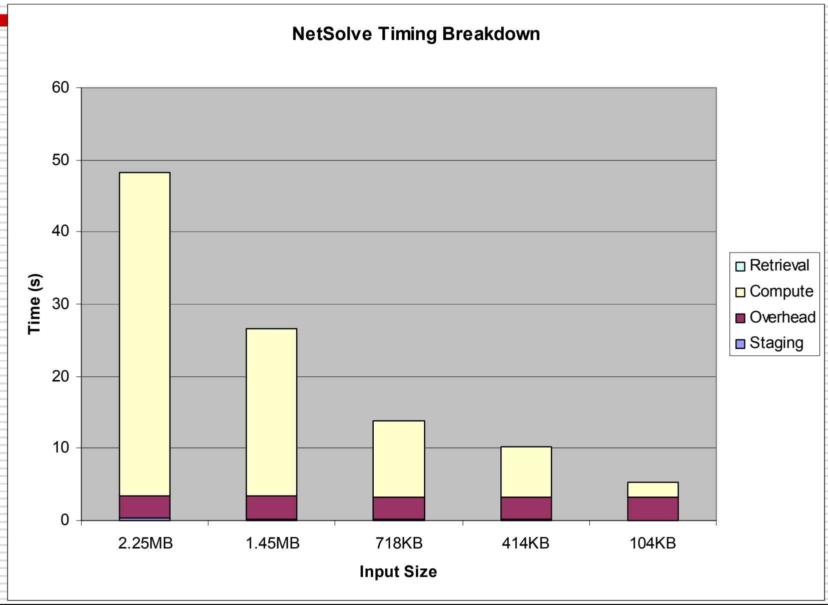
- The image processing code is run remotely using several different sizes for the input image.
- Computation parameters kept constant.
- Each image size is run 200 times on both NetSolve and Globus

- All remote jobs are run within the Torc Production cluster at UTK.
- Input data and result data are already compressed in Portable Network Graphic format (PNG), so no additional compression is performed during data transfer operations.

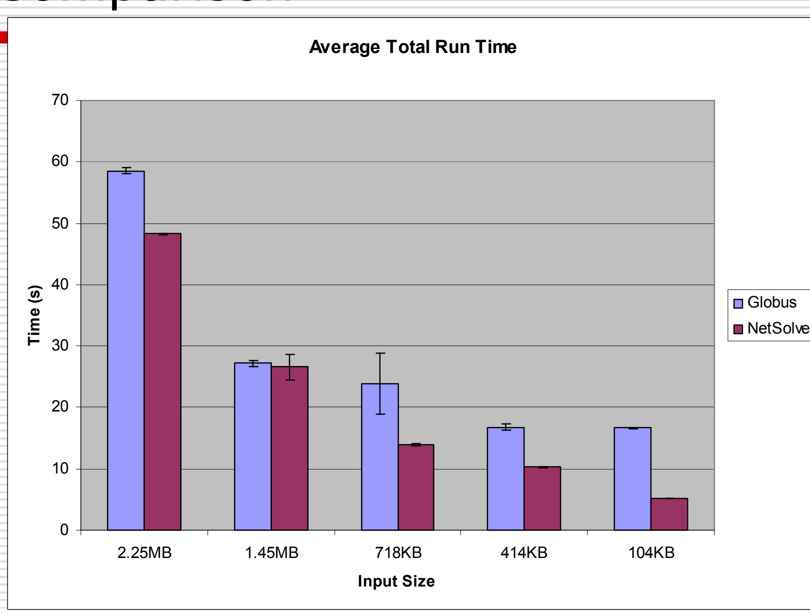
Globus Benchmark Results



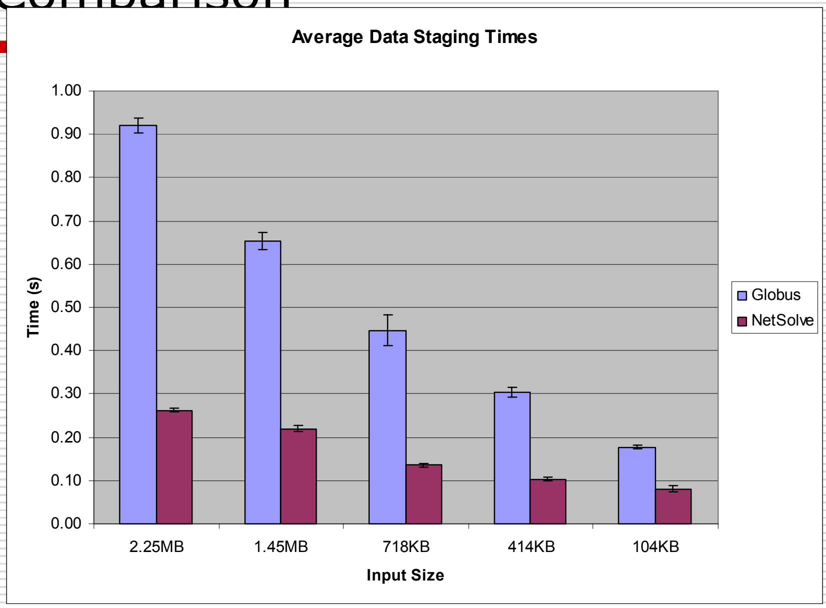
NetSolve Benchmark Results



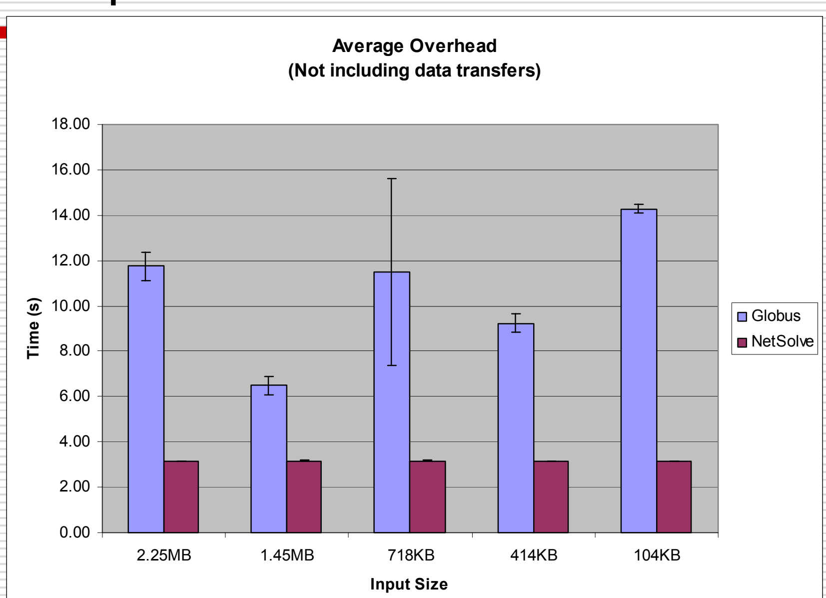
NetSolve / Globus Results Comparison



NetSolve / Globus Results Comparison



NetSolve / Globus Results Comparison

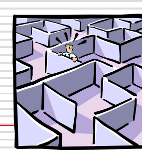


Conclusions

- For this application the NetSolve implementation is superior to the Globus implementation.
- Globus carries a larger overhead than NetSolve.
- The degree of variablity in the Globus measurements is much greater than the variability of the NetSolve measurements.
- Fluctuations in network traffic and server load may have influenced the large increase in variability in some of the test cases.

- Globus may have better results when used over a WAN or with larger data files, do to its ability to use multiple parallel data streams during data transfers.

Lessons Learned



- Grid magnifies performance related problems we haven't solved well on large scale systems, SMP, or in some cases sequential processors.
- Performance evaluation is hard
  - Dynamic nature
- Automate the selection
  - User doesn't want or know how
- Need performance model
  - Automagic would be best
- Need info on grid performance (NWS)
  - BW/Lat/processor/memory
- Monitoring tools

- Performance diagnostic tools are desperately needed.
  - Lack of tools is hampering development today.

- **This is a time for experimentation, not standards**

Conclusions: What is Needed

- Execution infrastructure for adaptive execution
  - Automatic resource location and execution initiation
  - Dynamic configuration to available resources
  - Performance monitoring and control strategies
    - deep integration across compilers, tools, and runtime systems
    - performance contracts and dynamic reconfiguration
- Abstract Grid programming models and easy-to-use programming interfaces
  - Problem-solving environments
- Robust reliable numerical and data-structure libraries
  - Predictability and robustness of accuracy and performance
  - Reproducibility and fault tolerance
  - Dynamic reconfigurability of the application

Thanks to

- Sudesh Agrawal
- Brian Drum
- Shirley Moore
- Keith Seymour
- Zhiao Shi
- Asim YarKhan