



# **DARPA HPCS Productivity Team Benchmarking and Execution Time Working Groups**

**Robert Lucas, Ph.D.  
David Koester, Ph.D.**

**DARPA HPCS Productivity Team**

**8 November 2004  
SC2004**



# Outline



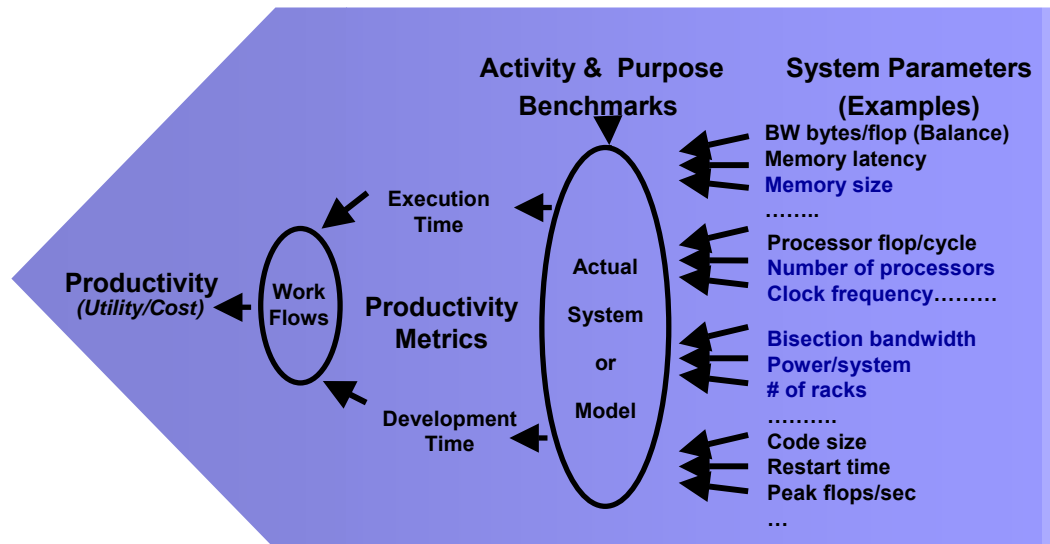
- **Benchmarking Working Group**
- **Execution Time Modeling Working Group**
  - **Quantifying HPCS/HPCchallenge Spatial/Temporal Axes**



# HPCS Benchmark Working Group Goals



- Provide the HPCS Vendors and HPCS Productivity Team the Benchmarks and Applications for
  - Scoping requirements
  - Productivity Testing
    - Execution Time Testing
    - Development Time Testing

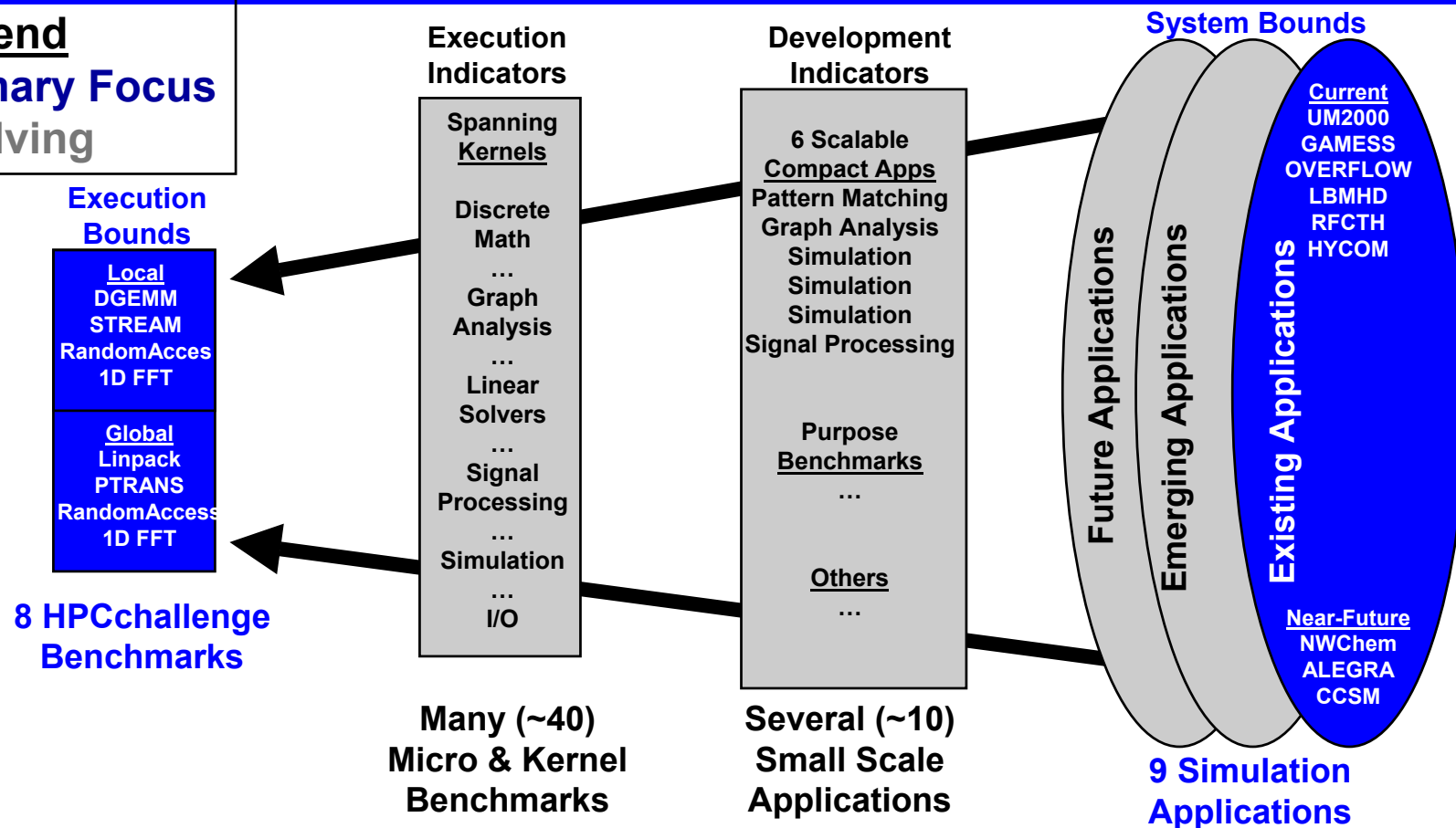




# HPCS Benchmark Spectrum



**Legend**  
**Primary Focus**  
**Evolving**

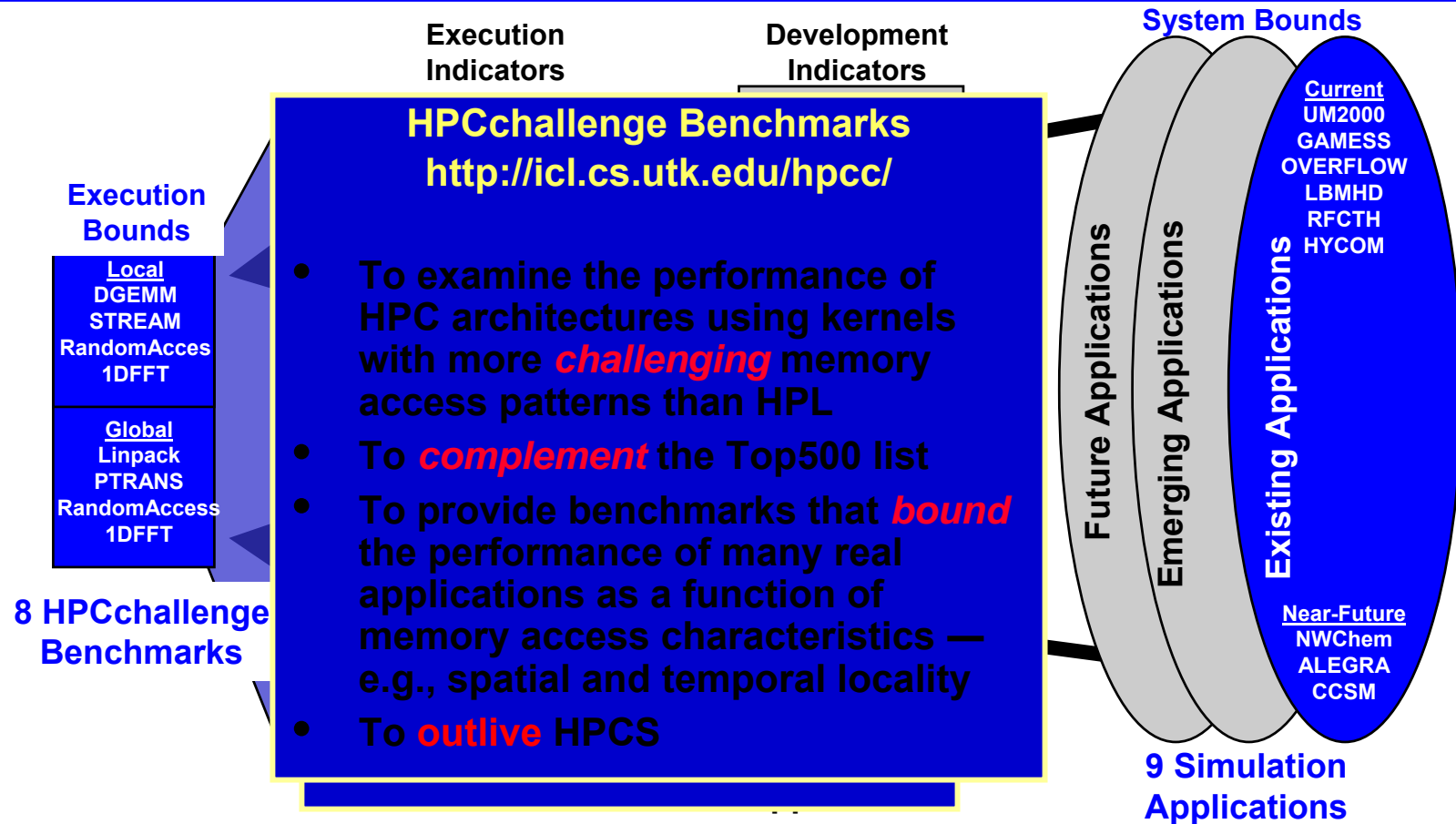


- Spectrum of benchmarks provide different views of system
  - HPCchallenge pushes spatial and temporal boundaries; sets performance bounds
  - Applications drive system issues; set legacy code performance bounds
- Kernels and Compact Apps for deeper analysis of execution and development time



# HPCS Benchmark Spectrum

## HPCchallenge Benchmarks

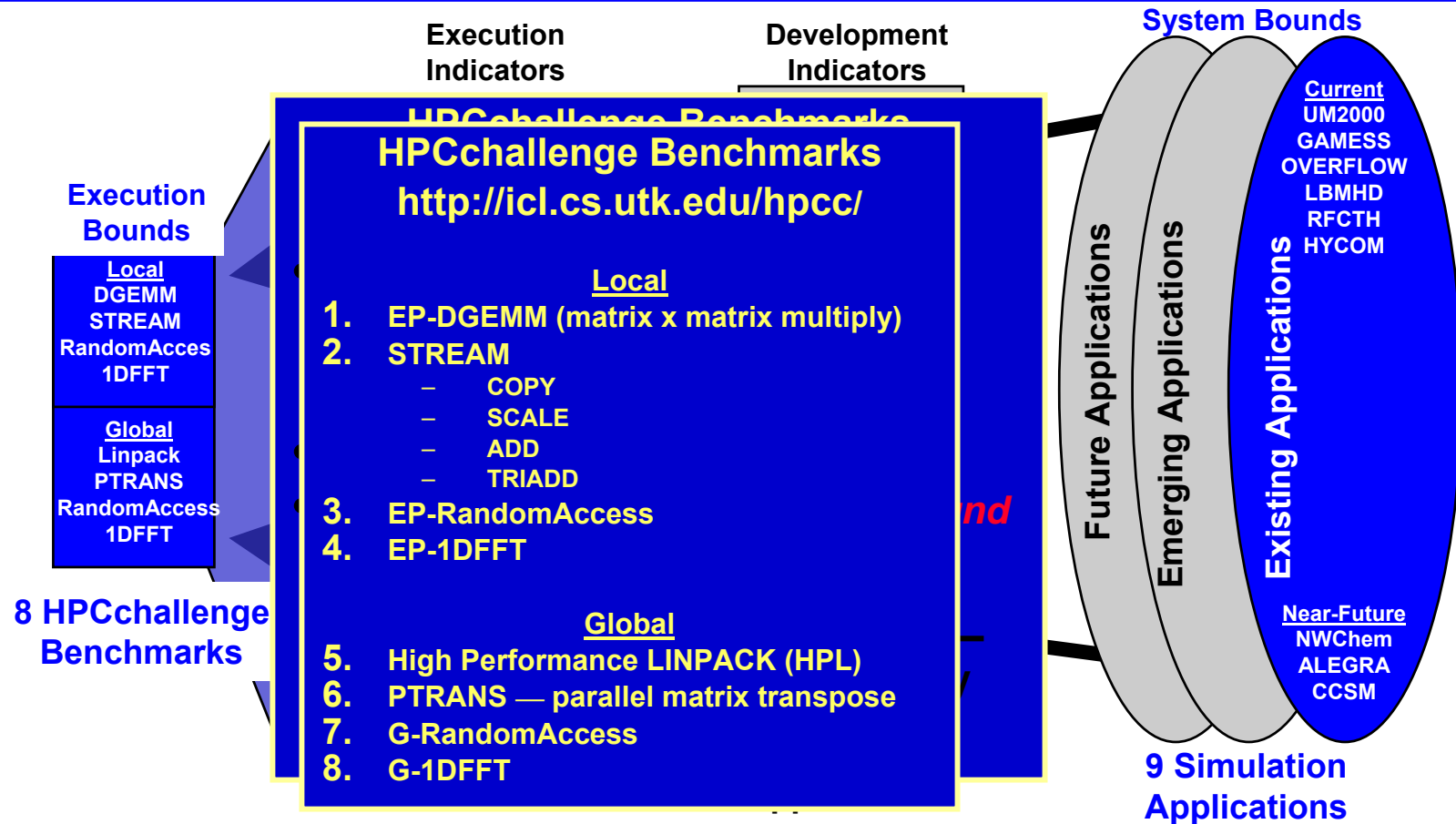


- HPCchallenge pushes spatial and temporal boundaries; sets performance bounds
- Available for download <http://icl.cs.utk.edu/hpcc/>



# HPCS Benchmark Spectrum

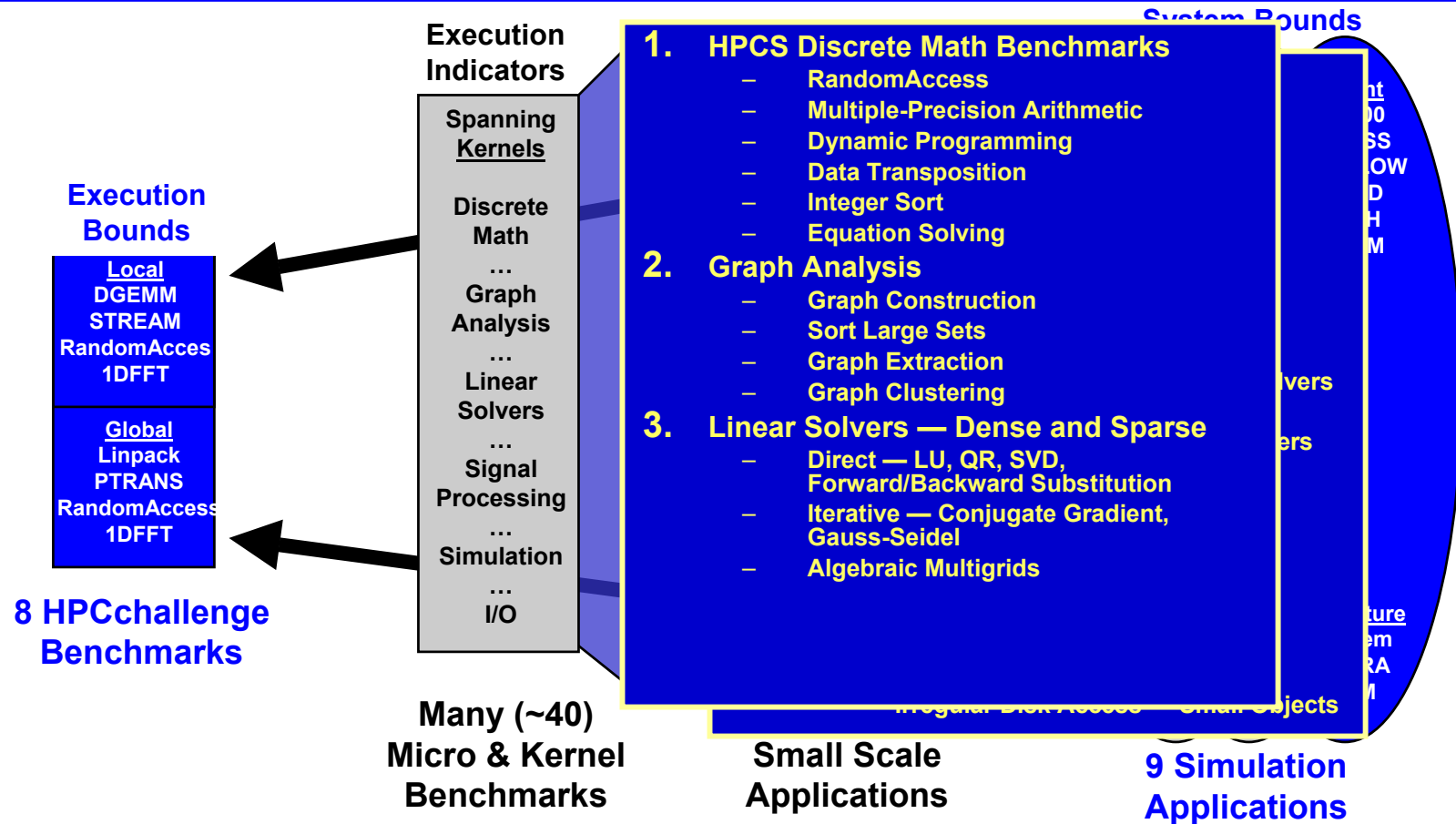
## HPCchallenge Benchmarks



- HPCchallenge pushes spatial and temporal boundaries; sets performance bounds
- Available for download <http://icl.cs.utk.edu/hpcc/>



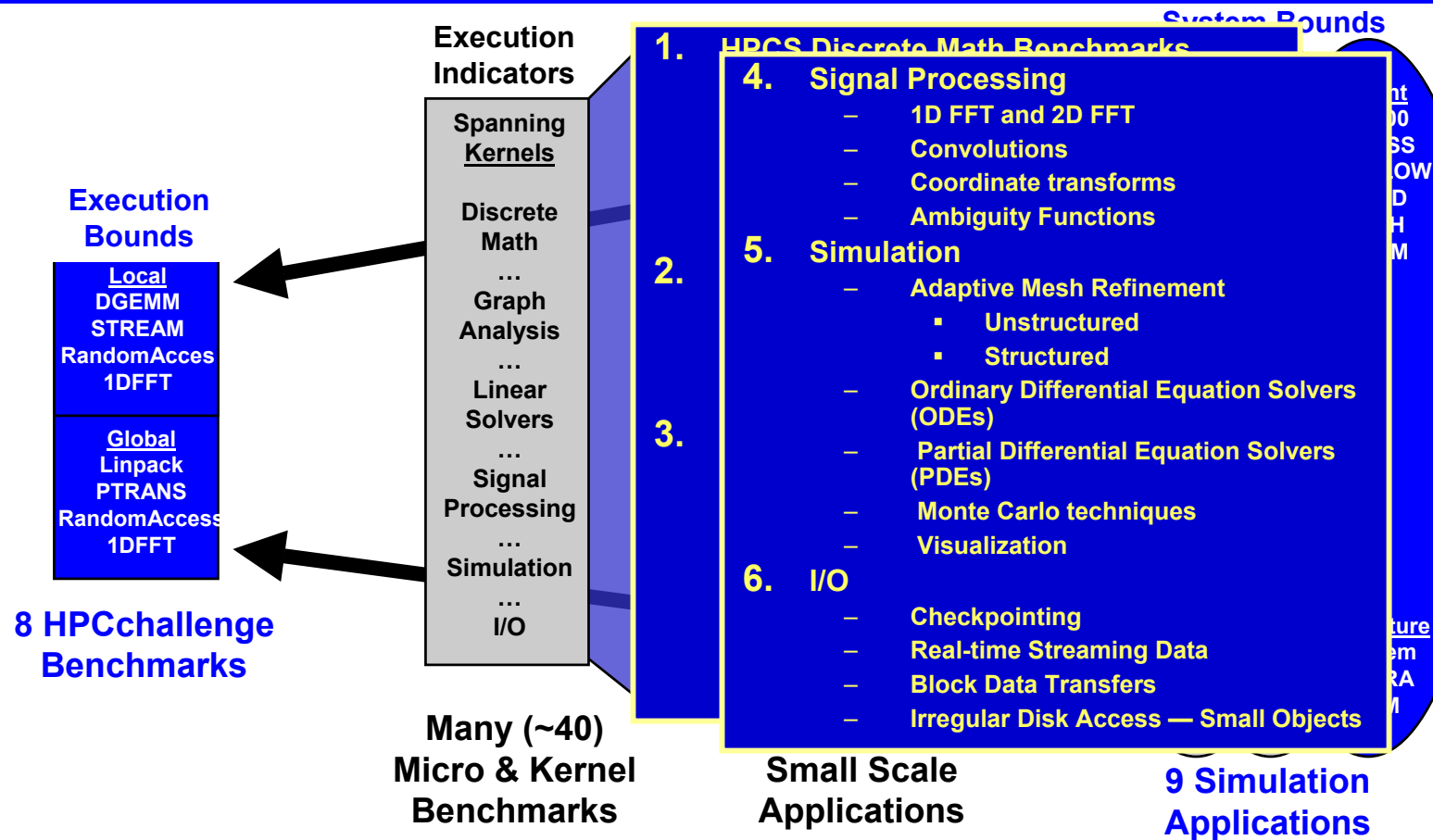
# HPCS Benchmark Spectrum Micro & Kernel Benchmarks



- Short codes that demonstrate Mission Partner computational requirements
- A spanning set of kernels from these benchmarks were used to define the HPCchallenge Benchmarks



# HPCS Benchmark Spectrum Micro & Kernel Benchmarks



- Short codes that demonstrate Mission Partner computational requirements
- A spanning set of kernels from these benchmarks were used to define the HPCchallenge Benchmarks





# Cray's "Application Kernel Matrix"



- Community repository and forum for informally comparing HPC programming languages
- Ten interesting/diverse/relevant parallel programming problems
- Participants submit solutions in their favorite languages
  - Generic solutions or tuned for performance on specific architecture
- Participants asked to log the development time they take, answer questionnaire on their programming background.
- Anecdotal data, but may suggest further systematic investigation

<http://akm.cray.com>

## Cascade: Application Kernel Matrix

[info](#) [the kernels](#) [the matrix](#) [programmer's log](#) [kernel submission form](#) [discussion forum](#)

### Kernel Specs & Solutions:

- [NASPB Conjugate Gradient](#)
- [Sweep3D](#)
- [NASPB Unstructured Adaptive](#)
- [Connected Components](#)
- [Chip Floorplan Design](#)
- [NASPB Fourier Transform](#)
- [NASPB Multigrid Benchmark](#)
- [Protein Sequence Matching](#)
- [Sparse Matrix Triangular Backsolve](#)
- [Vector Max and Prefix Sums](#)

### Links:

- [Cray, Inc.](#)
- [The Cascade Project](#)
- [HPCS: High Productivity Computing Systems program](#)
- [DARPA: Defense Advance Research Projects Agency](#)

### Contacts:

- [David Mizell](#)
- [John Feo](#)
- [John Lewis](#)
- [Brad Chamberlain](#)
- [Justin Garcia](#)

### Purpose:

Our goal is to enable site visitors to informally compare programming languages aimed at high performance computing.

### Project description:

We've chosen ten programming problems, or kernels, that are relevant to high performance computing. Programmers will design, implement and submit parallel programming solutions to these problems, in their choice of programming language. The site is set up to gather data about the relative productivity of programmers in various high performance languages.

### Background:

The Cascade Project is [Cray Inc.](#)'s project within the [High Productivity Computing Systems](#) program sponsored by the [Defense Advanced Research Projects Agency](#). This website is part of Cascade's contribution to the software productivity studies of the HPCS. A series of controlled, multiple-subject programming time experiments is also taking place within the HPCS program, under the direction of [Prof. Victor Basili](#) of the University of Maryland. The Application Kernel Matrix provides anecdotal information about programming languages, significant anomalies in which might suggest further formal experiments.

### Using the site:

- [Learn](#) about the programming problems we've chosen, and examine our sequential solutions to the problems.
- [Browse](#) the matrix of solutions submitted by others.
- [Log](#) your time while working on a solution to one of the problems.
- [Submit](#) your own solution and the time it took you to design, code, debug and performance-tune it. You'll also be prompted to fill in information about your programming experience.
- [Discuss](#) the kernels or your favorite programming language in the forum.

Last Modified: October 26 2004 04:52:47 PM

[W3C](#) [HTML](#) [W3C](#) [CSS](#)



# HPCS Benchmark Spectrum Small Scale Applications



- **Scalable Synthetic Compact Applications**

- Medium size scalable applications connecting several important kernels in a “real” context
- Bridge the gap between scalable synthetic kernel benchmarks and (non-scalable) real applications
- Representative of actual workloads within an application while not being numerically rigorous
  - memory access characteristics
  - communications characteristics
  - I/O characteristics
  - etc.
- No limits on the distribution to vendors and universities

**#1 Optimal Pattern Matching**

**#2 Graph Theory**

**#3–#5 Simulation**

**#6 Signal and Image Processing and Knowledge Formation**

## Development Indicators

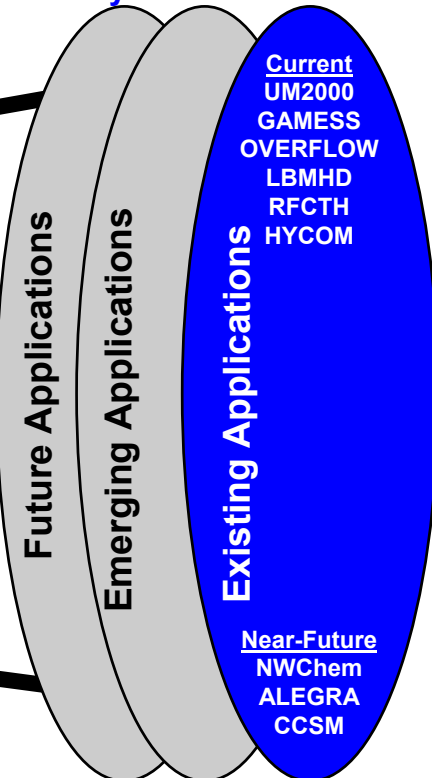
6 Scalable Compact Apps  
Pattern Matching  
Graph Analysis  
Simulation  
Simulation  
Simulation  
Signal Processing

Purpose Benchmarks  
...

Others  
...

**Several (~10)  
Small Scale  
Applications**

## System Bounds



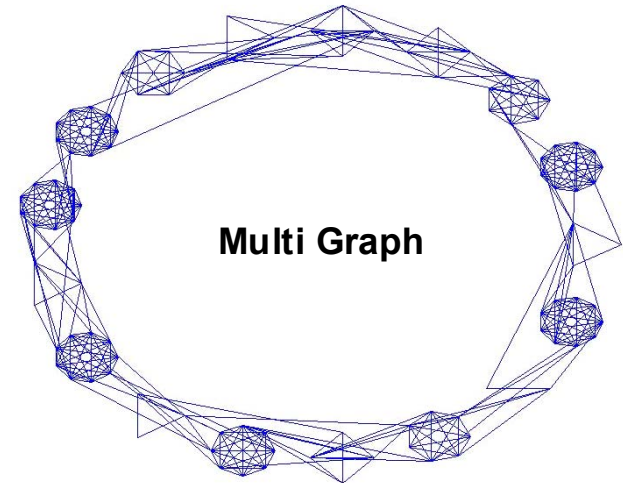
**Current**  
UM2000  
GAMESS  
OVERFLOW  
LBMHD  
RFCTH  
HYCOM

**Near-Future**  
NWChem  
ALEGRA  
CCSM

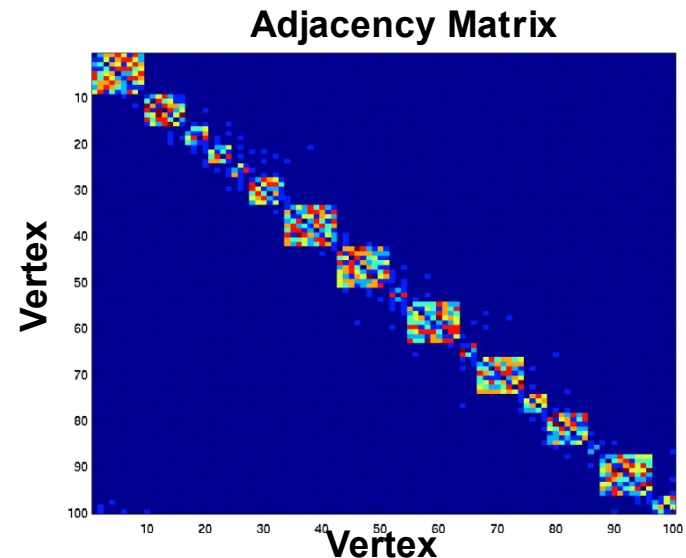
**9 Simulation  
Applications**

- Important for development experiments — small enough to measure productivity
- Multiple kernels to stress hardware architecture

- **Develop a scalable synthetic compact application that has multiple kernels accessing a single data structure representing a directed asymmetric weighted multigraph with no self loops**
  - Describe a Mission Partner requirement
  - Basis for development time experiments
- **Scalable data generator**
- **Four computational kernels**
  - Kernel 1 — Graph Construction
  - Kernel 2 — Sort on Selected Edge Weights
  - Kernel 3 — Extract Subgraphs
  - Kernel 4 — Partition Graph Using a Clustering Algorithm
- **Each kernel will require irregular access to the graph's data structures**
- **No single data layout will be optimal for all computational kernels**
- **To be entirely integer and character based**
  - Except for statistics



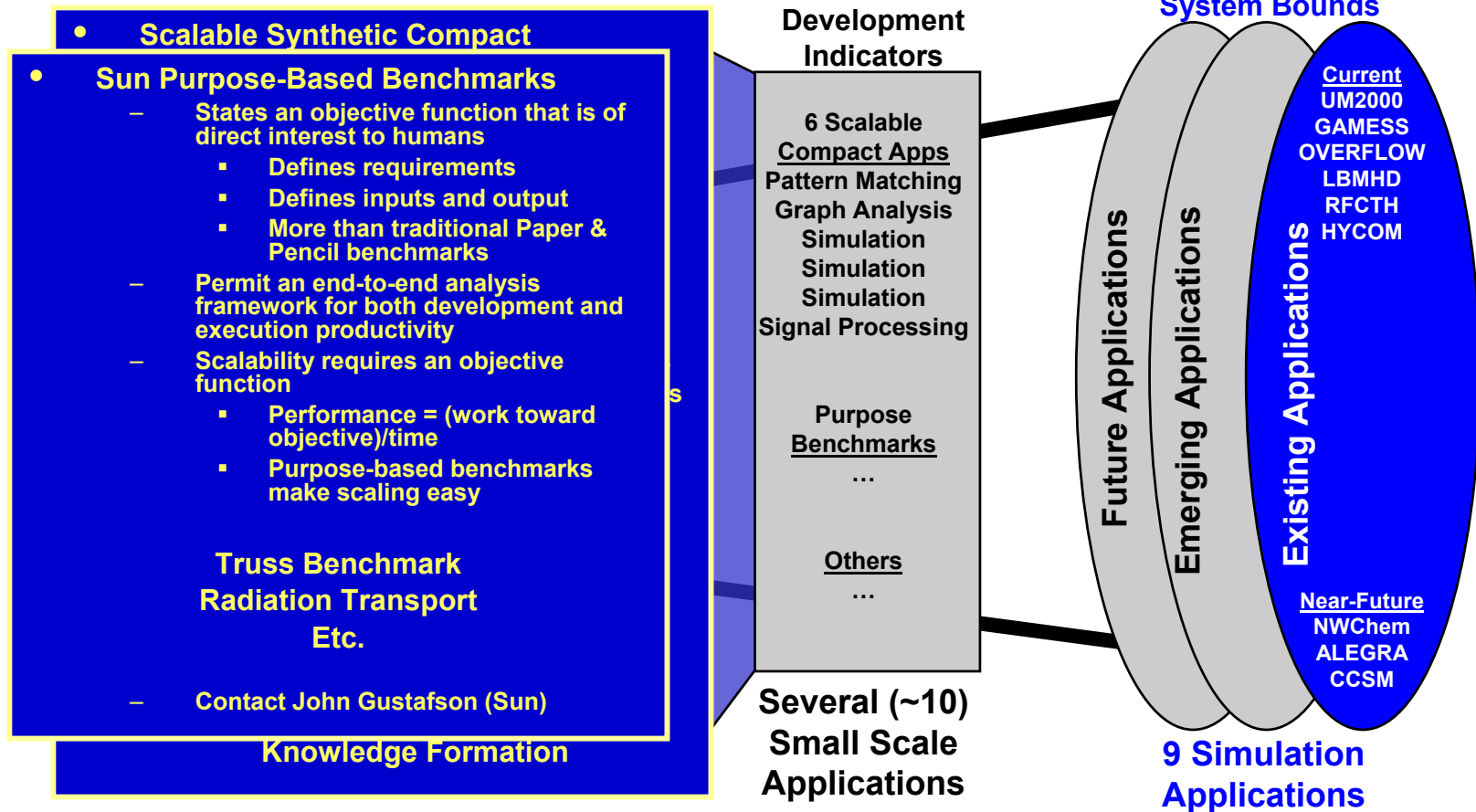
**Multi Graph**



**Adjacency Matrix**

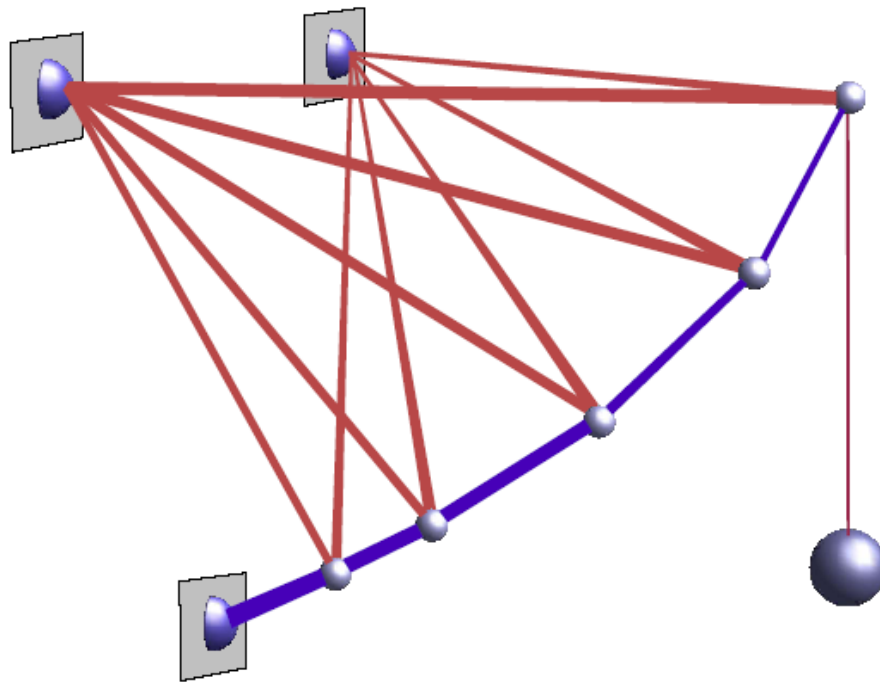


# HPCS Benchmark Spectrum Small Scale Applications



- Important for development experiments — small enough to measure productivity
- Multiple kernels to stress hardware architecture

## Truss Benchmark Example

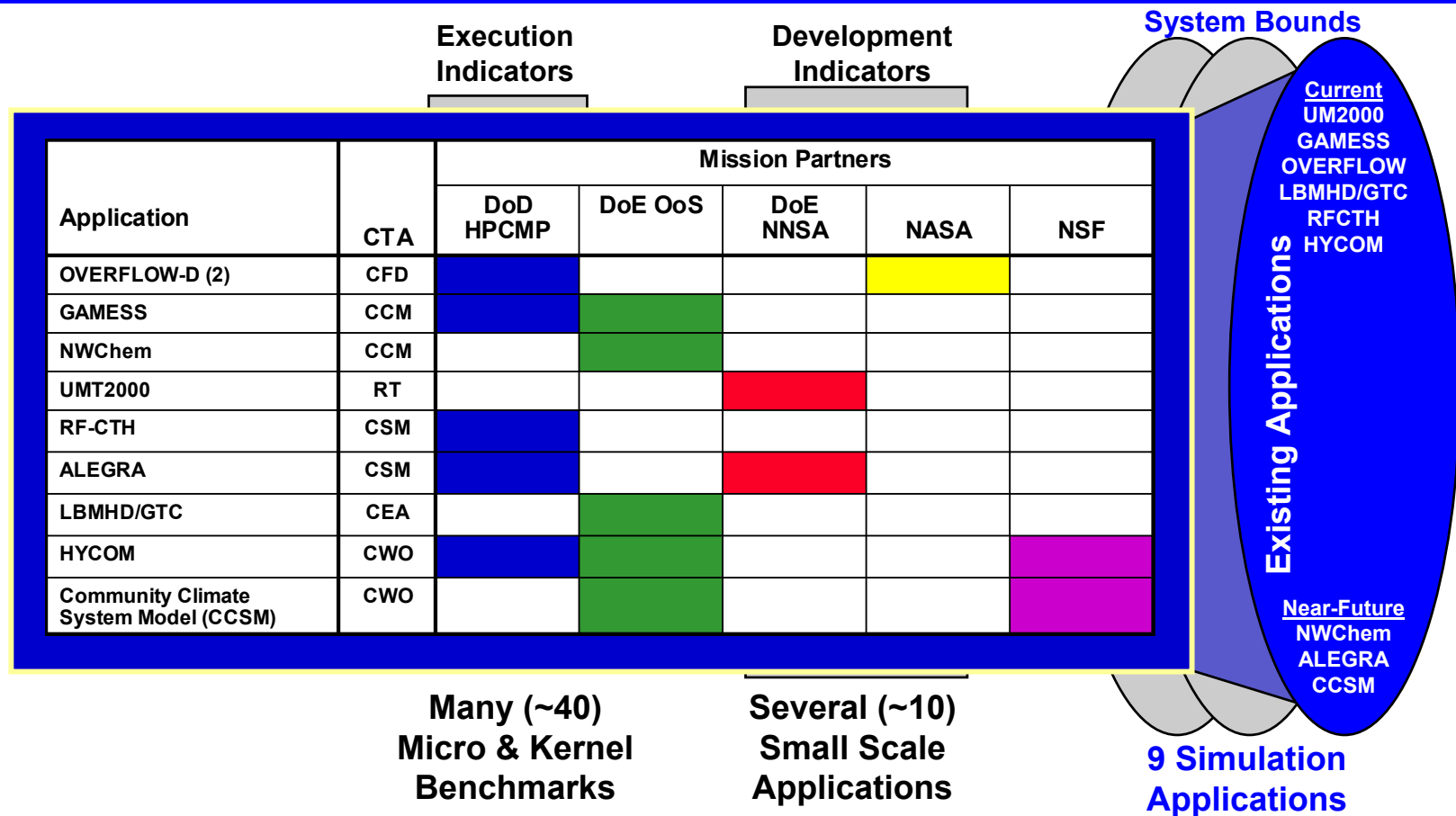


- Given set of support points, minimize the total weight of the structure that can support a given load.
- Multiple optimization problems
  - Shape optimization
  - Geometry optimization
  - Topology optimization





# HPCS Benchmark Spectrum Representative Applications



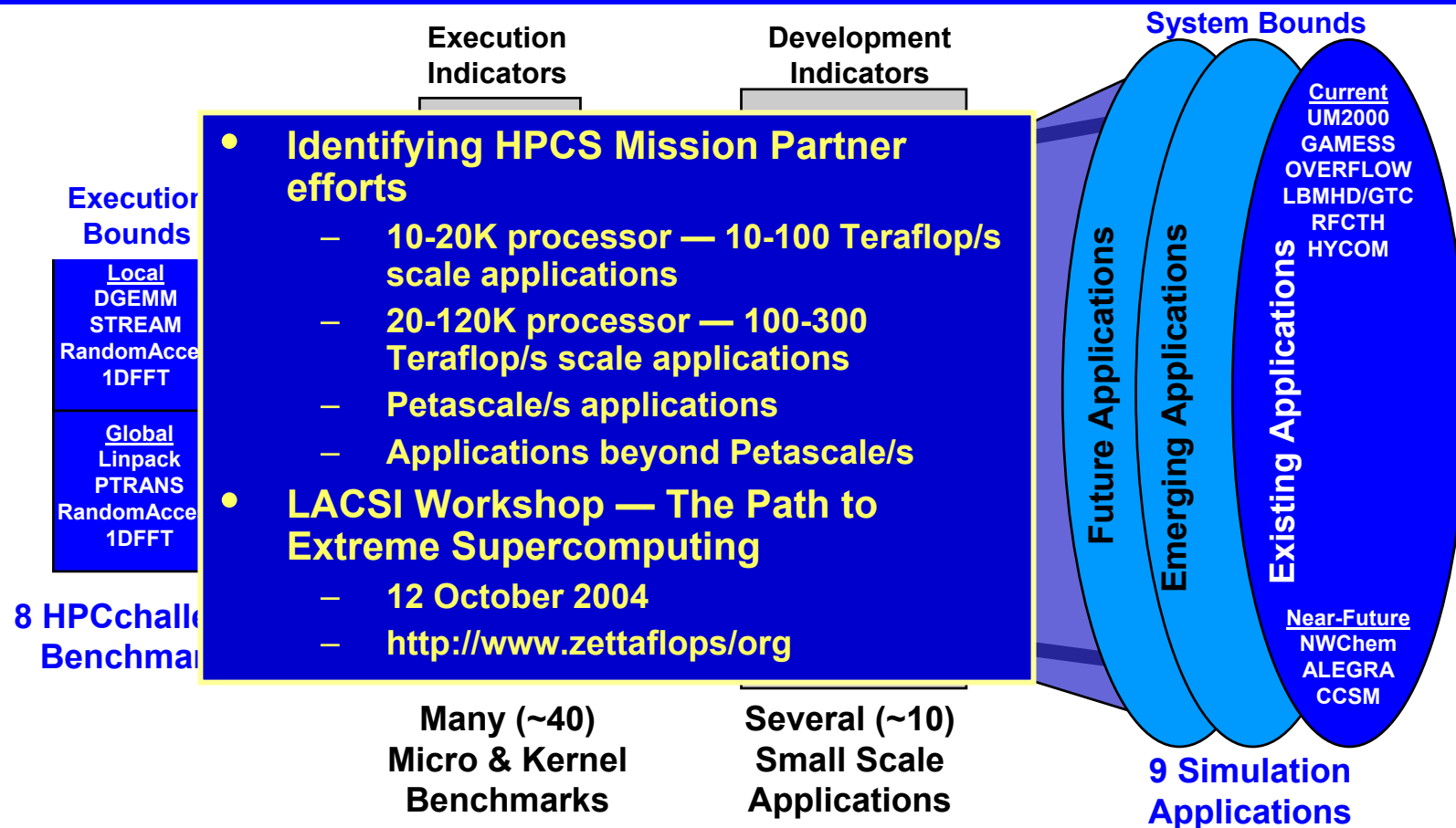
- Full application codes that demonstrate the scale of Mission Partner computational requirements
- For system analysis





# HPCS Benchmark Spectrum

## Future and Emerging Applications



- Scope out emerging and future applications for 2010
- What applications will be important in 2010?



# Coming soon ... HPCS I/O Challenges



- **1 Trillion files in a single file system**
  - 32K file creates per second
- **10K metadata operations per second**
  - Needed for Checkpoint/Restart files
- **Streaming I/O at 30 GB/sec full duplex**
  - Needed for data capture
- **Support for 30K nodes**
  - Future file system need low latency communication

**An envelope on HPCS Mission Partner requirements**

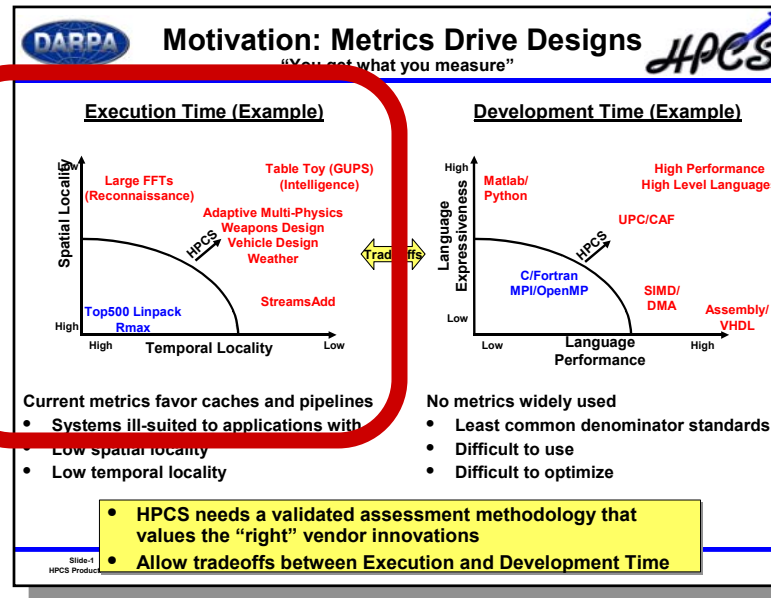




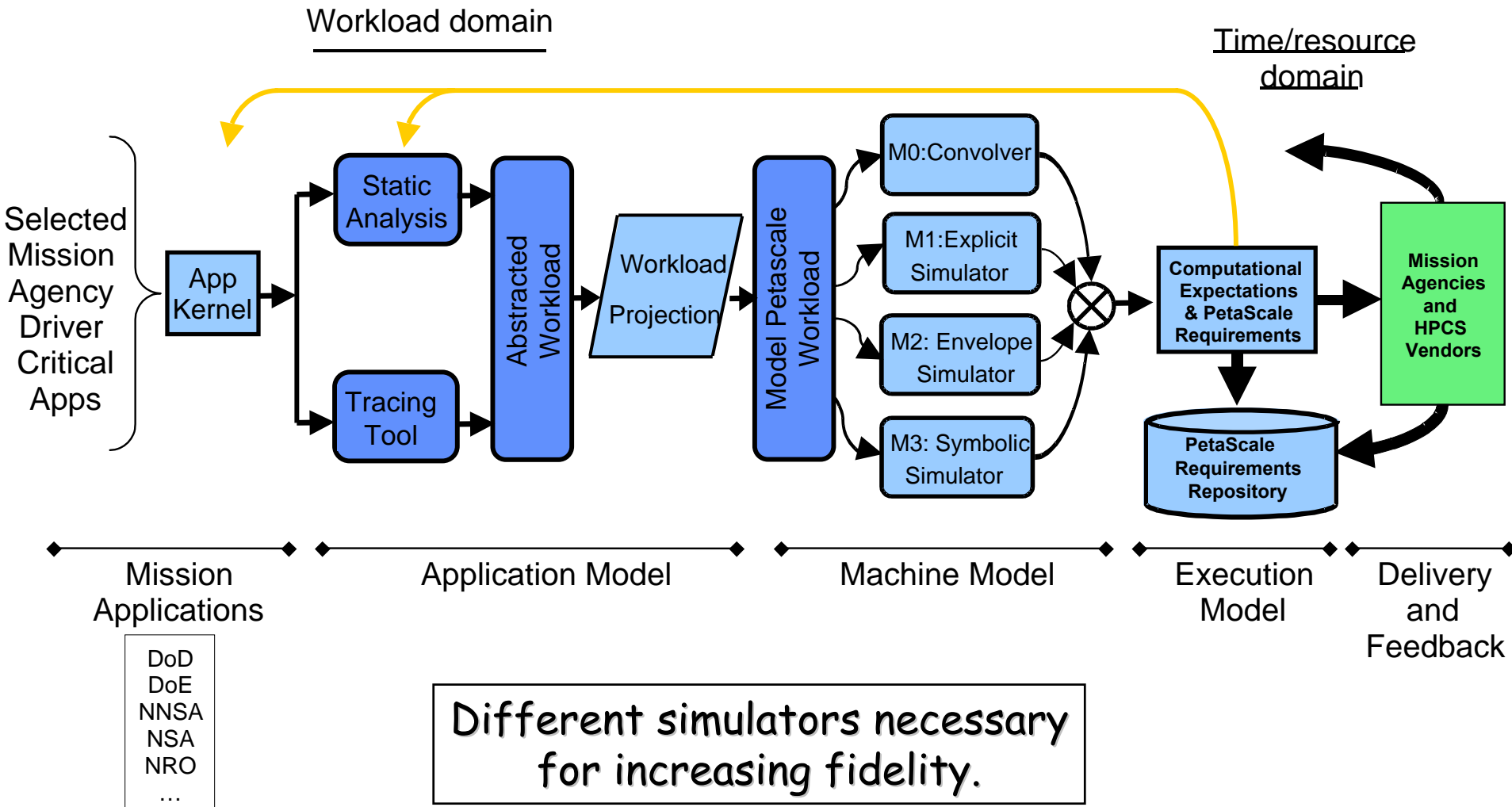
# Outline



- Benchmarking Working Group
- Execution Time Modeling Working Group
  - Quantifying HPCS/HPCchallenge Spatial/Temporal Axes



- Create models of 2010 Petascale application workloads
- Develop application-driven system requirements based on these workloads
- Explore architecture sensitivities for maximizing Petascale Execution Time performance





# Outline

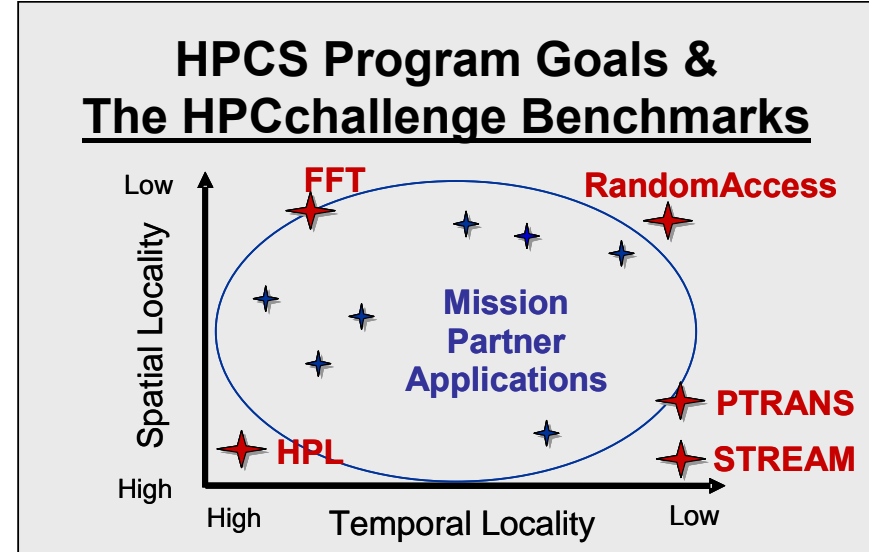


- **Benchmarking Working Group**
- **Execution Time Modeling Working Group**
  - **Quantifying HPCS/HPCchallenge Spatial/Temporal Axes**

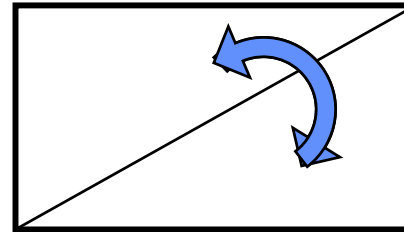
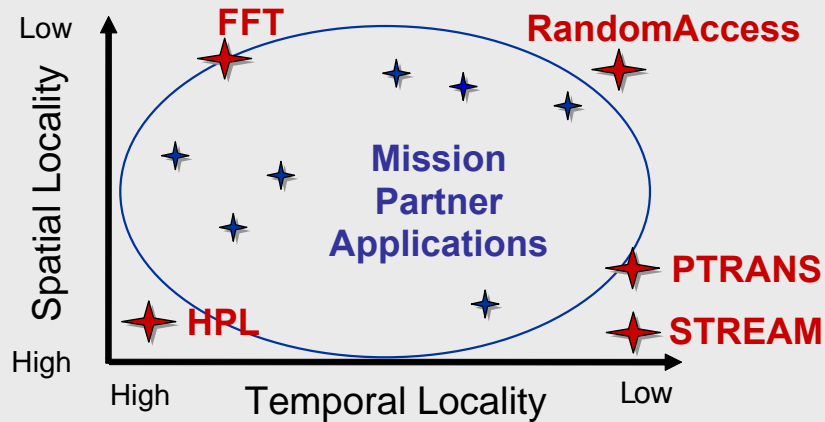
- Current focus of execution time people
- Work in progress
- Comments and feedback welcomed



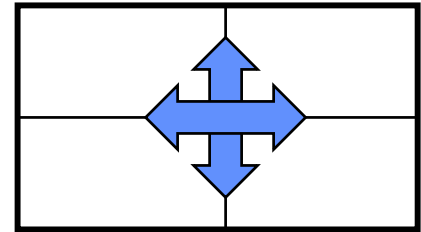
- Near term goals:
  - Define the axes
  - Add the implied “z” axis
  - Locate HPC challenge
  - Locate DOD applications



## HPCS Productivity Design Points

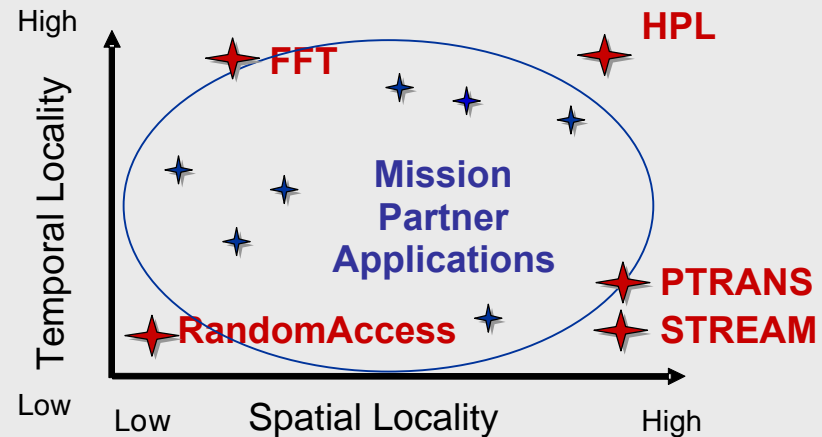


1. Switch Axes



2. Invert Ranges

## HPCS Productivity Design Points





# Locality Definitions

(adapted from Hennessy and Patterson<sup>‡</sup>) *HPCS*

- **Temporal locality** — “recently accessed items are likely to be accessed in the near future” (p.47)
  - “...tells us that we are likely to need this word again in the near future, so it is useful to place it in the cache where it can be accessed quickly” (p. 393)
- **Spatial locality** — “items whose addresses are near one another tend to be referenced close together in time” (p.47)
  - “...there is a high probability that the other data in the block will be needed soon” (p. 393)

<sup>‡</sup> Computer Architecture – A Quantitative Approach, 3<sup>rd</sup> Ed.,  
John L Hennessy & David A. Patterson



# Data reuse and locality

(adapted from Wolf & Lam, PLDI 1991)



- **Data reuse:**
  - a data item is reused if it is used multiple times in a computation
  - reuse is an *inherent* property of the computation
- **Data locality:**
  - data remains in the memory hierarchy level of interest between reuses
  - *reuse does not guarantee locality*
- **Locality analysis:**
  - mathematical framework for identifying and quantifying reuse in loop nests.





# Types of Reuse

(adapted from Wolf & Lam, PLDI 1991)



- ***Self-Temporal:***
  - reuse: a reference within a loop accesses the same data in different iterations
  - locality: data remains in the memory hierarchy level of interest (registers, caches) between reuses
- ***Self-Spatial:***
  - reuse: a reference accesses data items in close-by memory locations in different iterations
  - locality: data present in the memory hierarchy level of interest due to a previous reference to a close-by memory location
- ***Group-temporal, group-spatial reuse:***
  - distinct references access same or close-by locations

## LBL Apex-Map

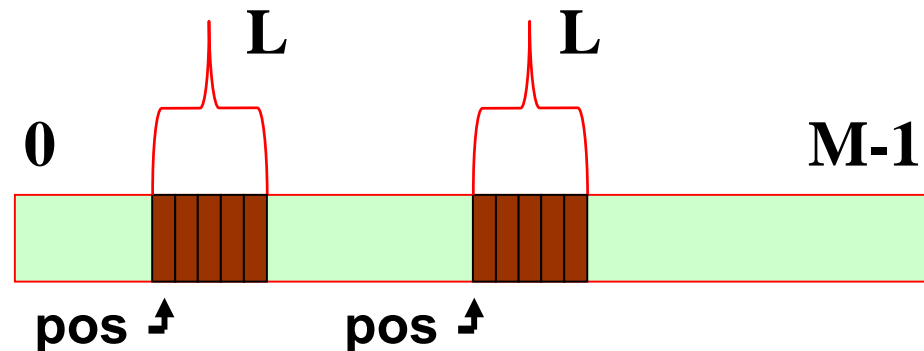
Invented by Erich Strohmaier

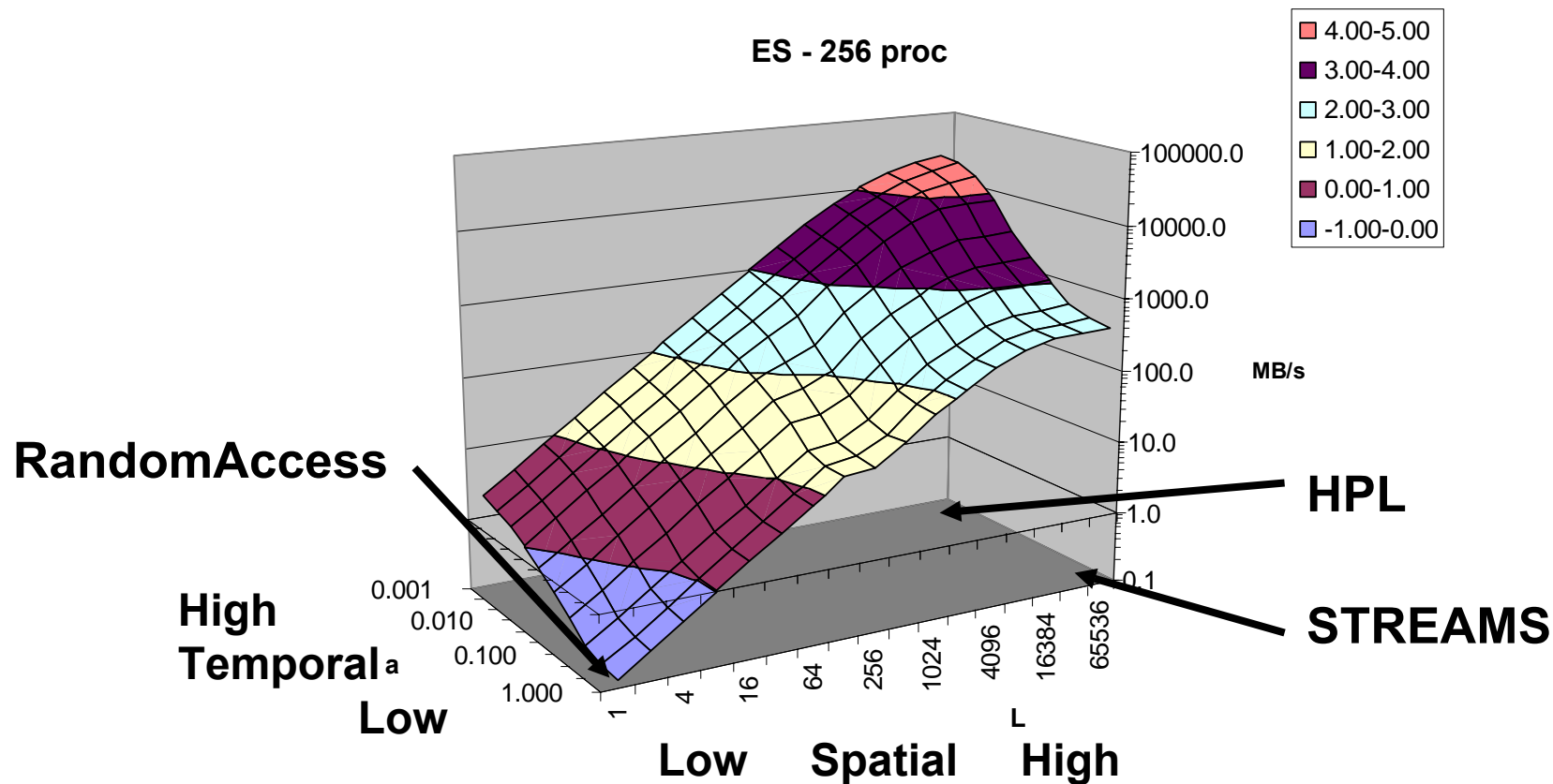
Covers the spatial/temporal space

```
for ( i = 0; i < N; i++) {
  initIndexArray(i);
  for (j = 0; j < I/4; j++)
    pos = ind[j*4];
    ...
    for (k = 0; k < L; k++) {
      res0 += data[pos + k];
      ...
    }
  }
}
```

pick a random address

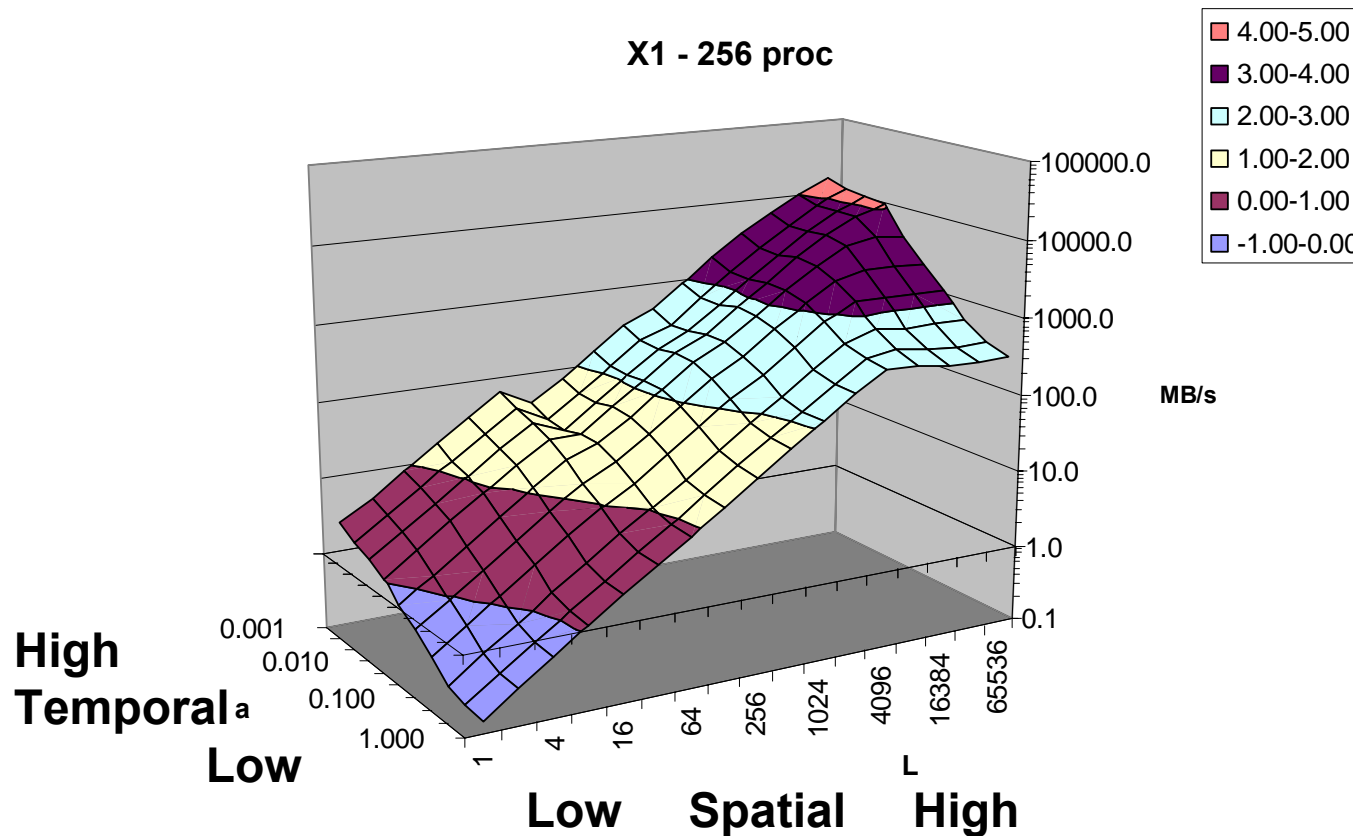
fetch a block of data





Earth Simulator

Data from Erich Strohmaier (LBNL APEX-Map)



**Cray X-1**

**Data from Erich Strohmaier (LBNL APEX-Map)**

**How could one quantify the spatial and temporal locality in a real code?**

$$\text{SpatialScore}(N) = \left( \sum_{i=1}^N (\text{Refs Stride } i / i) \right) / \text{Total Refs}$$

$$\text{TemporalScore}(N) = \text{Observed Reuse} / (\text{Total Refs} - \text{Spatial Refs})$$

**It's harder than it looks!!!**

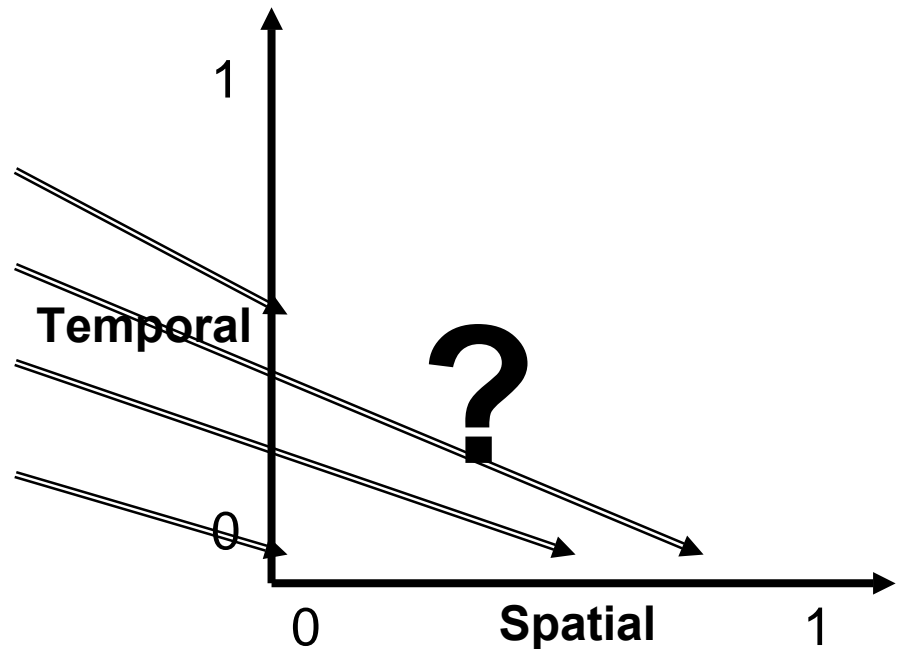
```
for ( i = 0; i < N; i++) {
    add = random_number;
    table[add] ^= random_number;
}
```

**Load + Store (temporal)**

**Two loads + Store**

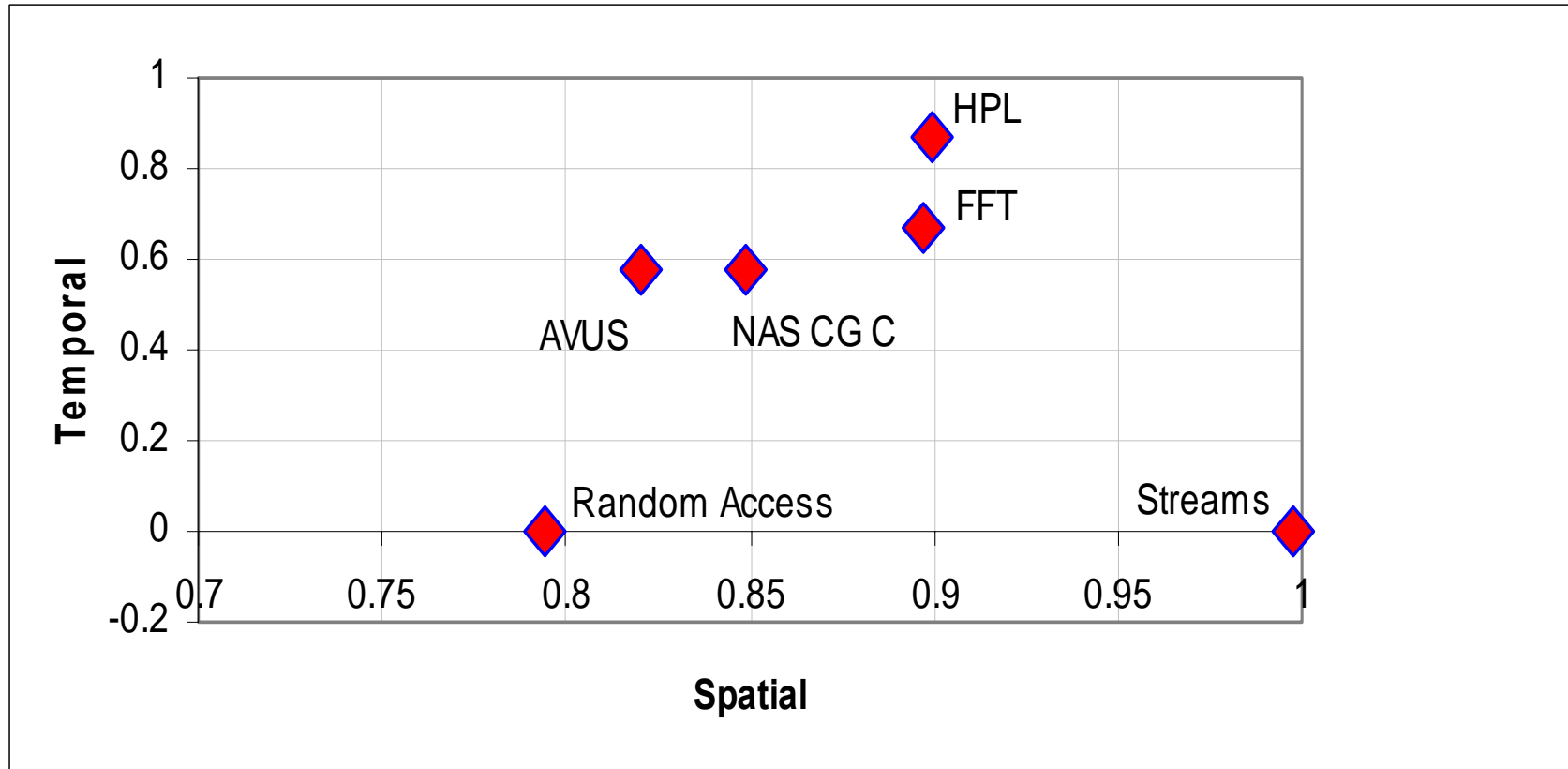
**Load + Store (spatial)**

**Update (design goal)**

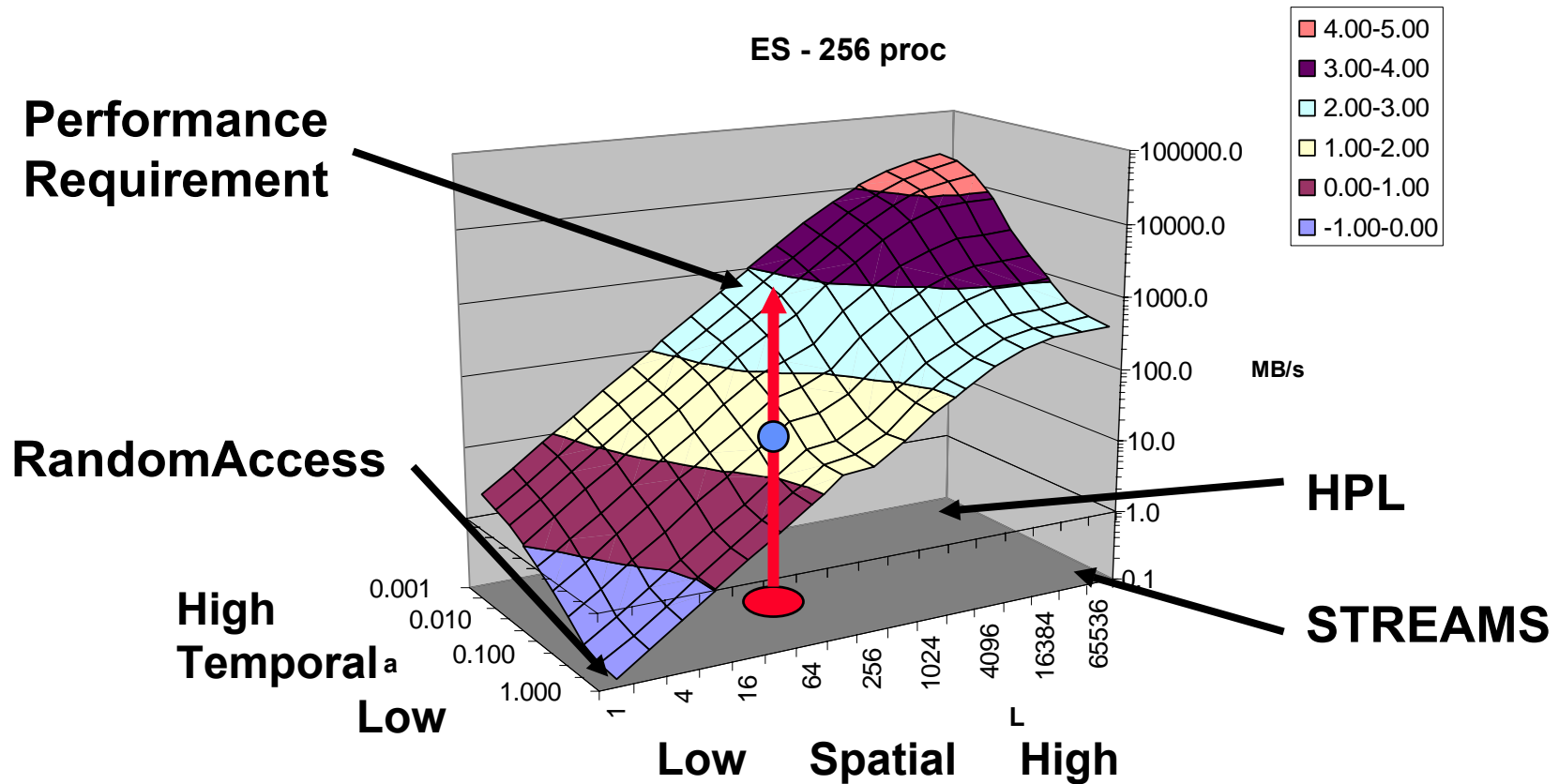




# HPC Challenge Benchmarks on axes of spatial and temporal locality



Data from Allan Snively (SDSC PMaC Project)







# Summary



- **Benchmarks**
  - Kernels identified and bounds provided by HPCchallenge
  - Scalable Compact Apps. identified and under construction
  - Mission partner apps. identified
- **Execution Time**
  - Warming up ... focused on understanding spatial/temporal locality space
  - Working on connecting compiler analysis to tracing tools
  - Developing envelope and convolver machine models

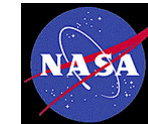


# HPCS Productivity Team Benchmark Working Group Contributors



- David Koester (MITRE) — Group Lead
- Jeremy Kepner (MIT LL)
- Bob Lucas (USC/ISI)
- Dolores Shaffer (STA)
- David A. Bader (UNM, IBM)
- David Mizell (Cray)
- Piotr Luszczek (ICL/UT)
- Jeffrey Vetter (ORNL)
- Dave Bailey (LBL)
- Jack Dongarra (ICL/UT)
- Larry Davis (DoD HPCMP)
- Allan Snaveley (SDSC)
- Henry Newman (Instrumental)
- John A Gunnels (IBM)
- Doug Post (LANL)
- Ram Rajamony (IBM)
- Tarek El-Ghazawi (GWU)
- Larry Votta (Sun)
- Theresa Meues (MIT LL)
- Bill Mann (MIT LL)
- Jeff Carver (UMD)
- And Many Others!

MITRE



MITRE

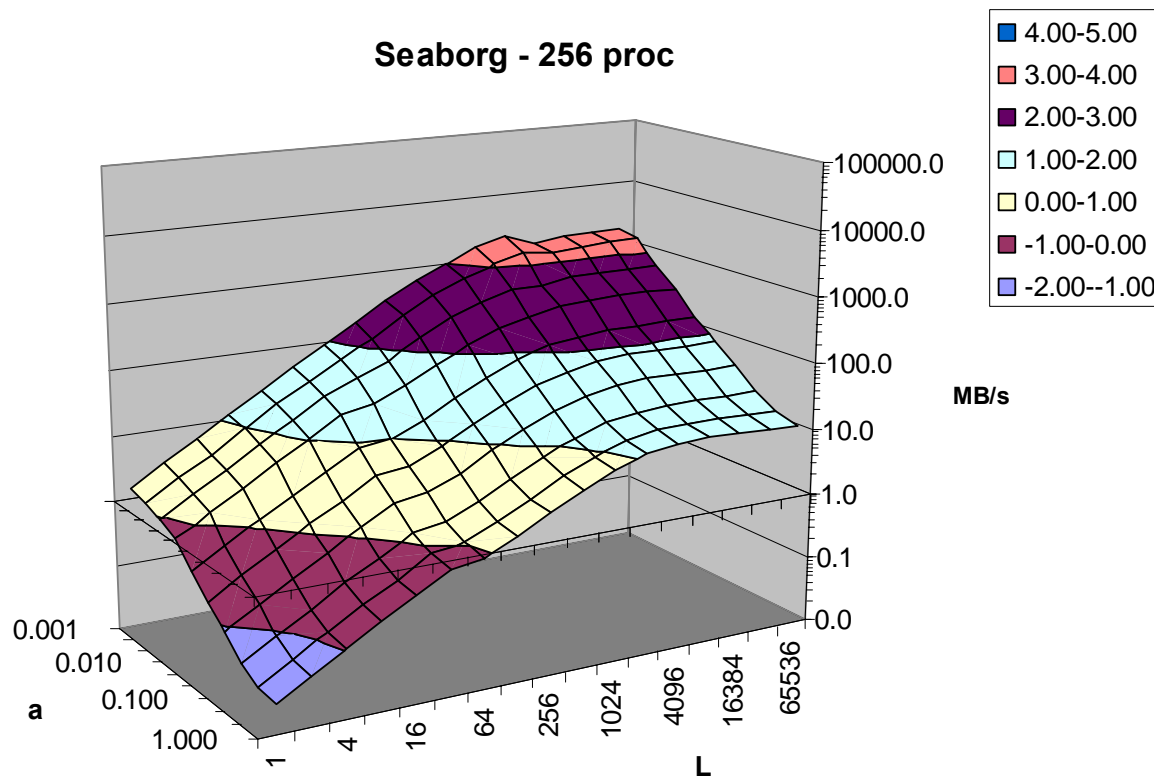
MIT Lincoln Laboratory

ISI



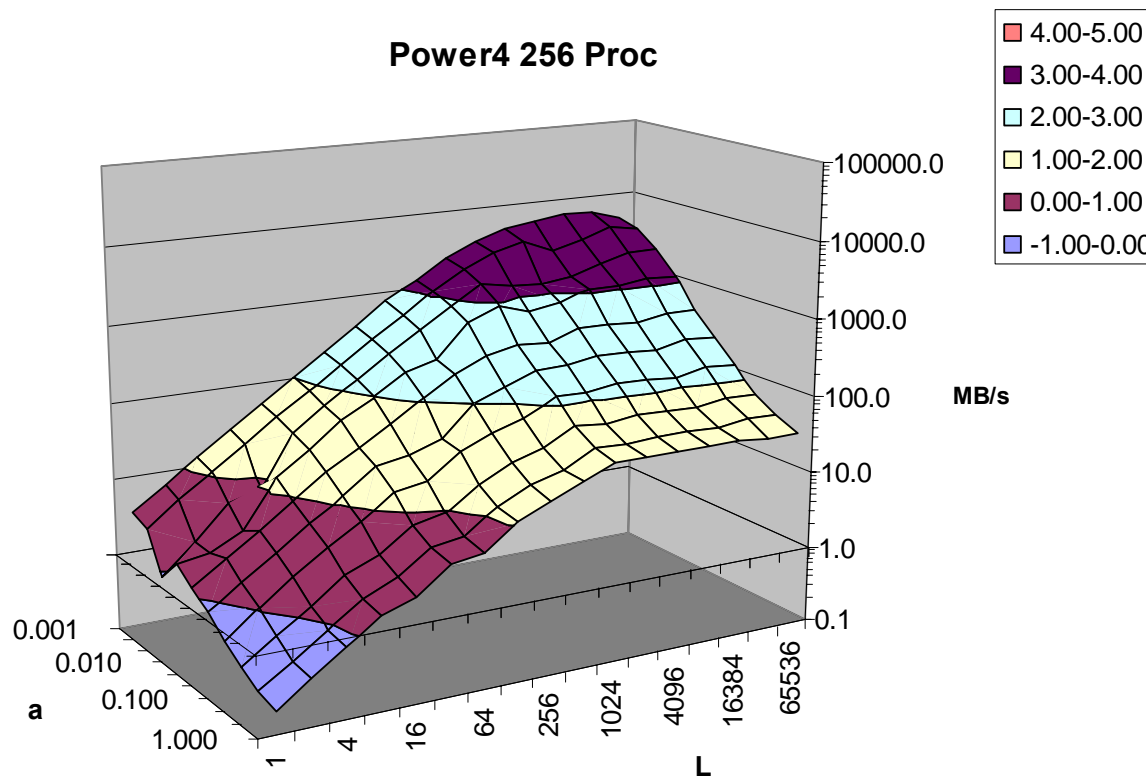
# Backup Slides





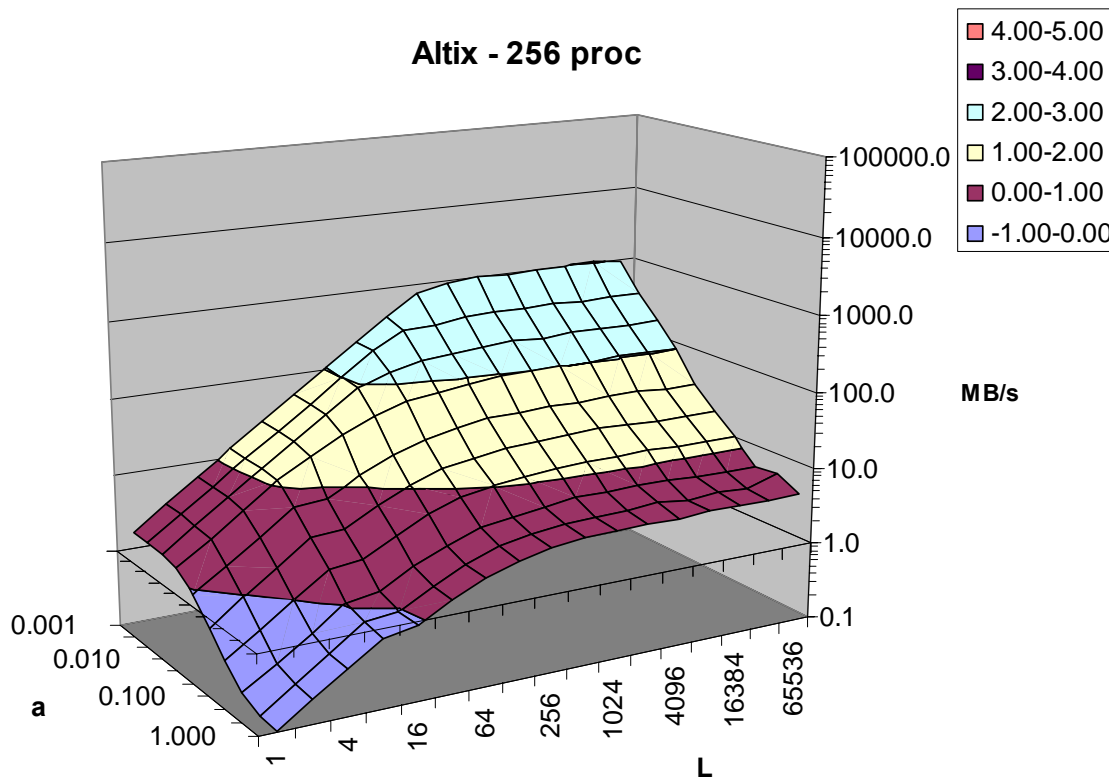
**IBM Power 3 SP**

**Data from Erich Strohmaier (LBNL APEX-Map)**



IBM Power 4 SP

Data from Erich Strohmaier (LBNL APEX-Map)



**SGI Altix**

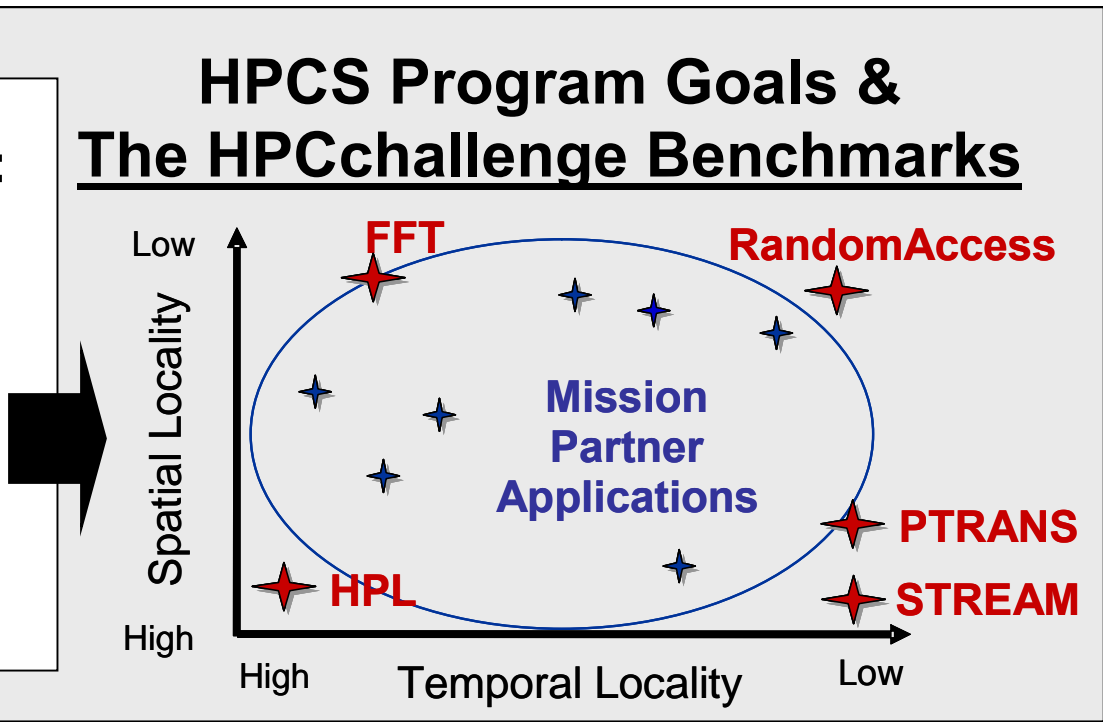
**Data from Erich Strohmaier (LBNL APEX-Map)**



# HPCS Program Goals and the HPCchallenge Benchmarks



- General purpose architecture capable of:  
Subsystem Performance Indicators
  - 1) 2+ PF/s LINPACK
  - 2) 6.5 PB/sec data STREAM bandwidth
  - 3) 3.2 PB/sec bisection bandwidth
  - 4) 64,000 GUPS



- If we use this slide, we need to fix the box on the right to match the slides used later
- Possibly, a morph from this slide to the way you will discuss this later



# HPCS Kernel Benchmarks



## 1. HPCS Discrete Math Benchmarks

- RandomAccess
- Multiple-Precision Arithmetic
- Dynamic Programming
- Data Transposition
- Integer Sort
- Equation Solving

## 2. Graph Analysis

- Graph Construction
- Sort Large Sets
- Graph Extraction
- Graph Clustering

## 3. Linear Solvers — Dense and Sparse

- Direct — LU, QR, SVD, Forward/Backward Substitution
- Iterative — Conjugate Gradient, Gauss-Seidel
- Algebraic Multigrids

## 4. Signal Processing

- 1D FFT and 2D FFT
- Convolutions
- Coordinate transforms
- Ambiguity Functions

## 5. Simulation

- Adaptive Mesh Refinement
  - Unstructured
  - Structured
- Ordinary Differential Equation Solvers (ODEs)
- Partial Differential Equation Solvers (PDEs)
- Monte Carlo techniques
- Visualization

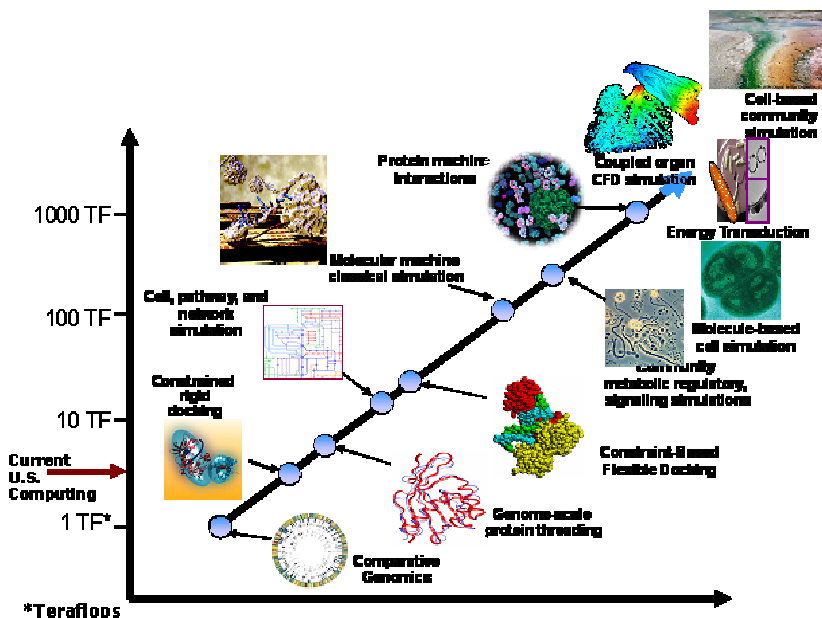
## 6. I/O

- Checkpointing
- Real-time Streaming Data
- Block Data Transfers
- Irregular Disk Access — Small Objects

- An alternative way to present this detail



- Recent reports have outlined the need for application-driven systems requirements
  - Scales, June 2003
  - HECRTF, June 2003
- New application algorithms and implementations at PetaFlop/s scale unknown to us
- However, existing applications will be run at Petascale for example:
  - Weapons design
  - Crash analysis



## How could one Quantify the Spatial and Temporal Locality in a Real Code?

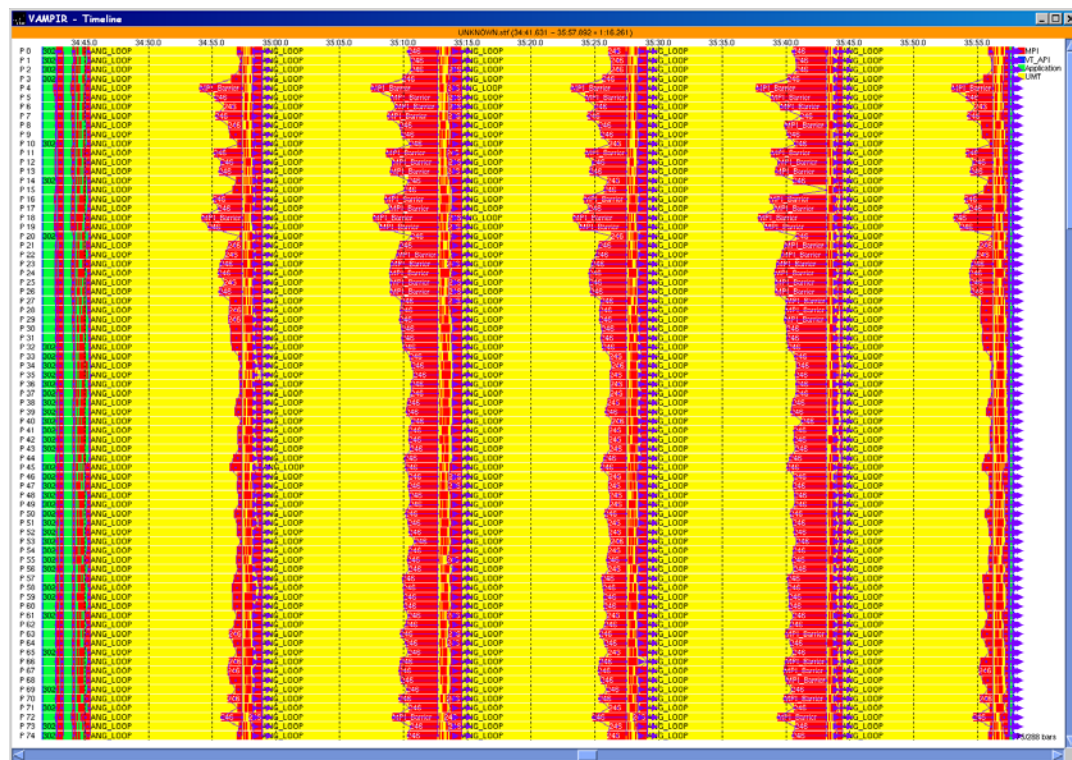
$$\text{SpatialScore}(N) = \sum_{i=1}^N (\text{Refs Stride } i / i) / \text{Total Refs}$$

$$\text{TemporalScore}(N) = \text{Measured Hit Rate} - \text{SpatialScore}(n)$$

$$\text{TemporalScore}(N) = \text{Observed Reuse} / (\text{Total Refs} - \text{Spatial Refs})$$



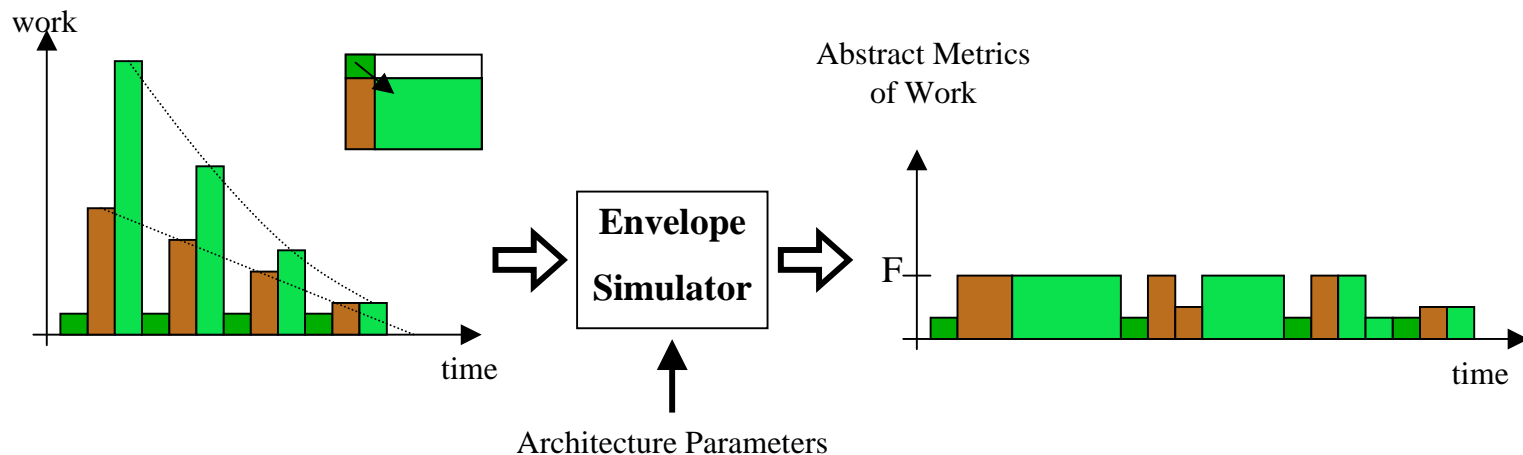
- **Compiler Analysis**
  - Control flow graphs
  - Bounds on operations
  - Performance assertions
  - Performance Assertions
- **Run time traces**
  - Operations Executed
  - Memory address patterns
  - Communication patterns
  - Synchronization events



Vampir trace of UMT2000

- **Consider explicit finite element analysis**
- **10X grid refinement requires:**
  - **scale work by 10000X**
  - **scale memory volume 1000X**
  - **scale communication volume 100X**
- **Maintain global synchronization points**
- **Approximate point-to-point communication patterns**

- **Envelope Simulator**
  - Generate best/worst case bounds on performance
- **Metasim Convolver**
  - Convolve Petascale workload with HPCS machine parameters
- **Explicit Simulator**
  - A discrete event like execution of the Petascale workload
- **Symbolic Simulator**
  - Could one generate a differentiable model?





- **How can HPCS modeling efforts collaborate?**
  - We intend to offer our model Petascale applications to other HPCS efforts
  - We would like to import application and system models from other groups
  - Common data formats and/or interfaces required
- **How do we interface with Development Time?**
- **How do we integrate with workflows?**
- **Want to develop a Common Modeling Interface with other HPCS investigators**



# Cascade: Application Kernel Matrix



- **Kernel Matrix:**

- *The matrix is a graphical representation of all the submissions that we have received and confirmed*
- *Programmers can submit a "generic" solution, or one that is tuned for high performance on a specific computer system*
- *If you submit a kernel solution in a language that hasn't been used in the matrix before, a new row gets added to the matrix*
- *Hover over a row, column, or cell for more information about already-submitted solutions*

| Select a target system: Generic Implementation <input type="button" value=""/> |    |     |    |     |     |     |     |     |     |     |  |
|--|----|-----|----|-----|-----|-----|-----|-----|-----|-----|--|
|  | CG | S3D | UA | CCG | CFD | NFT | NMG | PSM | SMB | VMP |  |
| Fortran  |    |     |    | 1   |     |     |     |     |     |     |  |
| Unified Parallel C   |    |     |    |     |     |     |     |     |     |     |  |
| Chapel   |    |     |    |     |     |     |     |     |     |     |  |
| Most recent submission: August 17 2004 @ 15:34:39                              |    |     |    |     |     |     |     |     |     |     |  |

| Select a target system: Tuned for the Cray X1 <input type="button" value=""/> |    |     |    |     |     |     |     |     |     |     |  |
|---|----|-----|----|-----|-----|-----|-----|-----|-----|-----|--|
|   | CG | S3D | UA | CCG | CFD | NFT | NMG | PSM | SMB | VMP |  |
| Fortran   |    |     |    |     |     |     |     |     |     |     |  |
| Unified Parallel C  |    |     |    |     |     |     |     |     |     |     |  |
| Chapel  |    |     |    |     |     |     |     |     |     |     |  |
| Most recent submission: August 17 2004 @ 15:34:39                             |    |     |    |     |     |     |     |     |     |     |  |

- **For additional information contact David Mizell (Cray)**
- **Available at <http://akm.cray.com/index.php>**