ASCAC Panel on Metrics Report
27 February 2007;
Gordon Bell, Co-Chair

Panel: F. Ronald Bailey, Gordon Bell, John Blondin, John Connolly, David Dean, Peter Freeman, James Hack, Steven Pieper, Douglas Post*

Centers: Ray Blair (ANL), Al Geist (ORNL), Douglas Kothe (ORNL), Bill Kramer (NERSC), David Skinner (NERSC), Francesca Verdier (NERSC).

Christine Chalk (DOE), Brad Comes (DOD HPC Modernization Program), Mike Levine (PSC), Mark Snavely (SDSC)

The Orbach charge*

“The sub-panel should weigh and review the

1.approach to performance measurement and assessment at these facilities,

2.appropriateness and comprehensiveness of the measures, and

3.computational science component of the science accomplishments and

4.their effects on the Office of Science’s science programs.

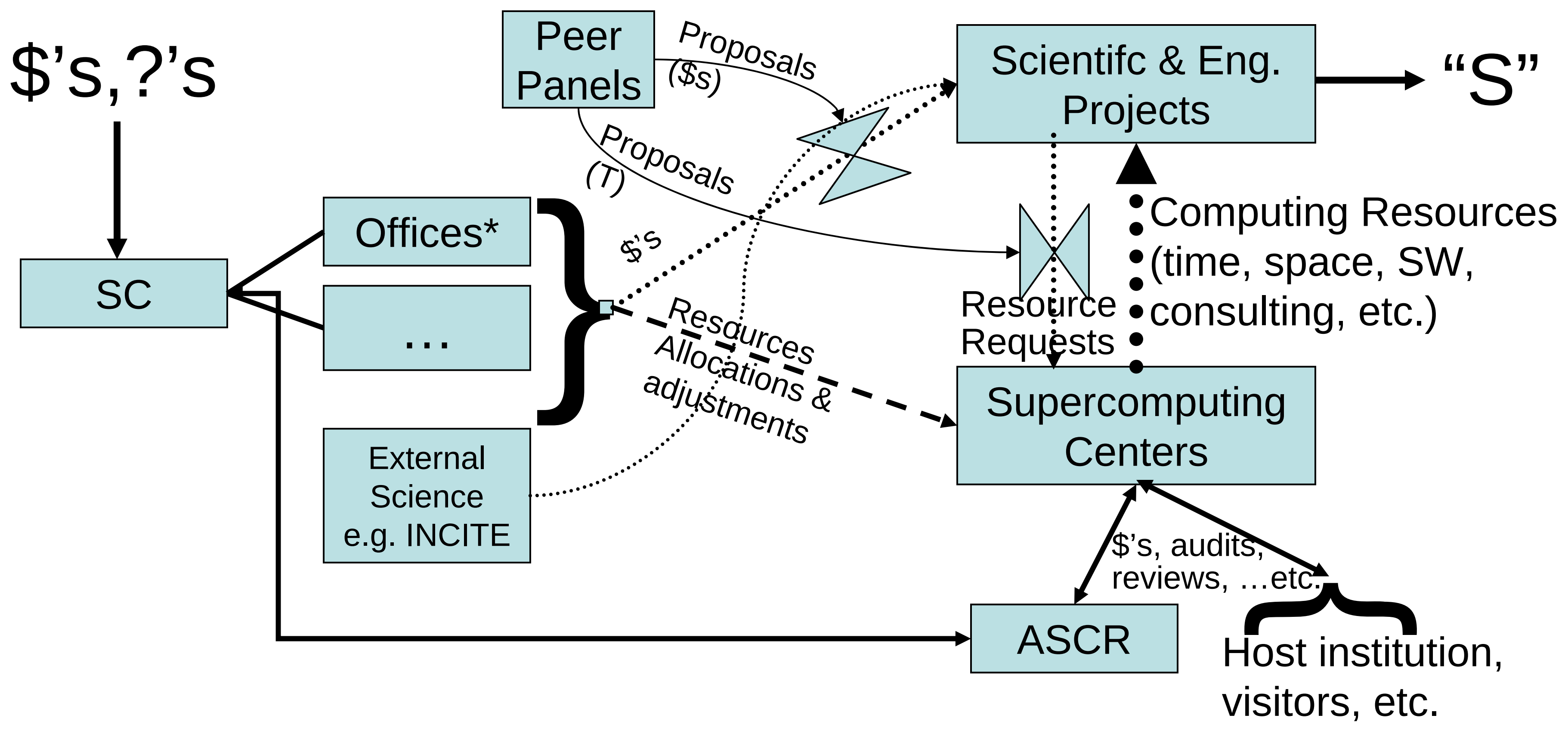
Additionally, the sub-panel should consider the

5. evolution of the roles of these facilities and

6. computational needs over the next three - five years, so that SC programs can maintain their national and international scientific leadership.

In addition … comments on observed strengths or deficiencies in the management of … ASCR’s portfolio and suggestions for improvement….”

ASCR – Centers Support of Science Offices


$’s,?’s
SC
Offices*
…
External Science e.g. INCITE
Peer Panels
Proposals ($s)
Proposals (T)
$’s
Resources Allocations & adjustments
Scientifc & Eng. Projects
“S”
Resource Requests
Computing Resources (time, space, SW, consulting, etc.)
Supercomputing Centers
$’s, audits, reviews, …etc.
ASCR
Host institution, visitors, etc.

*BER, BES, FES, HEP, NP

Element 1. Centers “control” metrics for performance measurement and assessment

The Panel recommends the following “control” metrics e.g. those used by PART for the centers performance:

1. User Satisfaction (overall) of provided services obtained via user surveys. *NERSC survey is recommended*
2. Scheduled Availability. Overall Availability is “observed”.
3. Response time to solve user problems as measured by the centers' trouble reporting systems.
4. Support for high capability work at LCF as per agreements; *observed and reported* distributions of jobs

Element 1. Centers “Observed” Metrics

- Constituent metrics that make up “user satisfaction” Indicate: user sophistication, level of support, unproductive scientist time, software and hardware reliability, need for system software, etc.
- Scheduled & overall system uptime, <u>… hardware & software reliability</u>
- Utilization of centers resources.
- Utilization of standard and specialized software, including research application codes that are shared by others.
- Size and growth in shared, on line experimental data and databases, such as the Protein Data Bank at NSF’s San Diego Center.
- Individual project metrics that need to be tracked over time include:
  - **Total computer resources as requested in Appendix 1: Project Checklist Metrics**
  - **job size distributions by runs, amount of time, and processors used;**
  - **percentage of successful job completion by number and by time**
- <u>Individual project program scalability and efficiency</u> on each platform

  *= speed-on-N-processors/(N*speed-on-one-processor)*
- Utilization of software engineering tools for configuration management, program validation and verification, regression testing, ….
- workflow management including the ability for “ensemble” experiments that exploit parallelism and allow many “computational experiments per day”

Element 2. Project metrics: complementary, comprehensive, and essential measures

2.1 Project Evaluation. The Panel's recommended checklist and metrics that cover the following seven aspects of projects … understood and tracked by the Projects, Centers, and Program Offices.

1. Project overview that includes clear goals
2. Project team resources
3. Project resource use from the center
4. Project "code" including portability, progress on scalability, etc.
5. Project software engineering processes
6. Project output and Computational Science Accomplishments … This provides a comprehensive listing of results that cover publications, people, technology, etc. over time.
7. Project future requirements

2.2 Code Improvement The Panel recommends a code improvement metric that is a combined measure of a scientific project's *mathematics, algorithms, code, and scalability*. Goal:
doubling the rate of solution every three years (replaces PART metric).

Implication: project database retained over a number of years!

Four Quadrant Scientific Project Views

```
Project (1), team (2),    | Scientific Output (6)
 & process                |
----------------------    | ------------------------
Centers resources         | The "code" &
  input (4)               |    code scalability (5)
```

Project Name

PIs and URL
DOE Office support: DOE program manager:
Scientific domain (chemistry, fusion, high energy, nuclear, other.),
Support for the development of the code
   Degree of DOE support to develop the code?
   SciDAC, DOE SC program
   internal institutional funding sources (e.g. LDRD,..),
   industry,
   other agencies, ….
What are the technical goals of the project?
   What problem or "grand challenge" are you trying to solve?
   What is the expect impact of project success?
What is the project profile in human resources including
   trained scientists, computational scientists and mathematicians,
   program development and maintenance,
   use(rs) of the team codes?
Ext communities & sizes, that code and/or datasets support.
Not shown, software engineering processes

Scientific Output

The scientific accomplishments 200x to present*:
The effect on the Office of Science programs*:
Publications/location:
Citations (last 5 years):
Dissertations?
Prizes and other honors?
Residual and supported, living datasets and/or databases that are accessed by a community? Size of the community?
Change in code capabilities and quality (t)
Code contributed to the centers
Code contributed to the scientific community at large
Company spin-offs based on code or trained people and/or CRADAs
Corporation, extra-agency, etc. use
Production of scientists & computational scientists during 2001-2005
Production of trained software engineers during 2001-2005

*Parts 3 & 4 of metrics approach

Centers resources

Steady state production use per month; per year
   Processor number
   Processor time
   Disk
   Tertiary amount and rate of change
Software provided by center
Consulting
Direct project support as a team member
What is the size of user jobs in terms of memory, concurrency (processors), disk, and tertiary store?
What is the scalability of these codes
What is the wall-clock time for typical runs?
What could the center provide that would enhance output?

27 February 2007

The Codes

Problem Type
Types of algorithms and computational mathematics
Code size (single lines of code, function points, etc.);
Code size as f (t) from the origin to the present
Computer languages
   LOC/ language 1/LOC…n
   What libraries used & fraction of code
Code Mix:
   To what extent does your team develop and use your own codes?
   Codes developed by others in the DOE and general scientific community?
   Commercial application codes provided by the center?
Platforms What is the present parallelism for each of the platforms
Projected or maximum scalability
   How is measured?
   Is the code massively parallel?
What memory/processor ratio do your project require? (e.g. Gbytes/processor)
Parallelization model (e.g. MPI, OpenMP, Threads, UPC, Co-Array Fortran, etc.)
What is the "efficiency" of the code? And how is it measured?
What are the major bottlenecks for code scaling?
What is the split between interactive and batch use?
Fraction f code development at center computer(s) versus own installation?

Element 3. The science accomplishments

1. **Publications, Code & Datasets, People, and Technology Transfer** (Appendix 1, Item 6 project checklist) goes beyond the traditional scientific measures... Equally important measures of output include professionals trained in computational science. … application codes and datasets that others use are comparably important measures of *computational* scientific output and should be identified as such., technology transfer, including the formation of new companies ….
2. **Project Milestones Accomplished versus the Proposal's Project Plan** is a near term and useful metric as to whether a project is on the path towards meeting well defined scientific goals. These goals have *presumably* been peer reviewed by the scientific community, and certified as having a legitimate scientific purpose..
3. **Exploiting Parallelism and/or Improved Efficiency aka Code Improvement** How well scientific applications take advantage of a given computational resource is key to progress through computation. Improved algorithms that increase code efficiency are critically important to the improvement of code effectiveness. … an increased computational rate for any given application can only be achieved by exploiting increased parallelism.
4. **Break-throughs; an immeasurable goal:** The Panel could not identify metrics or any method that could be used to anticipate discoveries that occur on the leading edge of fundamental science. The scientific communities can more effectively recognize breakthrough science, or even what constitutes a "significant advance.

Break-throughs

- solving a problem that had not been solved before -- new method of solving the protein folding problem using Monte Carlo techniques by Charles Strauss, LLNL;
- increasing code efficiency by several orders of magnitude --combustion calculation code by John Bell, LLBL;
- greatly reducing the disagreement between theory and experiment --the QCD calculations which validate the standard model of particle physics, by Christine Davies of Glasgow U.;
- greatly expanding the scope and scale of computational simulation to provide accurate or new results -- Fred Streitz, LLNL, showed that at least 8 million atoms are needed in a simulation in order to get consistent results for metal condensation. (Streitz won the 2005 Gordon Bell Prize competition using an IBM BlueGene/L with 131 thousand processors operating at 107 Teraflops.)

Element 4. *Computational Resources* Effects on the Office of Science's science programs.

- The Panel suggests that a clear process be implemented that measures the use and effects of the computational resources on the projects within each SC office.
- The Panel suggests that each SC office report the total investment in all projects, using a rough conversion of computer time to dollars.   The Panel believes that computational resources need to be treated in a substantially more serious and measured fashion by the program offices and project personnel.
- The Panel suggests the process of allocating computer time at the Centers through the program offices be re examined in light of the diversity of architectures. Given the variety of platforms at the Centers and need for user code portability, the efficiency of a particular code will be highly variable.

Element 5. Evolution of facilities and roles

- The plan is reasonable…premature to predict how the centers and machine roles
  - Will workload be able to exploit the changes?
  - Many more processors with limited memory bandwidth
  - Variable scale factors: processors, i/o, checkpoint, etc.
  - Machines can be segmented into various roles.
  - BG/L could become more of a factor
- Much more involvement of centers support to teams
- Much larger teams to manage the code and resulting data

Element 6. 3-5 year Computational Needs

- The effect of next generation micros is unclear: “while the science community has responded to increases in processor count in the past, it has taken time.”
- We can safely predict: “Based on the recent three decades of scientific computers whose performance has doubled annually, there is no end to the imaginable applications or amount of computing that science can absorb – *especially at zero apparent cost.*”
- “projections by scientists…to utilize any and every imaginable future computing resource is almost certain to outrun any and every realizable computational infrastructure.”
- “Science continues to demonstrate a nearly infinite demand for computing. Already researchers are discussing the needs and ability for exa-scale computers.”

End

Additonal comments

Chairman’s comments

1. **Centers are service factories and MORE –** ***learning curves apply …***
   e.g. Doubling jobs served improves performance or cost 10-15%
   a. <u>Benchmarking is the only way to improve centers operations</u>
   b. Centers meetings with DoD, NSF, DoE, NASA, centers would payoff
   c. Centers must take responsibility for scientist-experiment efficiency
   d. User training and support is a significant ongoing activity
2. **<u>All (DOD, DOE, NSF) centers need common, annual FASB-like report!</u>**
3. **Common administrative programs e.g. allocation, reporting, surveys?**
4. **Many machine types reduce code & user effectiveness. Fewer types?**
5. **Time to reach steady-state for managers, centers, machines, codes?**
6. **Is ASCR vs. Centers the right balance of central vs. local control?**
7. **What is the feasibility & effectiveness of NCAR-like discipline centers?**
8. **Does ASCR fund smaller, project centered machines <1 Mhr ($1M)?**
9. **Are the centers too small?**

Challenges for scientific codes for petaflop machines scaling from 1,000s to 100,000s processors

- <u>Constant state of variable and evolving machine types and characteristics</u> Somewhat undefined and highly variable architectures and configurations for the targeted petaflop computers requiring machine-specific performance expertise!
- Evolution of small code development teams to larger teams
- Increased use of multi-disciplinary and multi-institutional teams
- Achievement of adequate levels of code performance and efficiency
- Greater need for software project management practices
- Greater utilization of software engineering practices and metrics
- Relatively immature code development , production tools, and environment for parallel program control and development
- Verification and validation of applications for more complex systems
- Development of problem generation methods for larger and complex problems
- Calculating the trade-off of many different strongly interacting effects across more orders of magnitude of multiple time and distance scales
- Analysis <u>and visualization</u> of larger and more complex datasets

Challenge: Man & their Codes vs. Machine

- All machines are different!
  - NERSC 2 Clusters & Constellations
  - ORNL: Vector mP, 4 Clusters (Jaguar 05,06,07,08)
  - ANL: BG
- Nodes: 4+ processor-types, Processors/node; Memory/node
- Interconnection characteristics
- I/O
- *For learning curves, Less is MORE!*
- All code is portable… and necessarily not tuned for a machine.
- *Code can very likely be improved for any specific machine!*
- WORKING Code Algorithms, machines, scale vary ~10X!
- *Codes >1 Mhr/yr ≈ $1 M/yr likely to benefit by 1 Person!*

So many things to measure…so little time

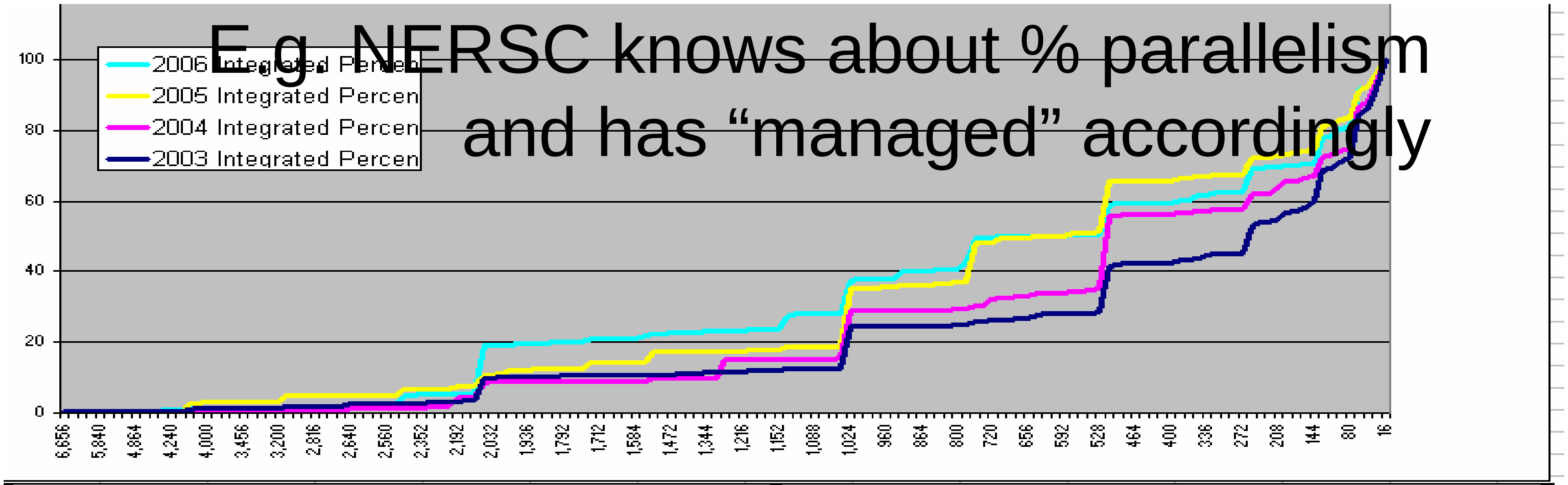
-------------- At the Centers--------------------------

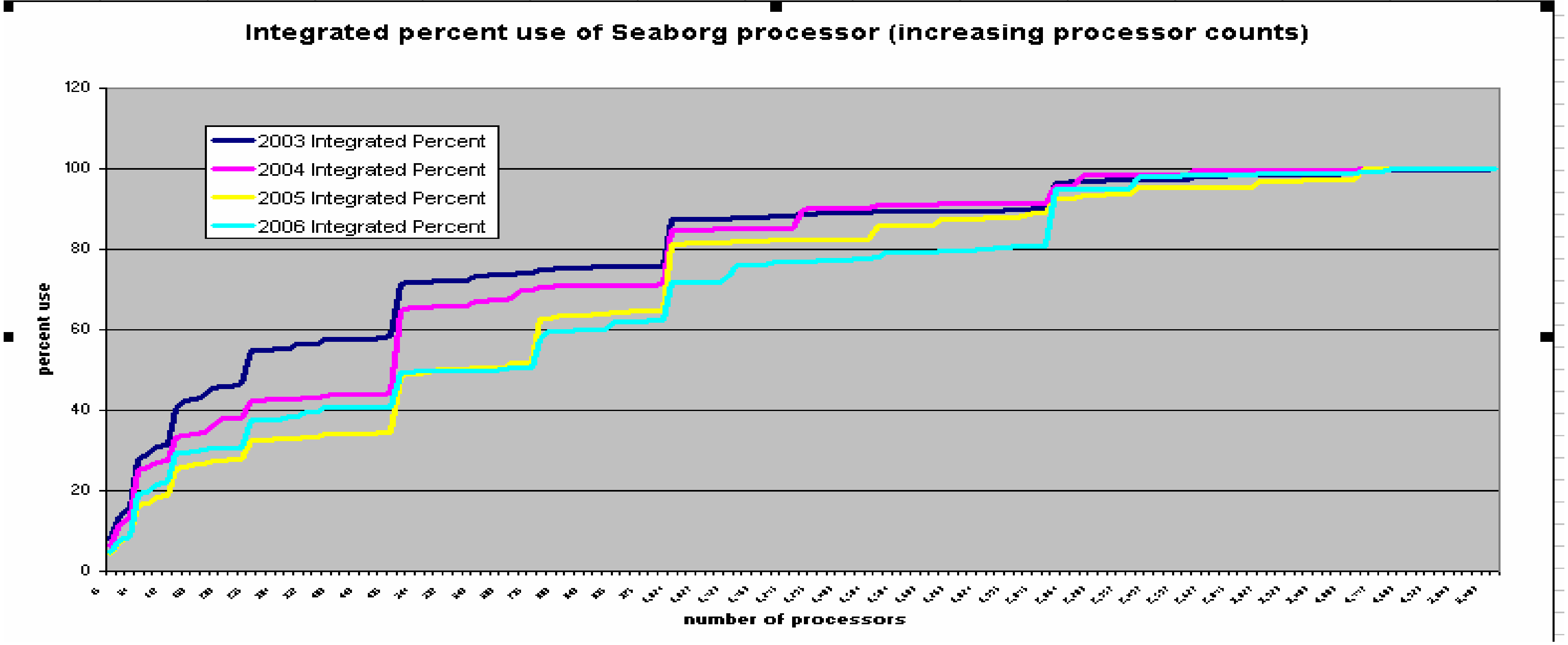
- System uptime: overall and scheduled; hardware and software reliability
- Delivered centers computing resources by users, job size distribution vs. time
- Aggregate user satisfaction: user sophistication, support level, unproductive time, software & hardware reliability, system software needs, …
- Utilization of standard and specialized software on an individual and centers basis, including codes shared by others that might be more widely used. "Web services" aka Grid? called to carry out high level remote functions
- Size and growth in shared, on line data and databases; and web services

------------ FOR Scientists and their Codes ----------------------

- Individual project metrics: computer resources, job size distributions;
  % <u>good time or good jobs</u> = the time and jobs that complete successfully / total number of non-debug jobs submitted
- Program scalability and efficiency (% of FLOP is an important *observed metric).*
  – Efficiency is a potentially harmful metric
  – Nearly all codes run on (utilizes) at least two hardware platforms!
  – For peak performance, each code needs to be understood and modified to operate on a specific machine, configuration, and scalability-level.
- Software engineering tools: configuration management, program validation and verification, regression testing, validating the science with physical experiments
- Workflow management for capability: "ensemble" experiments that exploit parallelism and allow many "computational experiments per day".

E.g. NERSC knows about % parallelism and has “managed” accordingly





Codes vary by purpose, size, & need for increasing need for SW Engineering

- **Engineering design and analysis**— Design of a passively safe reactor core for the Advanced Burner Reactor, tokamak reactors, high energy accelerators,…
- **Prediction of operational conditions**—path of a hurricane, evolution of space weather, path of a satellite, exploration of potential operating modes for a tokamak reactor experiment, …
- **Experimental analysis and design**—the analysis of experimental data from DOE research facilities; or the design of a new high energy particle detectors
- **Scientific discovery**—study of new scientific phenomena such as calculating the trade-off many different effects to determine the most important mechanisms; or calculation of the non-linear behavior of a complex system such as the generation of a high-resolution first principles turbulence simulation dataset
- **Scientific design and analysis**—of large datasets (e.g. screening of all known microbial drug targets against the known chemical compound libraries, design of materials with specific properties), analysis of large datasets of turbulence simulations,..

Four Quadrant Scientific Project Views

```
Project (1), team (2),     | Scientific Output (6)
 & process                 |
-------------------------- | ----------------------------
Centers resources          | The “code” &
  input (4)                |    code scalability (5)
```

Project Name

PIs and URL
DOE Office support: DOE program manager:
Scientific domain (chemistry, fusion, high energy, nuclear, other.),
Support for the development of the code
- Degree of DOE support to develop the code?
- SciDAC, DOE SC program
- internal institutional funding sources (e.g. LDRD,..),
- industry,
- other agencies, ….

What are the technical goals of the project?
- What problem or "grand challenge" are you trying to solve?
- What is the expect impact of project success?

What is the project profile in human resources including
- trained scientists, computational scientists and mathematicians,
- program development and maintenance,
- use(rs) of the team codes?

Ext communities & sizes, that code and/or datasets support.
Not shown, software engineering processes

Scientific Output

The scientific accomplishments 200x to present*:
The effect on the Office of Science programs*:
Publications/location:
Citations (last 5 years):
Dissertations?
Prizes and other honors?
Residual and supported, living datasets and/or databases that are accessed by a community? Size of the community?
Change in code capabilities and quality (t)
Code contributed to the centers
Code contributed to the scientific community at large
Company spin-offs based on code or trained people and/or CRADAs
Corporation, extra-agency, etc. use
Production of scientists & computational scientists during 2001-2005
Production of trained software engineers during 2001-2005

*Parts 3 & 4 of metrics approach

Centers resources

Steady state production use per month; per year
- Processor number
- Processor time
- Disk
- Tertiary amount and rate of change

Software provided by center
Consulting
Direct project support as a team member
What is the size of user jobs in terms of memory, concurrency (processors), disk, and tertiary store?
What is the scalability of these codes
What is the wall-clock time for typical runs?
What could the center provide that would enhance output?

The Code

Problem Type
Types of algorithms and computational mathematics
Code size (single lines of code, function points, etc.);
Code size as f (t) from the origin to the present
Computer languages
- LOC/ language 1/LOC…n
- What libraries used & fraction of code

Code Mix:
- To what extent does your team develop and use your own codes?
- Codes developed by others in the DOE and general scientific community?
- Commercial application codes provided by the center?

Platforms What is the present parallelism for each of the platforms
Projected or maximum scalability
- How is measured?
- Is the code massively parallel?

What memory/processor ratio do your project require? (e.g. Gbytes/processor)
Parallelization model (e.g. MPI, OpenMP, Threads, UPC, Co-Array Fortran, etc.)
What is the "efficiency" of the code? And how is it measured?
What are the major bottlenecks for code scaling?
What is the split between interactive and batch use?
Fraction f code development at center computer(s) versus own installation?

1.0 Project name (Background)
PI & URL

DOE Office support: DOE program manager:
Scientific domain (chemistry, fusion, high energy, nuclear, other.),
Support for the development of the code
- Degree of DOE support to develop the code?
- SciDAC, DOE SC program
- internal institutional funding sources (e.g. LDRD,..),
- industry,
- other agencies, ….

What are the technical goals of the project?
- What problem or “grand challenge” are you trying to solve?
- What is the expect impact of project success?

What is the project profile in total human resources including
- trained scientists,
- computational scientists and mathematicians,
- program development and maintenance,
- use(rs) of the team codes?

External communities & sizes that code and/or datasets support.

2.0 Project Team Resources

- Team size & structure
- Team institutional affiliation(s). (e.g. all the institutions involved, including universities, national labs, government agencies,..). I.e. to what extent is the team multi-institutional?
- To what extent are the code team members affiliated with the computer center institution? (e.g. are the team members also members of the computer center institution?)
- Team composition and experience total
  - domain scientists,
  - computational scientists, computer scientists, computational mathematicians, database managers
  - programmers
  - other
- Team composition by educational level (total)
  - Ph.D.,
  - MS, BS, undergraduate students, graduate students, post-docs, younger faculty, senior faculty, national laboratory scientists, industrial scientists, etc.)
- Team resources utilization: time spent on code and algorithm development, maintenance, problem setup, production, and results analysis

3.Project Code

- Problem Type (data analysis, data mining, simulation, experimental design, etc.)
- Types of algorithms and computational mathematics
- What platforms does your code routinely run on?
- Code size (single lines of code, function points, etc.);
  - Code age and yearly growth.
- Computer languages employed,
  - LOC/ language 1;,LOC/ language 2 LOC/ language 3
  - Structure of the codes (e.g. 250,000 SLOC Fortran-main code, 30,000 C++-problem set-up, 30,000 SLOC Python-steering, 10,000 SLOC PERL-run scripts,…)
- What libraries are used? And What fraction of the codes does it represent?
- Code Mix:
  - To what extent does your team develop and use your own codes?
  - Codes developed by others in the DOE and general scientific community?
  - Commercial application codes provided by the center?
- What is the present parallel scalability on each of the computers the code operates on
  - Projected or maximum scalability
  - How is measured?
  - Is the code massively parallel?
- What memory/processor ratio do your project require? (e.g. Gbytes/processor)
- Parallelization model (e.g. MPI, OpenMP, Threads, UPC, Co-Array Fortran, etc.) E.g. Does your team use domain decomposition and if so what tools do you use?
- What is the “efficiency” of the code
  - how is it measured?
- What are the major bottlenecks for scaling your code?
- What is the split between interactive and batch use?
  - Why, and is interactive more productive
- What is the split between code development on the computer center computers and on computers at other institutions?

4.0 Project resources input from the centers

- Plan with benchmarks & milestones
- Steady state user of resources on a production basis per month
  - Processor number
  - Processor time
  - Disk
  - Tertiary amount and rate of change
- Annual use of resources
  - Processor time
  - Disk
  - Tertiary storage rate of change
- Software provided by center
- Consulting
- Direct project support as a team member
- What is the size of their jobs in terms of memory, concurrency (processors), disk, and tertiary store?
- What is the scalability of these codes
- What is the wall-clock time for typical runs?

5.0 Software Engineering, Development, Verification and Validation Processes

- Software development tools used (
  - parallel development,
  - debuggers,
  - visualization,
  - production management and steering
- Software engineering practices. Please list the specific tools or processes used for
  - configuration management,
  - quality control,
  - bug reporting an tracking,
  - code reviews,
  - project planning,
  - project scheduling an tracking
- What is your verification strategy?
- What use do you make of regression tests?
- What is your validation strategy?
- What experimental facilities do you use for validation?
- Does your project have adequate resources for validation?

5a. Project code productivity & scalability (Project-specific measures)

- Measures of experiment productivity and performance including scalability of runs
- Scaling limits including i/o, node memory size, interconnect b/w or latency, algorithm
- History of scaling
- Projected scalability

6. Scientific | Engineering Output

The scientific accomplishments 200x to present*:

The effect on the Office of Science programs*:

Publications/location:

Citations (last 5 years):

Dissertations?

Prizes and other honors?

Residual and supported, living datasets and/or databases that are accessed by a community? Size of the community?

Change in code capabilities and quality (t)

Code and/or data contributed to the centers

Code and/or data, results, contributed to the scientific and engineering community at large

Company spin-offs based on code or trained people and/or CRADAs

Corporation, extra-agency, etc. use

Production of scientists & computational scientists during 2001-2005

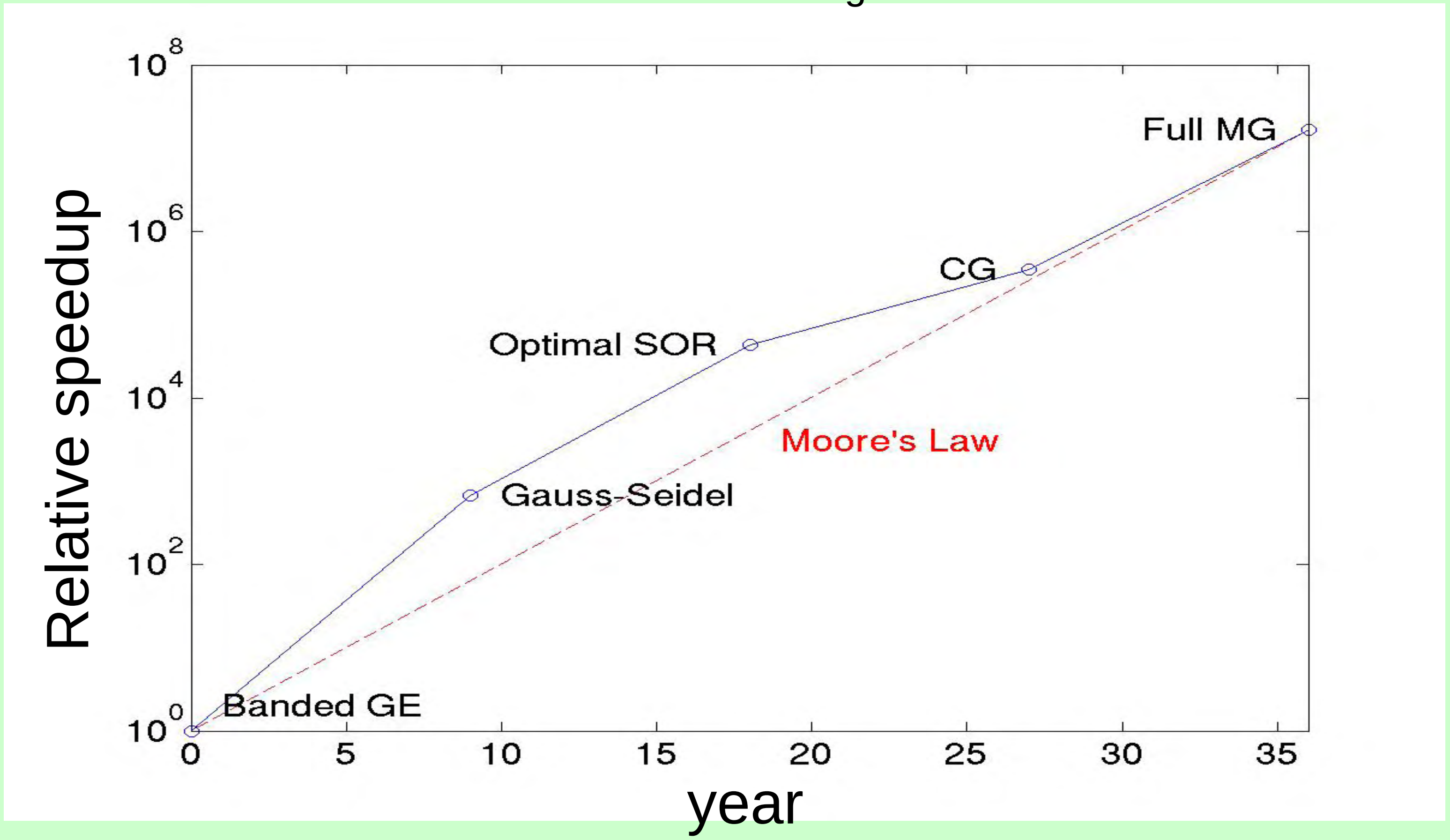
Production of trained software engineers during 2001-2005

*Parts 3 & 4 of metrics approach

Mathematics, Algorithms, and Code are all subject to "learning curves" courtesy, Jack Dongarra

Algorithms and Moore's Law

- This advance took place over a span of about 36 years, or 24 doubling times for Moore's Law
- $2^{24} \approx 16$ million $\Rightarrow$ the same as the factor from algorithms alone!



"Moore's Law" for combustion simulations

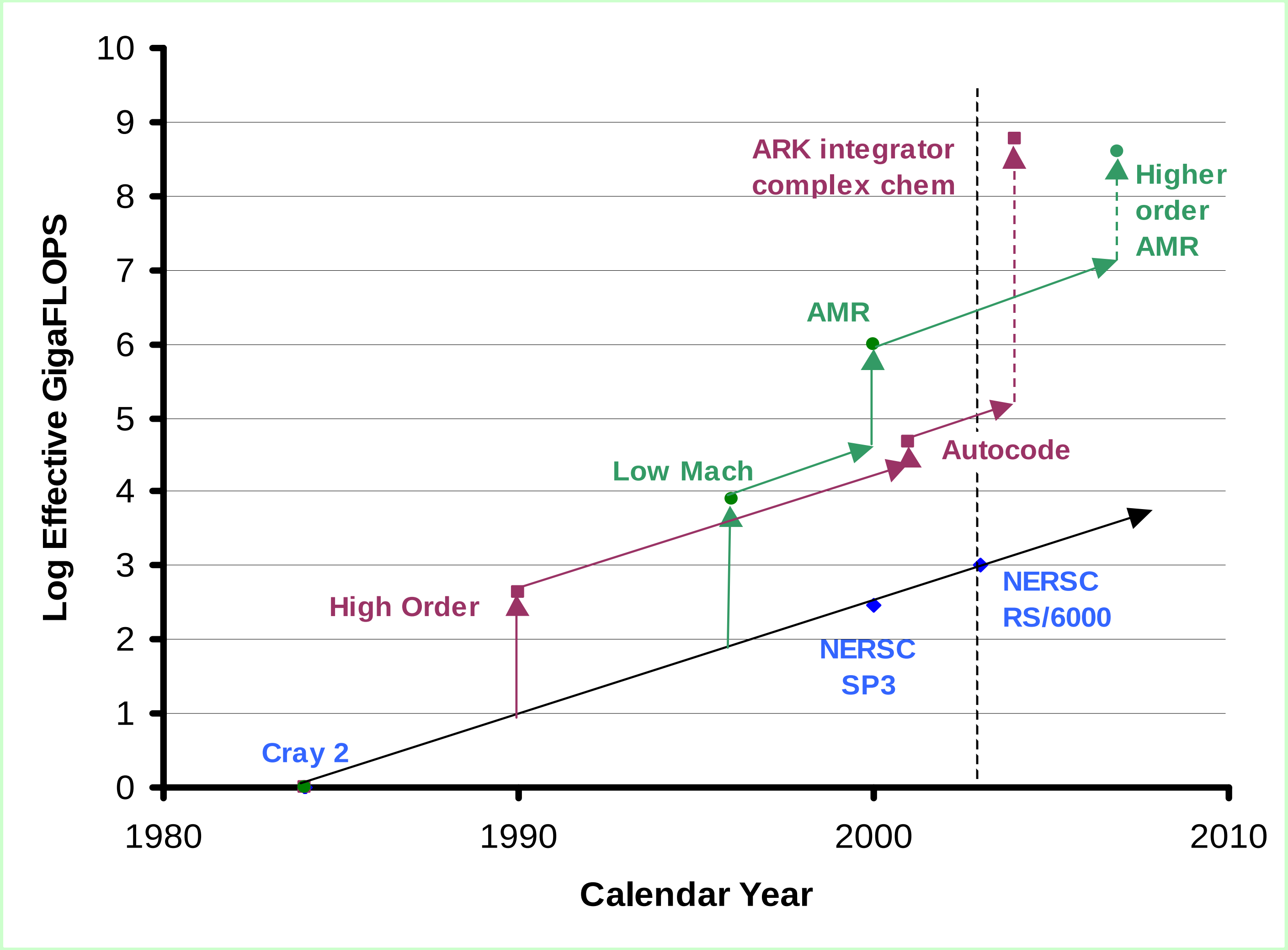


Figure from SCaLeS report, Volume 2

“Moore’s Law” for MHD simulations

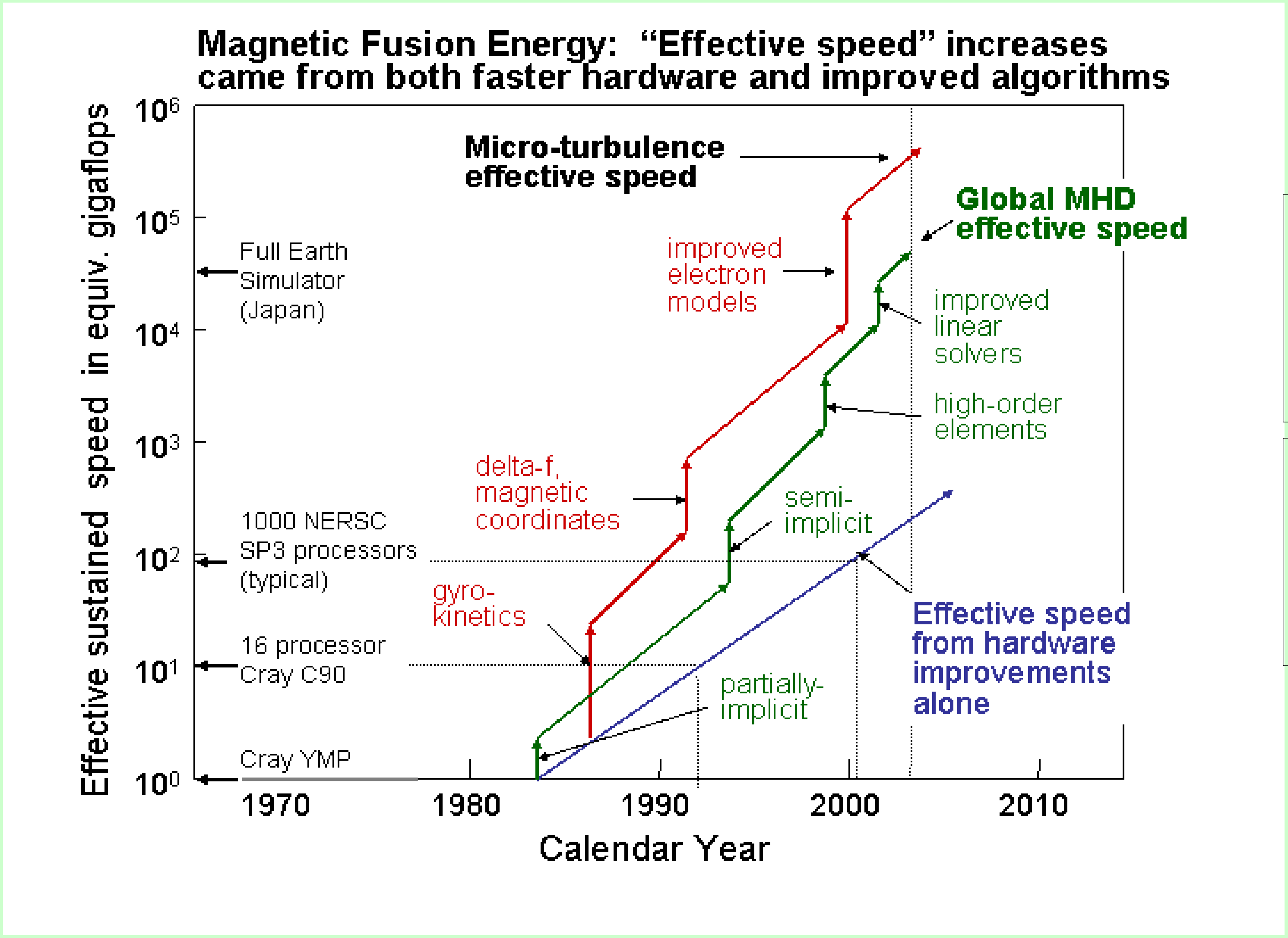


Figure from SCaLeS report, Volume 2

“Semi-implicit”:

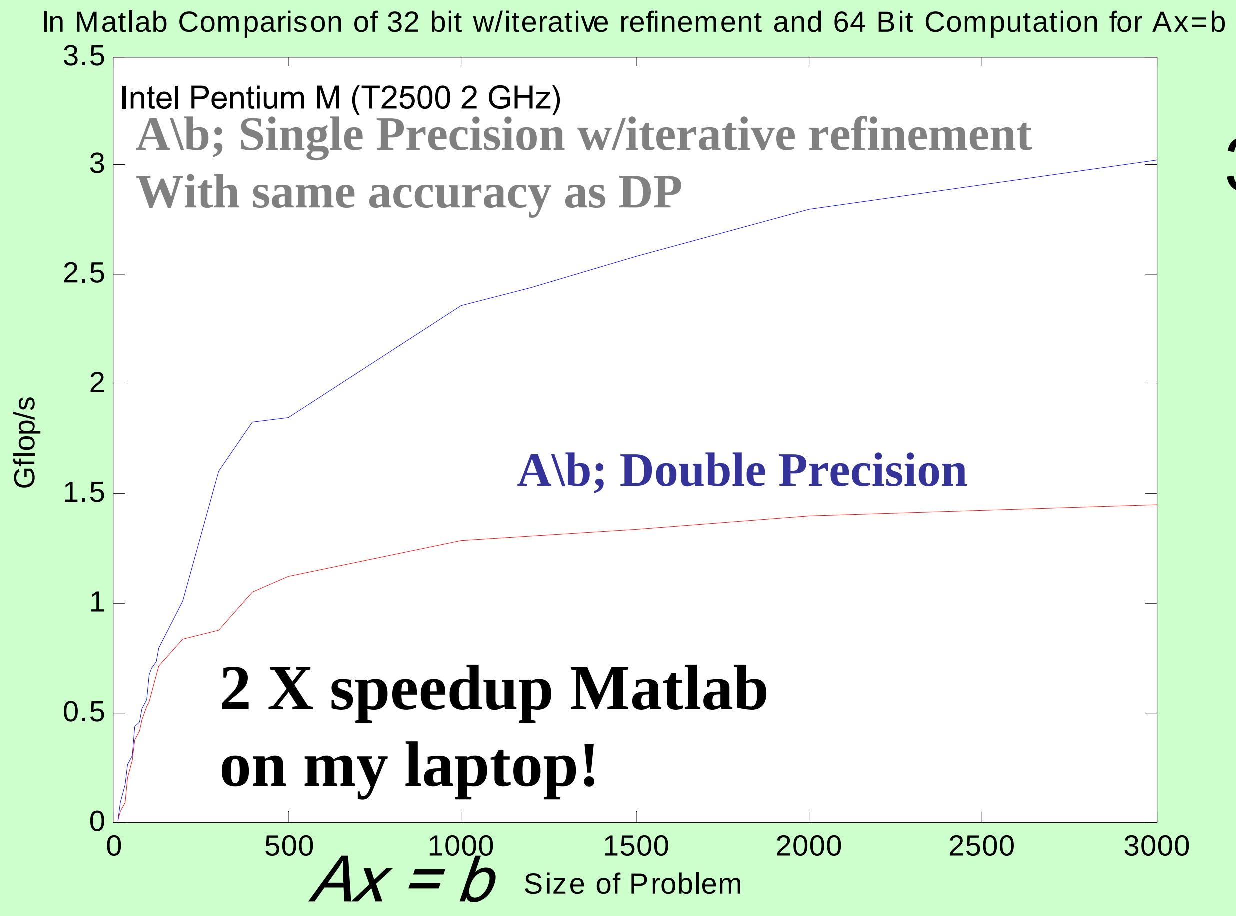
All waves treated implicitly, but still stability-limited by transport

“Partially implicit”:

Fastest waves filtered, but still stability-limited by slower waves

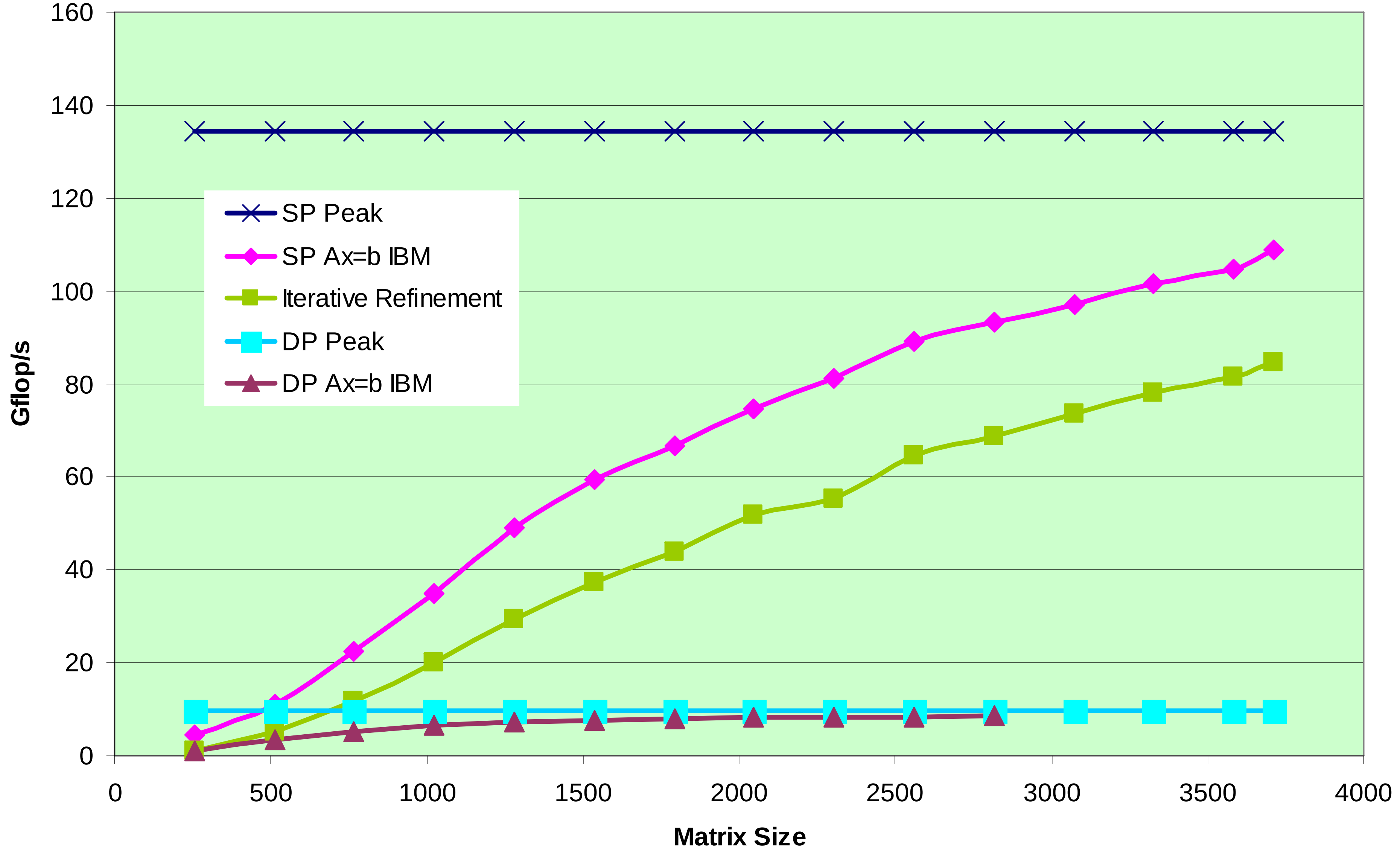
Another Look at Iterative Refinement

- On a Pentium; using SSE2, single precision can perform 4 floating point operations per cycle and in double precision 2 floating point operations per cycle.
- In addition there is reduced memory traffic (factor on sp data)



3 GFlop/s!!

IBM Cell 2.4 GHz, Ax=b Performance



Courtesy Jack Dongarra