

Grid Computing: Back to the Future

David H. Bailey
Lawrence Berkeley Natl. Lab.
<http://crd.lbl.gov/~dhbailey>

Thanks to (among others):
Fran Berman (SDSC), Ian Foster (ANL)
and William Johnston (LBNL)

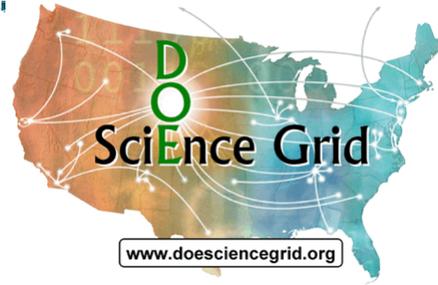
Grid Computing Today



NEESgrid



European
GRID
Forum



NPAC
GRID

TERAGRID



GEON

EUROGRID

*NSF
Cyberinfrastructure*



Driving the Grid: Scientific Collaborations



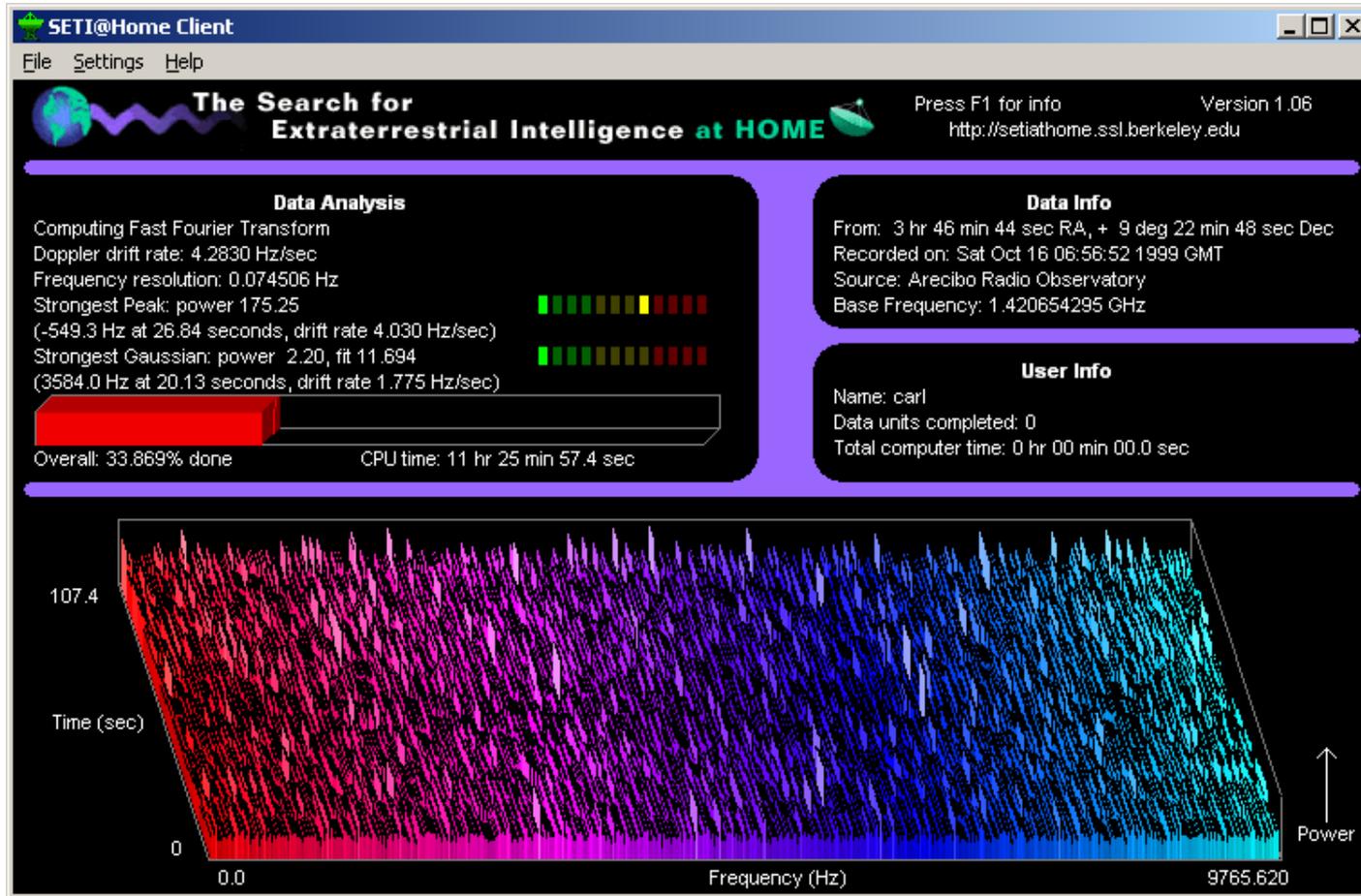
Large-scale science and engineering problems require collaborative use of computational, data, and instrument resources:

- ◆ Developed by independent teams of researchers.
- ◆ Obtained from multiple instruments.
- ◆ Housed at different geographic locations.

The required infrastructure includes:

- ◆ High-speed networks and services.
 - ◆ Very high-speed computers and large-scale storage.
 - ◆ Highly capable middleware, including support for distributed data management and collaboration.
-

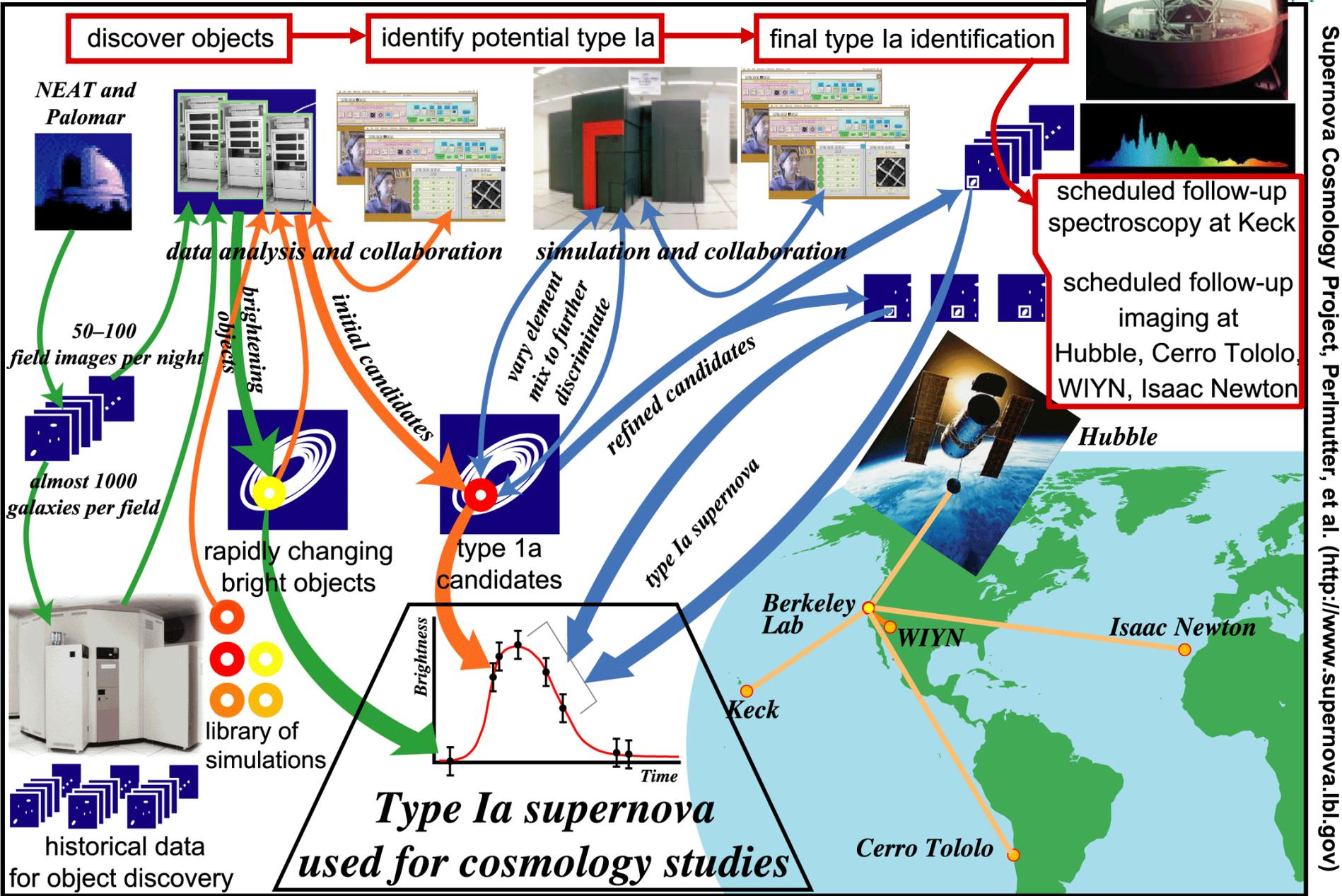
Case Study: SETI@Home



Seti@home sustains 35 Tflop/s on 2M+ systems
1.7 x 10²¹ flops over 3 years

Supernova Cosmology Infrastructure

[Thanks to W. Johnston, LBNL]



The NSF/DOE TeraGrid



- ◆ Over 20 Tflop/s aggregate distributed at 9 sites (SDSC, NCSA, ANL, PSC, Caltech, Indiana, Purdue, ORNL, UTenn Knox, UTex Austin).
- ◆ Over 1 PByte mass storage distributed at 5 sites.
- ◆ Fast national network with 40 Gbyte/s between hubs.
- ◆ Linux-based software environment with uniform administration.

The screenshot shows a web browser window displaying the TeraGrid website. The address bar shows the URL: `http://repo.teragrid.org/inca/cgi-bin/stack.cgi?file=../src/samples/teragrid/etc/ctss_prod.xml`. The page title is "Common TeraGrid Software Stack 1.1".

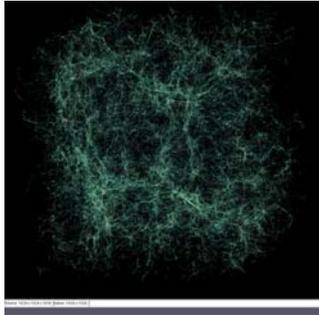
Under the heading "Find Status of:", there is a list of software packages:

- ATLAS
- BIOS
- BLAS
- Condor-G
- FFTW
- FPSWA
- GPFS
- GPT
- GPT_Wizard
- Goto
- Intel_C_Cplusplus
- Intel_Fortran
- Java_COG
- LAPACK
- MKL
- MPICH
- MPICH-VMI
- Myricom_MPICH-GM
- PETSc
- PVFS
- Qlogics
- Sysconnect drivers
- TotalView
- VMI-CRM
- gcc
- glibc
- globus
- gsi-ncftp
- gsi-openssh
- hdf4
- hdf5
- kernel
- maui
- myricom_gm
- openpbs
- openssh
- openssl
- python
- softenv
- srb
- suse
- xcat

Below the list, there is a table showing the availability of these packages across different sites. The table has columns for the package name, version, and the sites (ANL, CalTech, NCSA, PSC, SDSC).

Package	Version	ANL	CalTech	NCSA	PSC	SDSC
ATLAS	[download]					
BIOS						
	version					
	897					
BLAS	[download]					
	version					
	3.0					
Condor-G	[download]					
	subpackages					
	version					
NMI_Binary_Bundle	2.1					

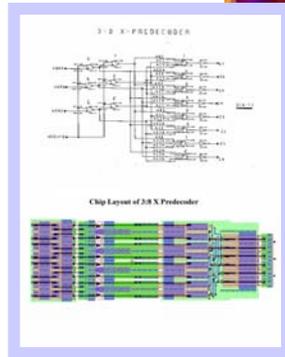
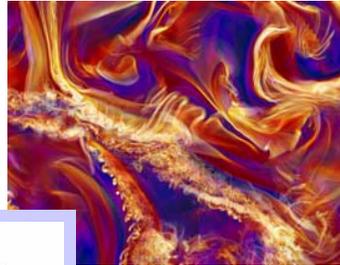
Some TeraGrid Applications



ENZO

(Astrophysics)

PPM
(Astrophysics)



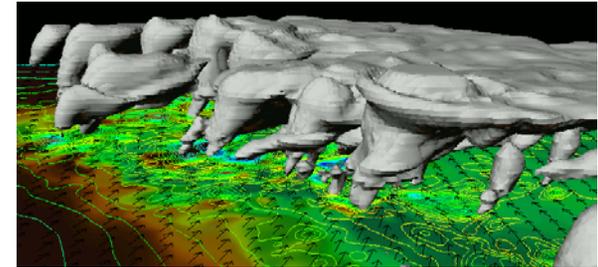
GridSAT

(Computer Science)



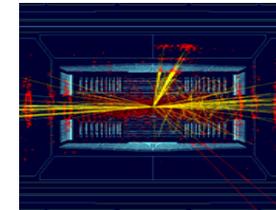
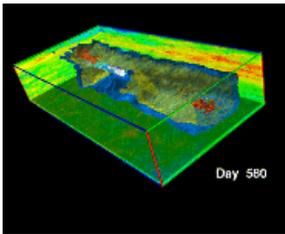
AtlasMaker

(Astronomy)



MEAD (Atmospheric Sciences)

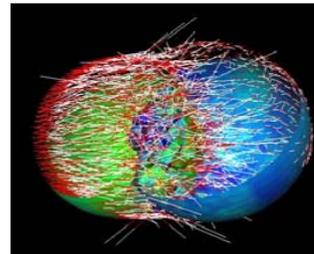
GAFEM
(Ground-water modeling)



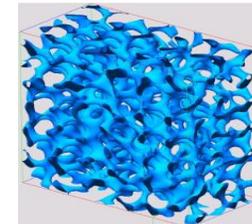
**CMS/
GriPhyN**
(Physics)



Encyclopedia of Life
(Biosciences)



VTF
(Shock Physics)



TeraGyroid
(Condensed Matter Physics)

BioCoRE (Biomedicine)
Biological Collaborative Environment

WalMart Inventory Control



- ◆ Satellite technology is used to track every item.
- ◆ Bar code information is sent to remote data centers that update inventory database and produce cash flow estimates.
- ◆ Satellite networking is used to coordinate vast operations.
- ◆ Inventory is adjusted in real time to avoid shortages and predict demand.



Vision of Grids



- ◆ Usable
 - ◆ High capacity
 - ◆ High capability
 - ◆ Evolutionary
 - ◆ Persistent
 - ◆ Stable
 - ◆ Scalable
 - ◆ Integrative
-

Implementation Is the Real Problem



- ◆ Building grid environments makes research questions out of previously solved problems:
 - ◇ Installation
 - ◇ Configuration
 - ◇ Accounting
 - ◆ And brings added complexity to existing problems:
 - ◇ Performance analysis
 - ◇ Debugging
 - ◇ Scheduling
 - ◇ Security
 - ◇ Fault tolerance
-

Potential for Overselling the Grid



- ◆ “All supercomputer computations will soon be done on grids.”
- ◆ “With the grid, every scientist will have access to all scientific data.”
- ◆ “All corporate data processing will soon be done by ‘computing utilities’.”
- ◆ Etc.

What the Grid Does Well



- ◆ Providing national or international access to important scientific datasets.
 - ◆ Providing a uniform scheme for remote system access and user authentication.
 - ◆ Providing a high-performance parallel platform for certain very loosely coupled computations.
 - ◆ Providing a high-capability platform for large computations that can run on a single remote system (chosen at run time).
 - ◆ Enabling new types of multi-disciplinary, multi-system, multi-dataset research.
-

What the Grid Doesn't Do So Well



- ◆ Scientific computations that require heavy interprocessor communication.
 - ◇ Probably the majority of high-end scientific computations are of this nature.
 - ◇ This doesn't rule out such applications running remotely on a single system connected to the grid.
- ◆ Many classified or proprietary computations.
 - ◇ Current grid security and privacy are not convincing for many of these users
 - ◇ This doesn't rule out "internal grids" -- some have been quite successful.

History of Parallel Computing



- ◆ 1982-1986: Academic studies and demos.
 - ◆ 1986-1990: First large systems deployed.
 - ◆ 1990-1994: Over-hyped.
 - ◆ 1994-1998: Funding cuts.
 - ◆ 1998-2002: Reassessments.
 - ◆ 2002-2006: Recovering?
-

Parallel Performance Practices, circa 1990



- ◆ Performance results on small-sized parallel systems were linearly scaled to full-sized systems.
 - ◇ Example: 8,192-CPU CM-2 results were linearly scaled to 65,536-CPU results.
 - ◇ Rationale: “We can’t afford a full-sized system.”
 - ◇ Sometimes this was done without any clear disclosure in the paper or presentation.

Parallel Performance, circa 1990



- ◆ Highly tuned programs were compared with lightly tuned or untuned implementations on other systems.
- ◆ Inefficient algorithms were used.
 - ◇ Algorithms were often chosen to exhibit a high Mflop/s rate on the target system, not for fundamental run-time efficiency.
 - ◇ Some scientists used explicit schemes where implicit schemes were known to be much better.
 - ◇ One paper described doing a discrete Fourier transform on a parallel architecture, rather than by using an FFT (n^2 operations rather than $5n \log_2 n$).

- ◆ Performance rates with 32-bit floating-point data were compared with 64-bit performance on other systems.
 - ◇ Using 32-bit data instead of 64-bit data effectively doubles data bandwidth, thus giving artificially high performance.
 - ◇ Some serious technical computations can be done safely with 32-bit accuracy, but most cannot, especially very large calculations.

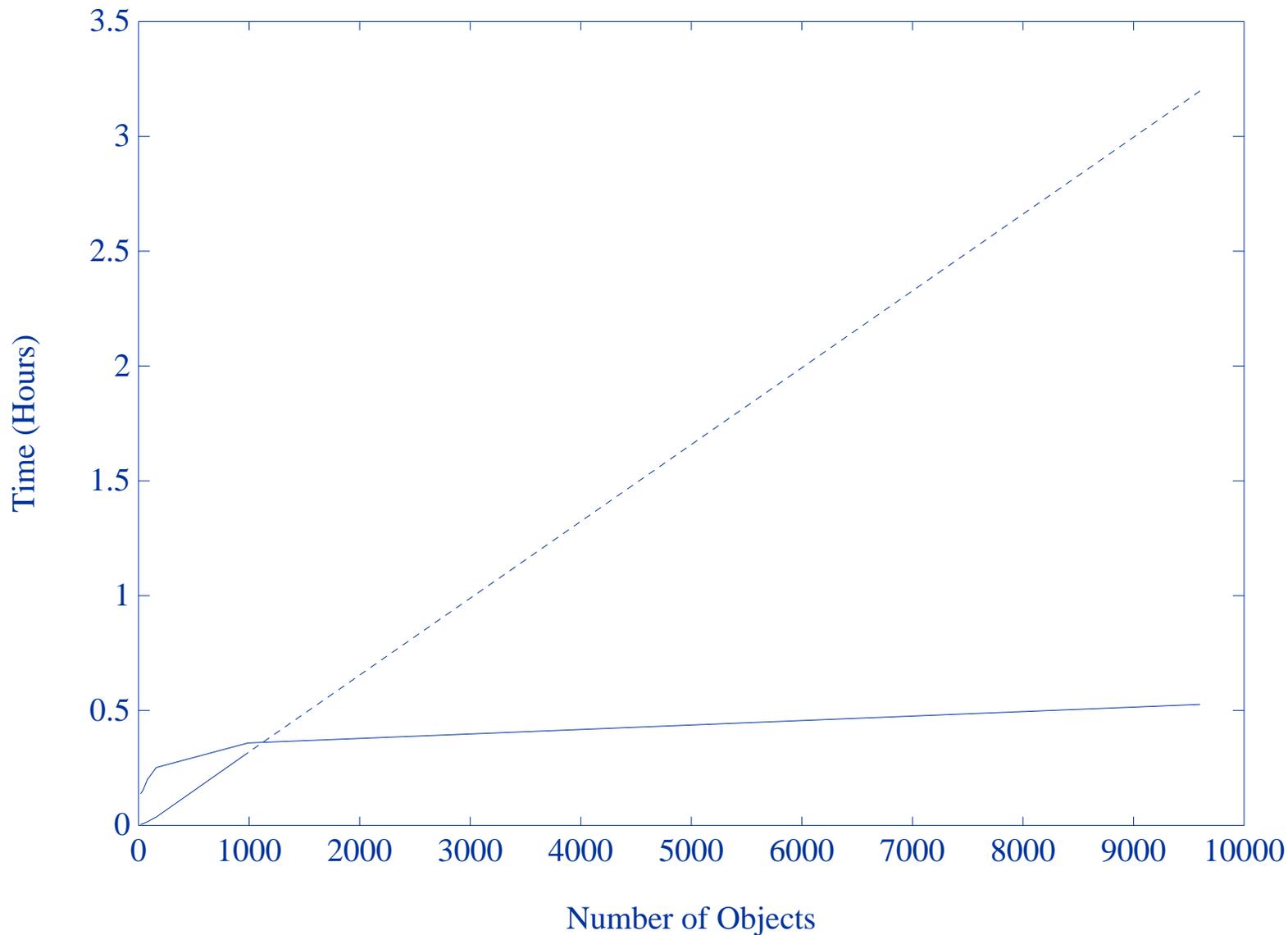
Parallel Performance, circa 1990



- ◆ Scientists were just as guilty as commercial vendors of “hyping” and “stretching” their results.
 - ◇ All the examples in my files are from journal papers and conference proceedings, written by professional scientists.
 - ◇ One example is from an award-winning paper.

Scientists should have a higher standard than vendor marketing people...

Performance Plot A



Data for Plot A

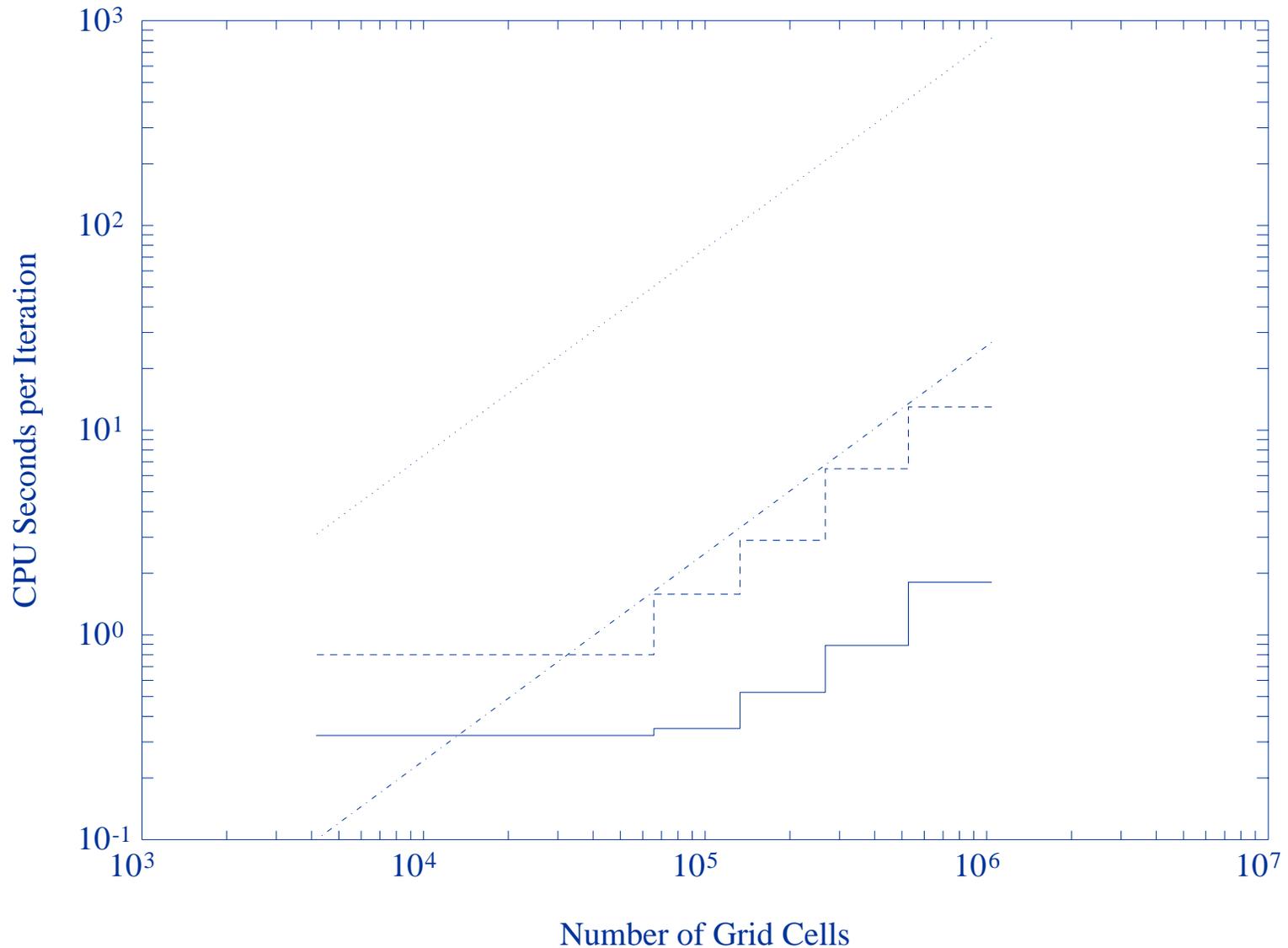


Total Objects	Parallel Run Time	Vector Run Time
20	8:18	0:16
40	9:11	0:26
80	11:59	0:57
160	15:07	2:11
990	21:32	19:00
9600	31:36	3:11:50*

Notes:

- ◇ Vector code is not “optimized.”
- ◇ In last entry, the 3:11:50 figure is an estimate.
- ◇ Vector performance is better except for last entry.

Performance Plot B



Facts for Plot B



- ◆ 32-bit performance on parallel system is compared with a 64-bit performance on vector system.
- ◆ Parallel results have been linearly extrapolated to a full-sized system from a small system (likely only 1/8 size).
- ◆ Vector version of code is “unvectorized.”
- ◆ Vector system “curves” are straight lines – i.e., they are linear extrapolations from a single data point.

It appears that of all points on four curves in this plot, at most two points represent real timings.

Twelve Ways to Fool the Masses



1. Quote only 32-bit performance results, not 64-bit results.
2. Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application.
3. Quietly employ assembly code and other low-level language constructs.
4. Scale up the problem size with the number of processors, but omit any mention of this fact.
5. Quote performance results projected to a full system.
6. Compare your results against scalar, unoptimized code on conventional systems.

Twelve Ways to Fool the Masses



7. When direct run time comparisons are required, compare with an old code on an obsolete system.
8. If Mflop/s rates must be quoted, base the operation count on the parallel implementation, not on the best sequential implementation.
9. Quote performance in terms of processor utilization, parallel speedups or Mflop/s per dollar.
10. Mutilate the algorithm used in the parallel implementation to match the architecture.
11. Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment.
12. If all else fails, show pretty pictures and animated videos, and don't talk about performance.

Twelve Ways: Basic Principles



- ◆ Use well-understood, community-defined metrics.
- ◆ Use efficient algorithms, not schemes chosen just to exhibit artificially high performance rates (i.e., base the operation counts for calculating Mflop/s rates on efficient algorithms).
- ◆ Provide full details of experimental environments, so that performance results can be reproduced by others.
- ◆ Disclose any details that might affect a reasonable interpretation of the results.
- ◆ Honesty and reproducibility should characterize all work.

Danger: We can fool ourselves, as well as others.

Technology

Measuring How Fast Computers Really Are

By JOHN MARKOFF

IN the world of scientific and technical computing, everyone agrees that computer speeds are increasing at a geometric rate. But measuring that speed is a vexing task. Rival supercomputer and work station manufacturers are prone to hype, choosing the performance figures that make their own machines look best.

"It's like the Wild West," said David J. Kuck, of the Center for Supercomputing Research and Development at the University of Illinois. "They say whatever they want to."

In fact, said David H. Bailey, a scientist at the National Aeronautics and Space Administration, "It's not really to the point of widespread fraud, but if people aren't a little more circumspect, the entire field could start to get a bad name."

The matter is complicated by a new generation of computers that have dozens, or even thousands, of separate processors. These parallel computers split problems into small parts and solve them simultaneously to reach greater speeds.

As a result, dozens of programs for determining benchmarks — measurements of computer speed — have been developed by scientists at universities and in government agencies. Some are based on how long a computer takes to solve a certain set of equations, while more sophisticated benchmarks attempt to match the operations re-

quired by real-world programs. But each benchmark generally measures only a single aspect of computer performance.

Just as a car buyer might buy a vehicle with the highest E.P.A. gas mileage rating for the price, a computer buyer could use benchmarks in deciding which machine to buy. But like their counterparts in the auto business, computer makers would do well to remind customers, "Your mileage may vary." The industry has no independent organization, analogous to the Environmental Protection Agency, to establish a single standard.

The proliferation of benchmarks is particularly problematic among the fastest scientific machines, where more than a dozen start-up companies compete to sell to university, corporate and Government laboratories.

These machines sell for hundreds of thousands of dollars or more, and the sale of only a few can mean success for a company. Supercomputers and smaller scientific work stations work on problems ranging from designing pharmaceuticals and weapons to weather modeling and the simulated crashing of automobiles.

Uneasy about the tendency for manufacturers to cite inflated claims, Mr. Bailey of NASA wrote a tongue-in-cheek indictment of performance claims for Supercomputing Review magazine in August. Titled "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers," it pokes fun at the tendency of computer mak-

Different Benchmarks, Different

The six fastest computers according to various benchmark point operations per second. Slalom, the only one accomplished in a set amount of time. The Perfec

LINPACK

Cray Y-MP/16	403
NEC SX-3/14	314
Cray Y-MP/832	275
Fujitsu VP2600/10	249
Cray X-MP/416	178
Cray 2S/4-128	129

Sources: Oak Ridge National Laboratory, Supercomputing Review, University of Illinois, University of Tennessee

ers to play fast and loose with speed claims.

It is common practice to "tune" computers and software to score better on benchmarks. "I know of a couple of companies who have full-time people, and all they do is optimize programs to achieve better benchmark results," said Gary Smaby, president of the Smaby Group, a consulting and market research firm in Minneapolis.

Such optimization is permissible under the rules established by benchmark designers to insure that computer makers can extract the full capability from their systems.

But some manufacturers go further and insert modules called "recognizers" into their compilers — software that translates a

Excerpts from NYT Article



“Rival supercomputer and work station manufacturers are prone to hype, choosing the performance figures that make their own systems look better.”

“It’s not really to the point of widespread fraud, but if people aren’t somewhat more circumspect, it could give the field a bad name.”

What Is Needed in the Grid Performance Arena



- ◆ A handful of well-designed, robust, scalable performance benchmarks.
 - ◇ Must be produced by a community-based effort.
 - ◇ Must be based on sample codes that have some credibility as a “useful” application.
 - ◇ Must be easily implemented without lengthy, highly expert effort.
 - ◇ Must be appropriate for modest-sized grids as well as very large, national or international-scale grids.
-

What Is Needed, Cont.



- ◆ Performance benchmarks must be accompanied by some well-thought-out “ground rules.”
 - ◇ How much tuning of the benchmark is permitted?
 - ◇ How is the extent of tuning measured?
 - ◇ How will disputes be settled?

If ground rules can be abused, they will be abused.

What Is Needed, Cont.



- ◆ A rational scheme for calculating performance rates.
 - ◇ How is run time measured?
 - ◇ Is required initialization included in the run time?
 - ◇ How will operation counts or work be reckoned?
- ◆ A well-defined test to validate the correctness of the results.
 - ◇ It is best if the benchmark includes its own scalable validity test.
 - ◇ At the least, spot checks of results are needed.

What Is Needed, Cont.



- ◆ A well-supported repository of results.
 - ◇ Kept up to date.
 - ◇ Includes all environmental and system information.
 - ◇ New results periodically solicited.
 - ◇ A searchable database is preferred.
 - ◇ Multi-lab, multi-university, multi-discipline support.

Sample issue: How will the load on the grid at the time be handled? Will multiple runs be allowed?

Twelve Ways: Back to the Future



- ◆ Use well-understood, community-defined metrics.
- ◆ Use efficient algorithms, not schemes chosen just to exhibit artificially high performance rates (i.e., base the operation counts for calculating Mflop/s rates on efficient algorithms).
- ◆ Provide full details of experimental environments, so that performance results can be reproduced by others.
- ◆ Disclose any details that might affect a reasonable interpretation of the results.
- ◆ Honesty and reproducibility should characterize all work.

Danger: We can fool ourselves, as well as others.