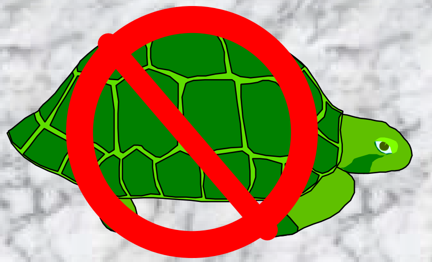


Delphi: Integrated, *Language-Directed* Performance Prediction, Measurement, and Analysis

Dan Reed (Illinois) David Padua (Illinois)

Ian Foster (Argonne) Dennis Gannon (Indiana)

Bart Miller (Wisconsin)

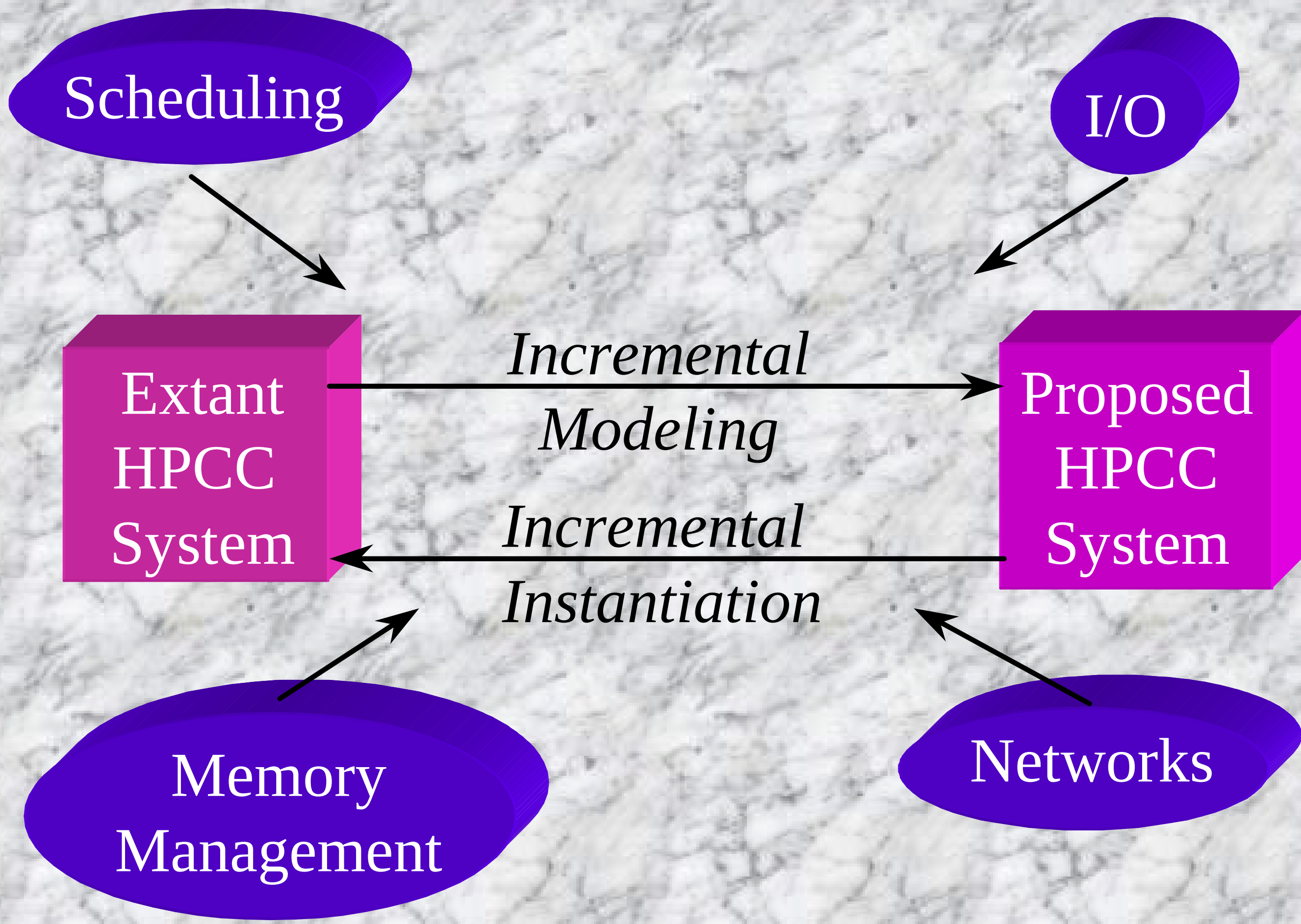
Presentation Outline

- Challenges and approaches
- Delphi design and components
  - integrated prediction
  - performance measurement
  - resource performance models
- Computational grid directions

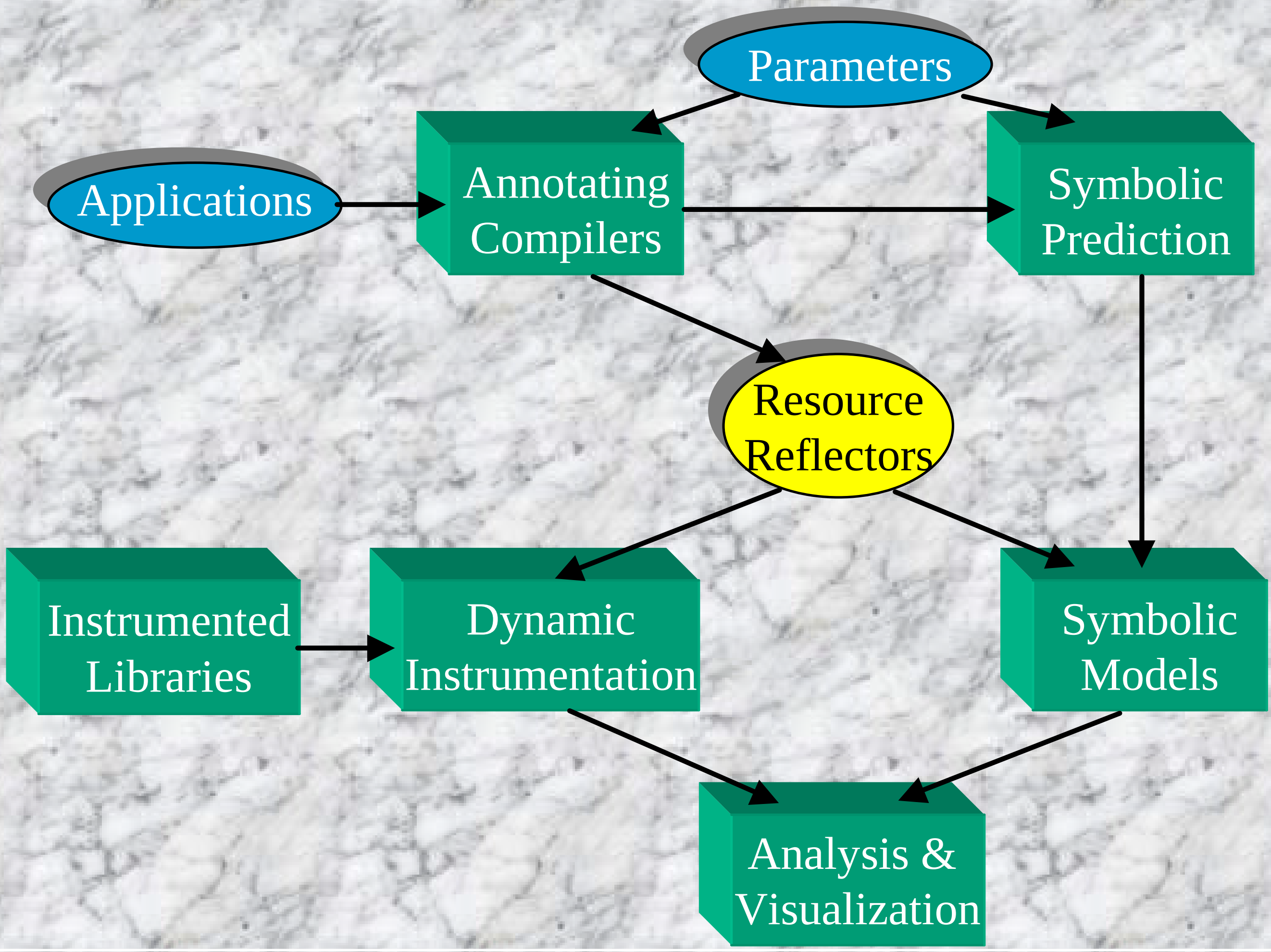
Performance Directed Design

- Integrated approach
  - compiler-directed analysis and instrumentation
  - performance measurement
  - system component modeling
  - scalability prediction
- Static and dynamic evolution
  - static (repeatable behavior)
  - *dynamic* (possibly non-repeatable behavior)

Performance Directed Design



Delphi Organizational Overview



Leverage and Goals

- High leverage (or baggage :-)
  - Polaris, Sage++, HPC++ and HPF
  - Paradyn, Pablo, Autopilot and Virtue
  - Nexus, Globus, I/O libraries and ORBs
- Research goals
  - integrated measurement and modeling
  - language-based analysis
  - comparative system evaluation
  - wide-area computational grid analysis

Integration Approach

- Initial pairwise integration
  - Wisconsin/Argonne: Paradyn/Globus/Pablo
  - Illinois: compiler/symbolic models and I/O
  - Indiana/Argonne: ORB models
- Validation
  - DSM and computational grids
  - SFExpress and PACI DSM codes
- Complete integration

Paradyn/Globus Integration

- Thread instrumentation and measurement
  - Paradyn dynamic instrumentation
  - performance measurements
- Globus integration
  - Paradyn wide area measurements
- Pablo integration
  - SDDF support

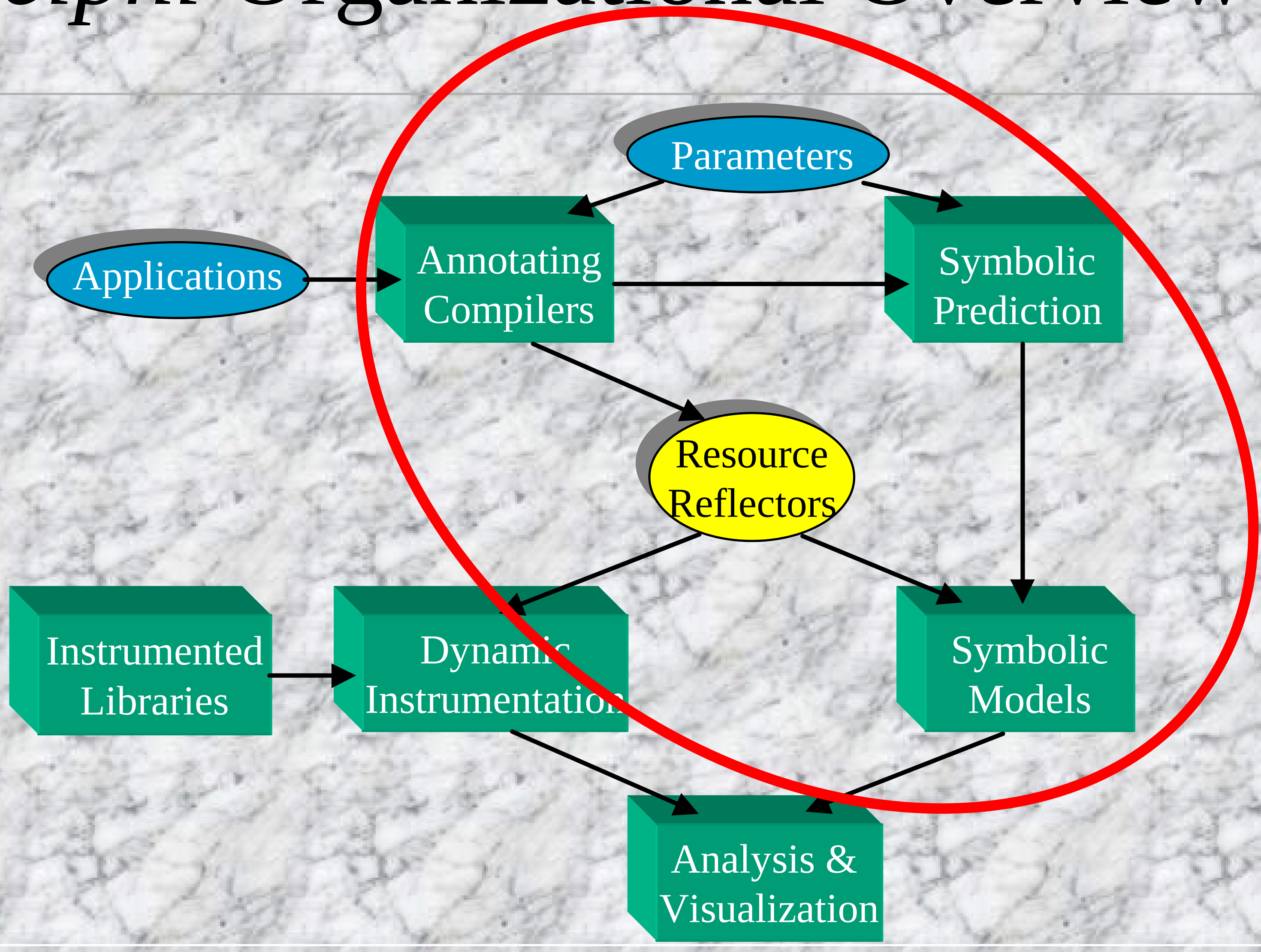
Pablo/Polaris Integration

- Memory access time models
  - stack distance estimates
  - benchmark validation
- Symbolic scalability models
  - Polaris operation counts with memory costs
  - benchmark validation
- SvPablo integration
  - source code performance prediction
  - dynamic measurements (operations and I/O)

HPC++/Globus Integration

- Threading and scheduling
  - thread package performance measurements
  - comparative models
- Wide-area ORB behavior
  - Nexus/Globus substrate measurements
  - Java comparisons

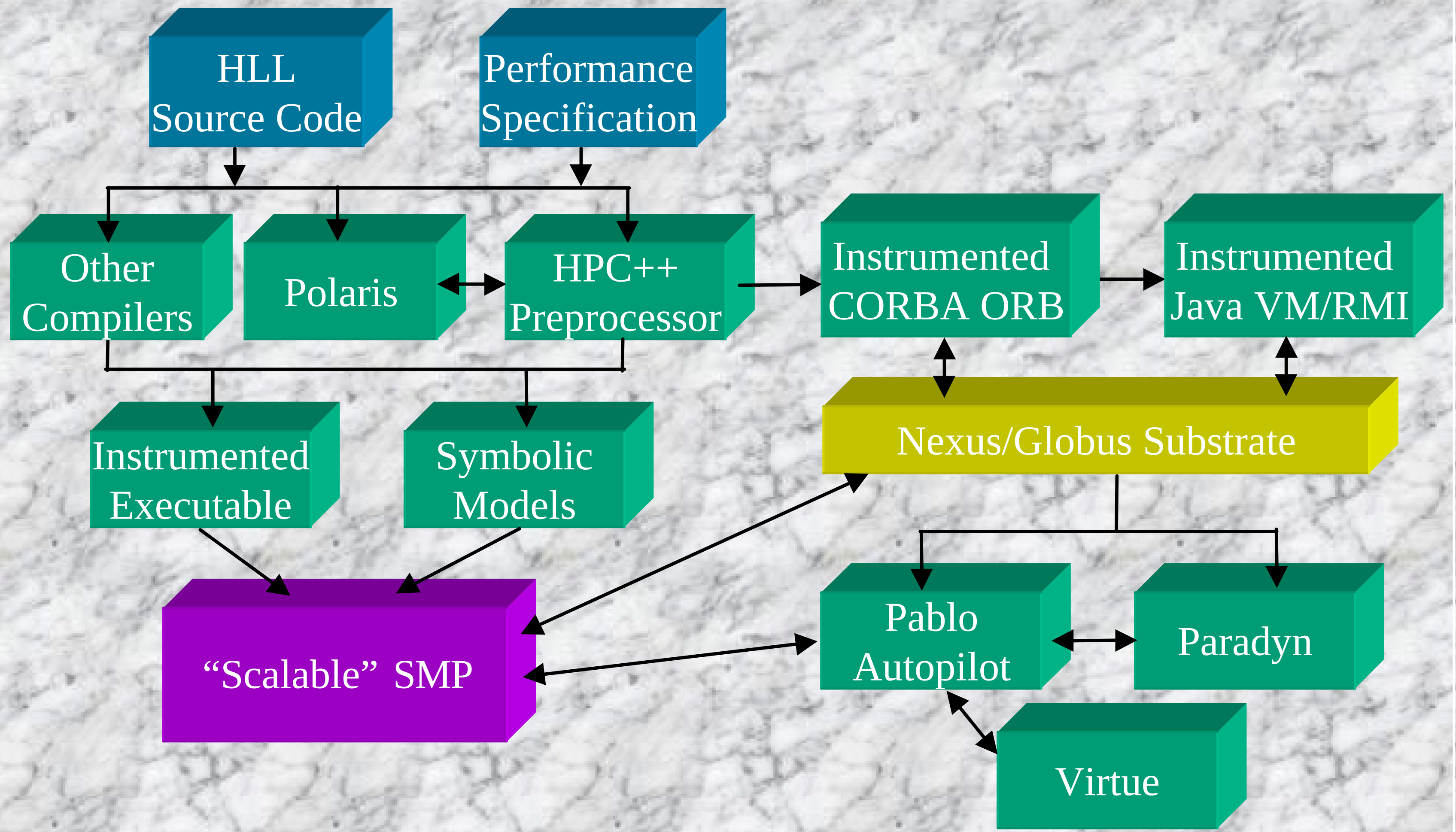
Delphi Organizational Overview



High-Level Language Integration

- Motivations
  - emerging high-level languages (HPF and HPC++)
  - aggressive code transformations for parallelism
  - large semantic gap between user and code
- Goals
  - relate dynamic performance data to source
  - generate instrumented executable/simulated code
  - support performance scalability predictions

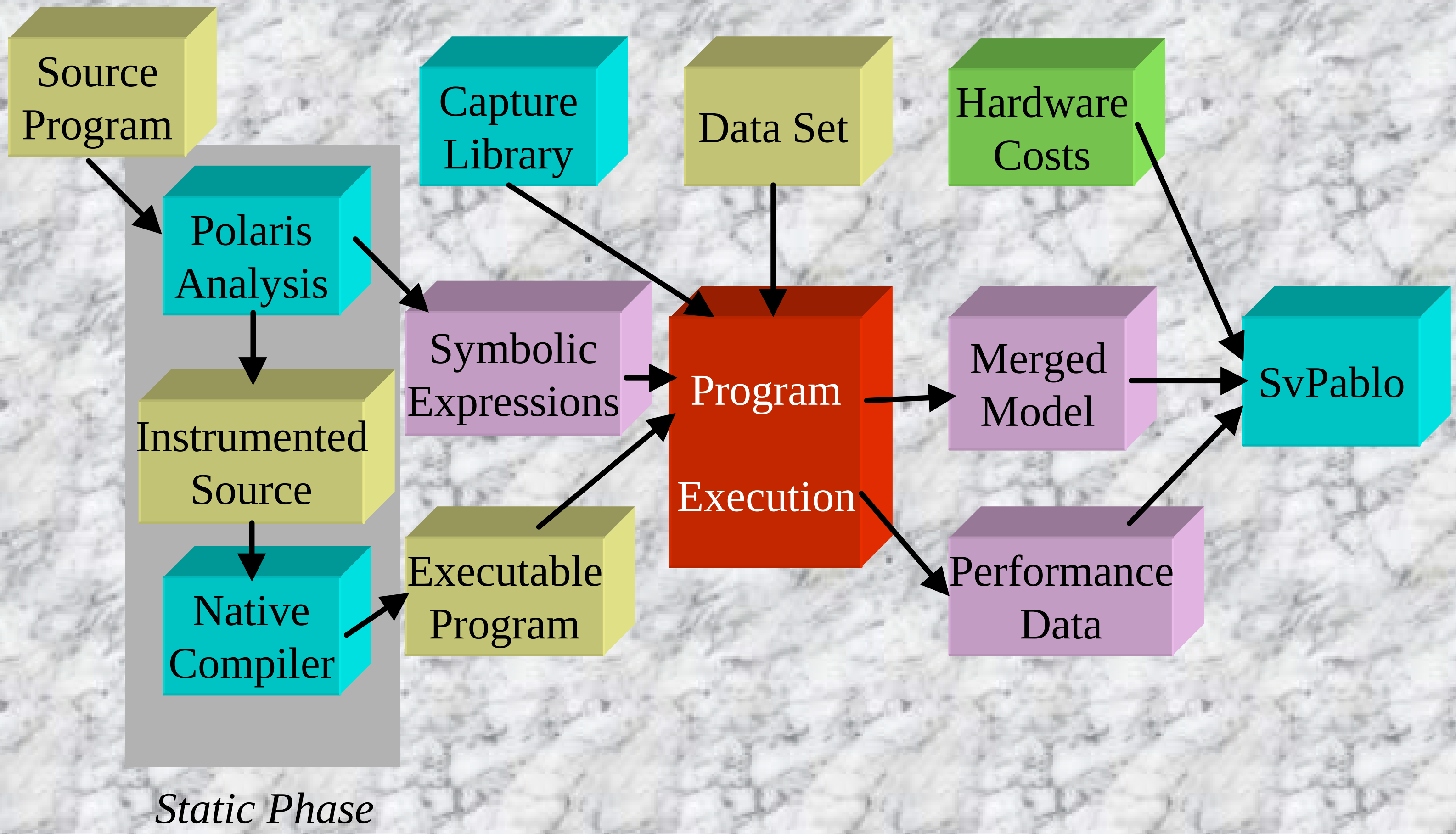
High-Level Language Integration



Symbolic Performance Prediction

- Rationale
  - rapid assessment of design alternatives
  - identification of performance bottlenecks
- Approach
  - compile-time generation of cost expressions
  - augmentation with selected measurements
    - runtime behavior (benchmarks and applications)
    - hardware counters (micro-benchmarks)

Polaris/Sage++ Integration

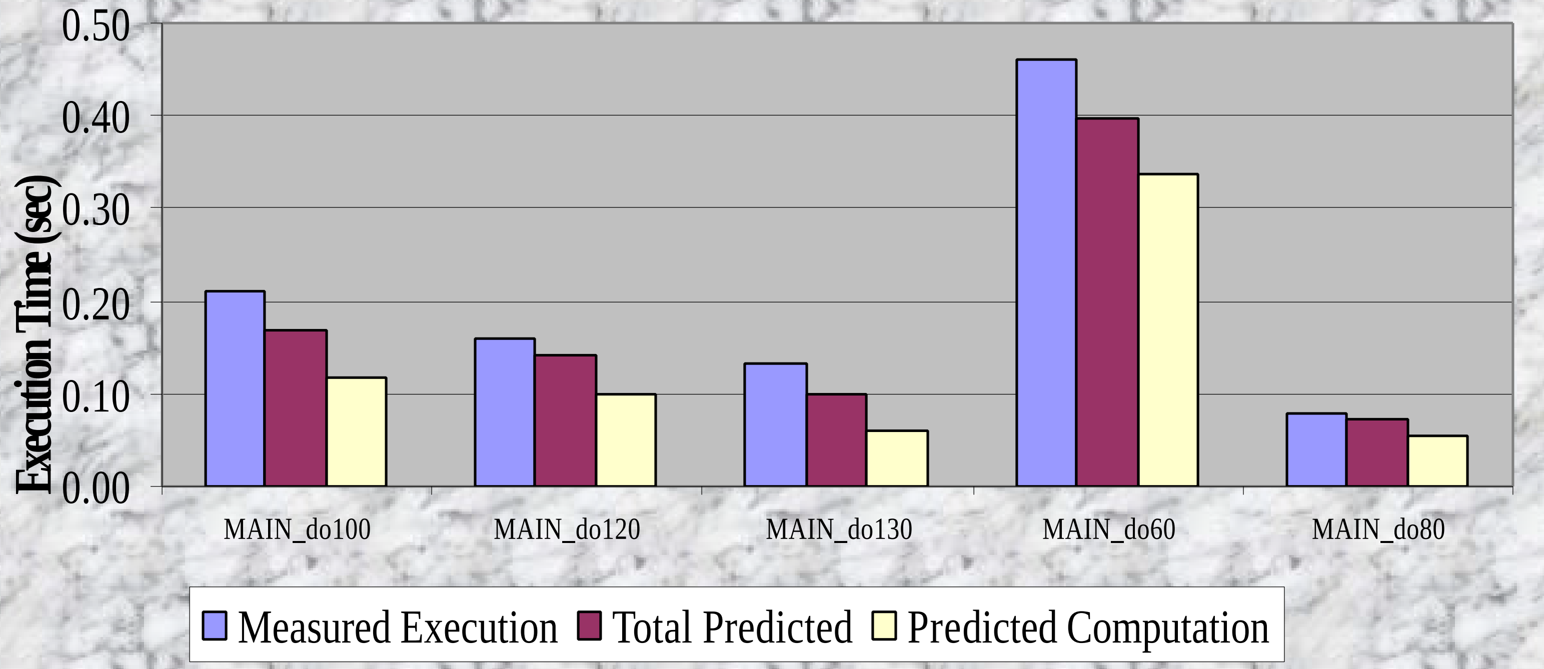


Symbolic Performance Prediction

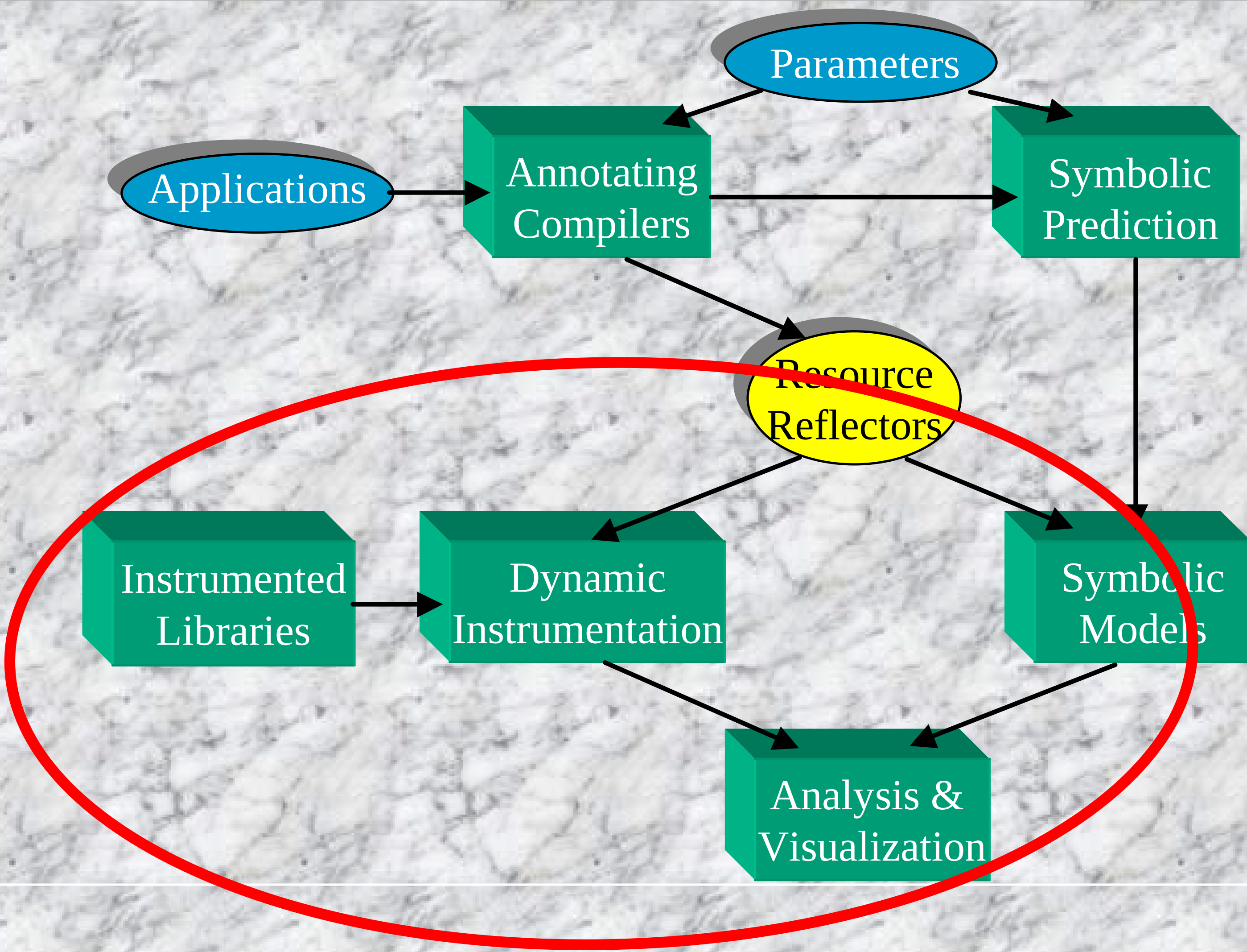
- Two phase prediction approach
  - static traversal of compile-time AST
  - dynamic integration of measured data

- Performance predictions
  - instantiate array sizes (loop bounds)
  - specify number of processors
  - evaluate symbolic expression

TOMCATV Predictions



Delphi Organizational Overview

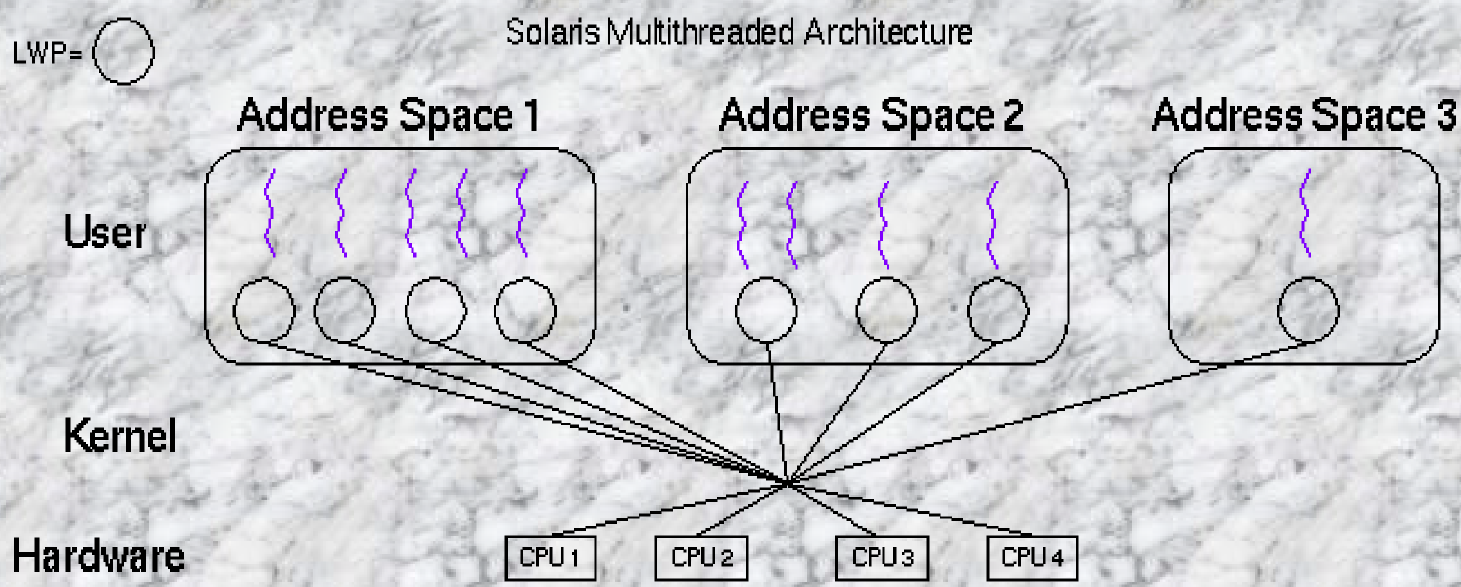


Instrumentation Integration

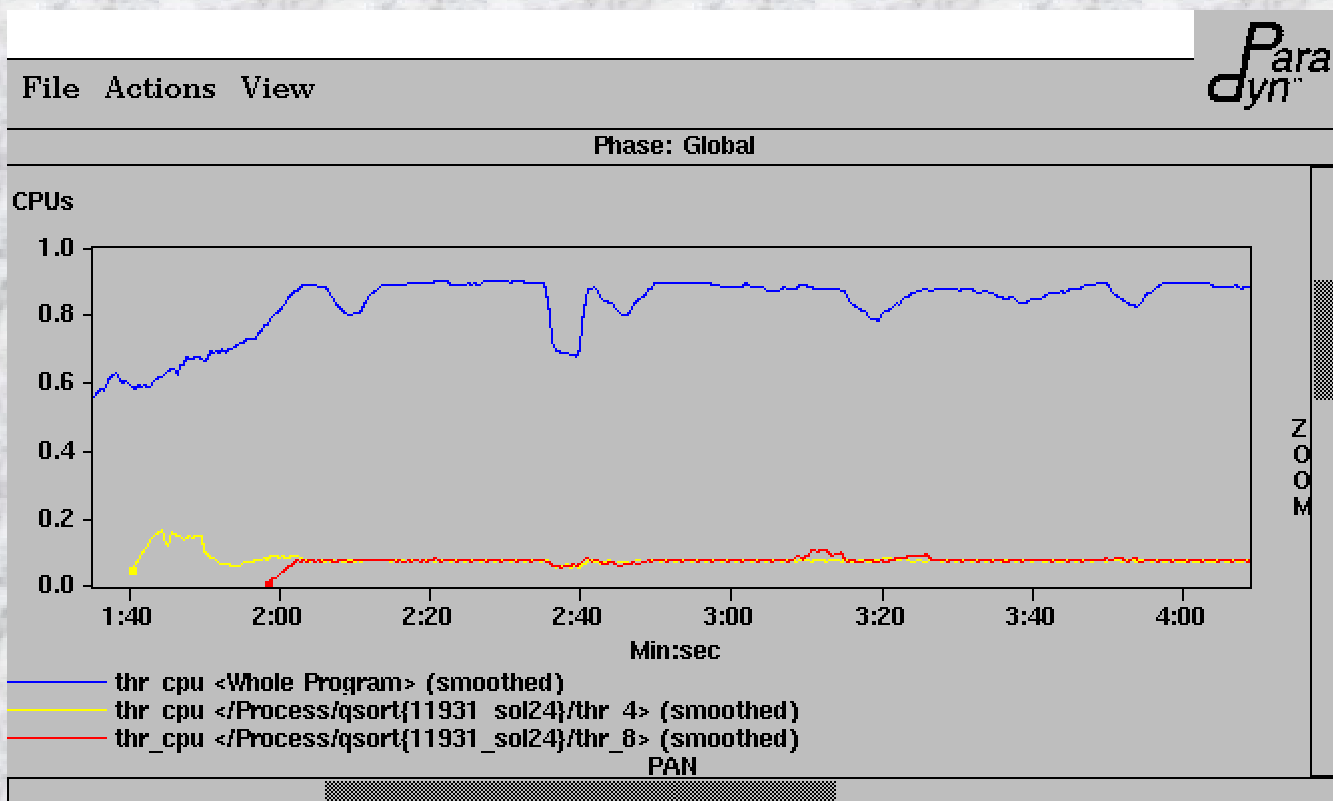
- Leverage best toolkit features
  - Paradyn
    - dynamic object code patching
    - standard software metrics
    - hardware performance data support
  - Pablo/Autopilot/Virtue
    - real-time data analysis
    - flexible data metaformat
    - adaptive resource control
    - performance visualization

Paradyn Instrumentation

- Multithreaded support rationale
  - exploit multiprocessor hardware, application concurrency
  - used heavily in parallel programming, UI's, servers



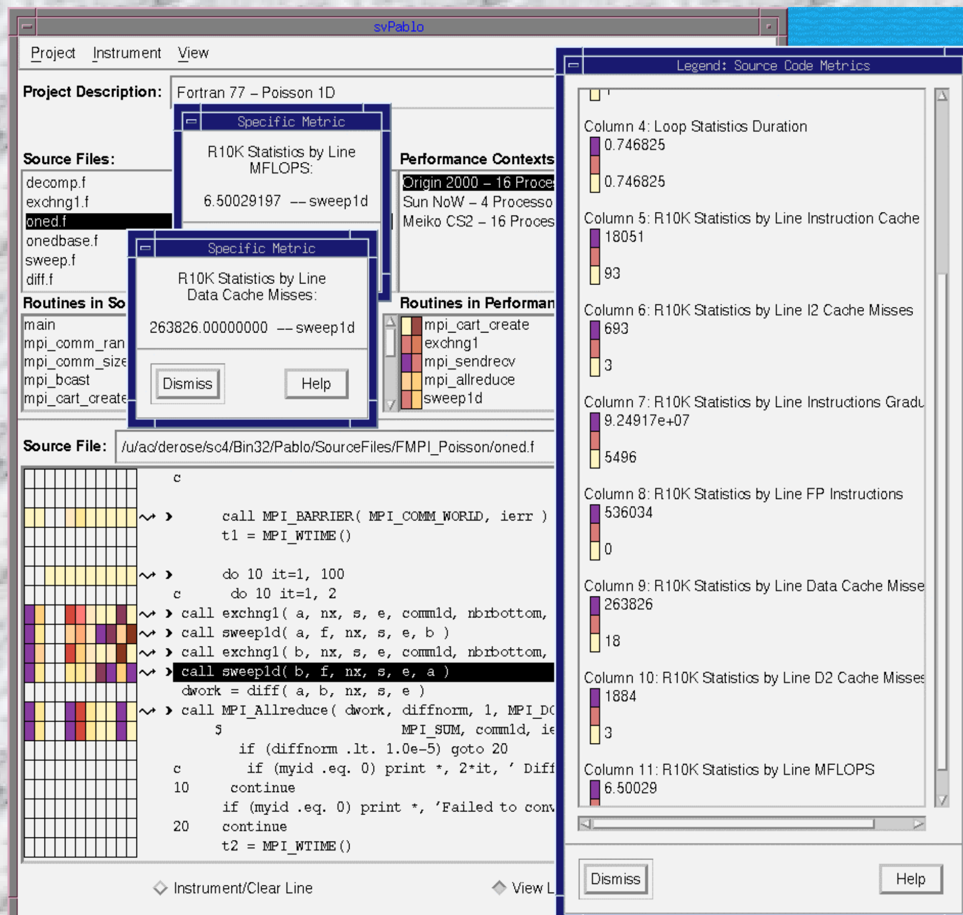
Paradyn Measurements



SvPablo Performance Browser

- Instrumentation
  - automatic
    - PGI HPF
  - interactive
    - ANSI C
    - Fortran 77/Fortran 90
- Data capture
  - dynamic software statistics
  - SGI R10000 counter values

SvPablo Code Browser



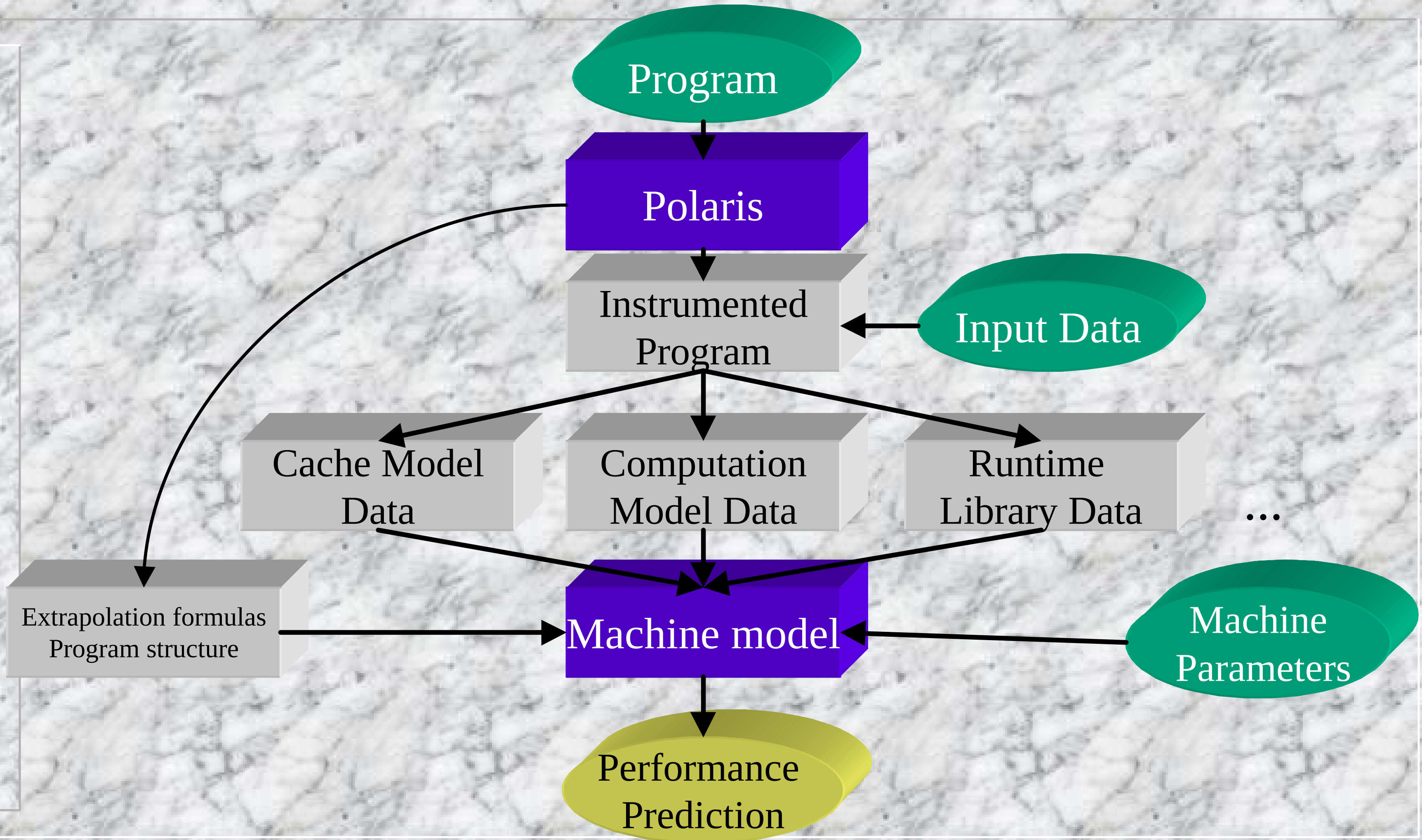
SvPablo Language Transparency

- Meta-metaformat for performance data
  - language defined by line and byte offsets
  - metrics defined by mapping to offsets
  - SDDF records
    - performance mapping information
    - performance measurements
- Result
  - language independent performance browser
  - mechanism for scalability model integration

Model and Library Integration

- Rationale
  - instrumented libraries for resource measurement
  - models of resource use for proposed systems
- Four foci
  - memory hierarchy (caches and remote memory)
  - scheduling (threads and tasks)
  - communication (network QoS and ORBs)
  - input/output (MPI-IO and other APIs)

Multimodal Modeling



Memory Hierarchy Models

- Rationale
  - estimate memory access and operation costs
  - validate and tune symbolic models
- Compile-time models
  - iteration space analysis and restructuring
  - cache size and replacement algorithms
- Runtime models
  - hardware counter measurements
  - probabilistic estimates

Scheduling Models

- Rationale
  - understand *irregular*, *adaptive* computations
    - complement to POEMS effort
  - analyze memory/scheduler interactions
  - enable distributed shared memory analysis
- Approach
  - analytic, simulation, and stochastic models
  - augmented by compile-time information

Communication Models

- Rationale
  - analyze *wide-area metacomputing* systems
  - estimate bandwidth and latency impacts
  - understand Quality of Service (QoS) implications
  - assess Object Request Broker (ORB) performance
- Approach
  - exploit compile-time and runtime data
  - build calibrated QoS and ORB models
  - integrate with Globus metacomputing toolkit

Threads and Objects

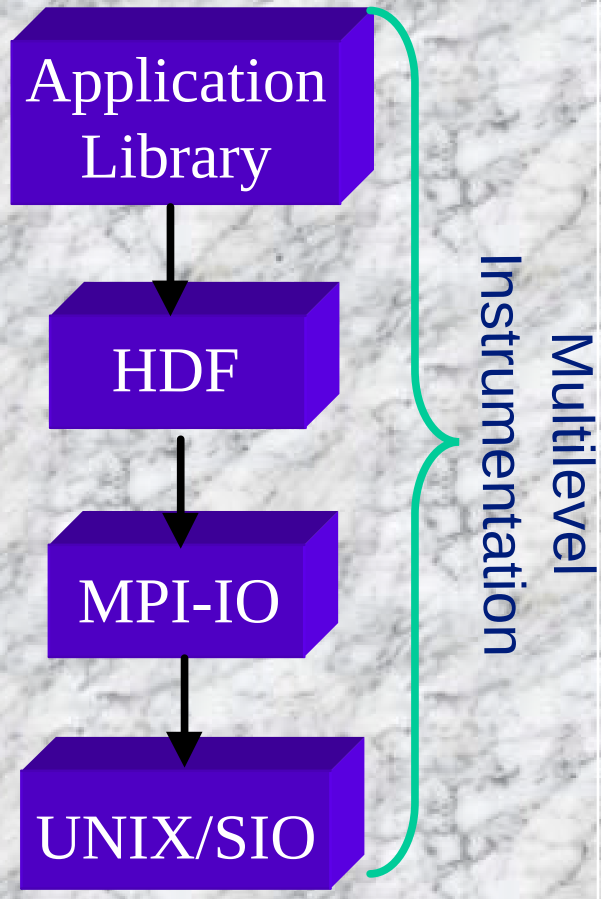
- Consider
    - an adaptive, multithreaded flow solver
  - coupled to
    - a remote, multithreaded tree-based N-Body code
  - coupled to
    - a visualization in a CAVE
- Optimization using performance models?

Some First Steps

- Demonstrate a simple, first order model for thread behavior in HPC++ programs.
- Illustrate how distributed object remote method invocation performance analysis is not as easy as one might hope
- Next steps
  - demonstrate distributed object tuning with applications

I/O Characterization and Models

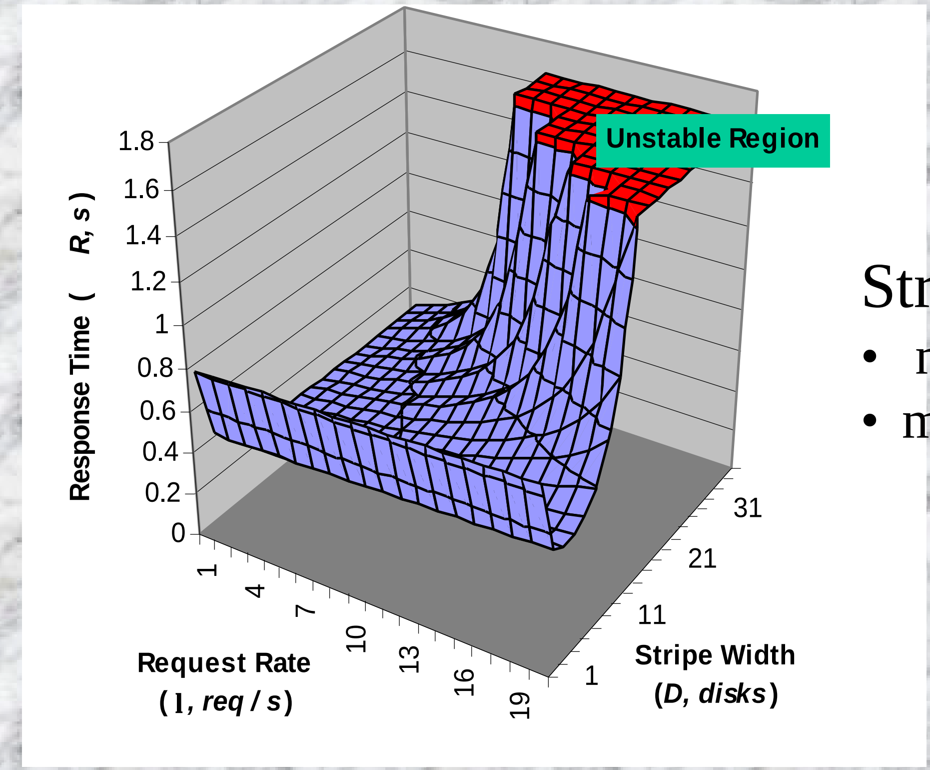
- Multilevel analysis
  - correlation across levels
  - policy mismatch studies
  - extended SIO toolkit
- Rationale
  - performance sensitivity
  - library interface data sharing
  - library optimization guide



I/O Characterization

- Approach
  - statistical summaries (production codes)
  - detailed event traces (exploratory analysis)
- Instrumentation toolkits
  - HDF version 4 (operational and available)
  - HDF version 5 (operational)
    - evolving in collaboration with HDF 5 development
  - MPI-IO (ROMIO)
    - operational and now available

Disk Striping Models



Striped disk access

- multiple disks increase transfer rate
- multiple disks decrease throughput

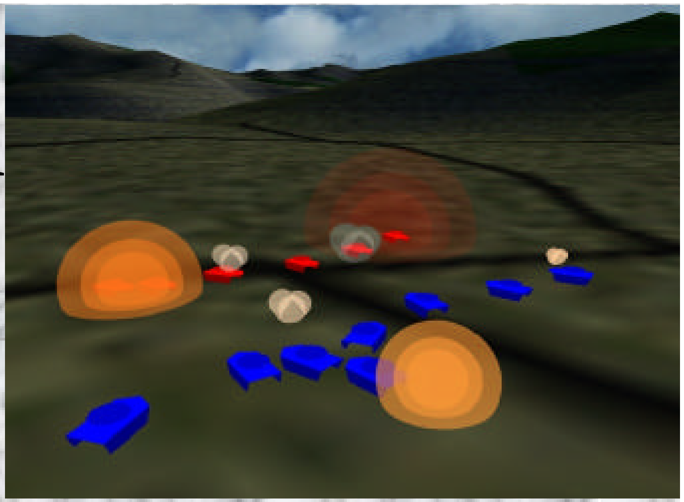
Grids and Delphi

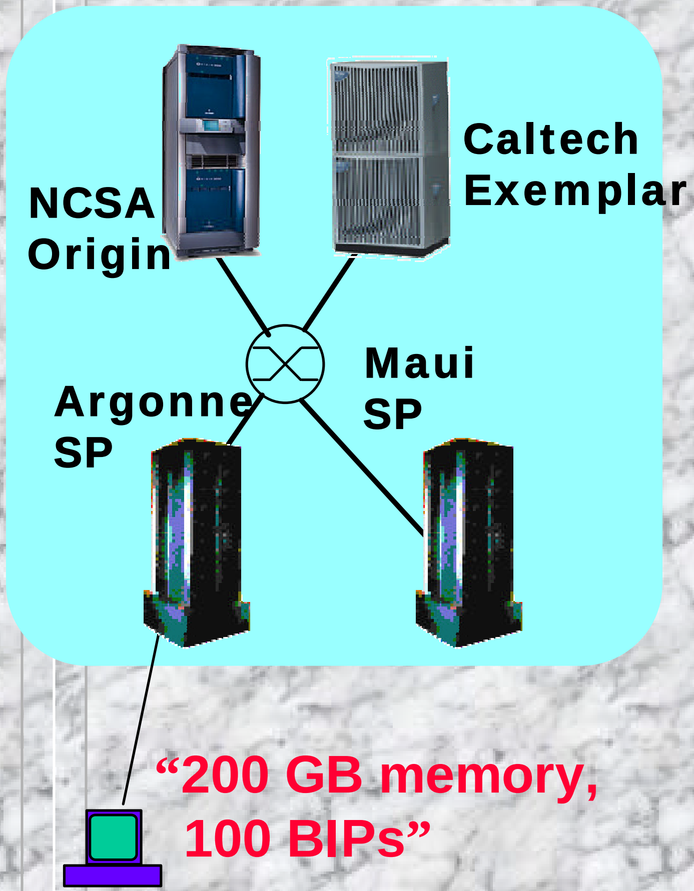
- Grids are a “double whammy”
  - complex architectures, dynamic behaviors
  - complex applications, dynamic behaviors
- Accurate performance estimation
  - resource selection, scheduling, configuration, and adaptivity
- Delphi enables first steps in this area

Grids and Delphi

- Build on existing technology base
  - Globus, Paradyn, Autopilot, HPC++
- Develop instrumentation for grid environments via Globus
- Study application performance
- Basis for
  - performance analysis
  - characterization and estimation

SF-Express: Distributed Interactive Simulation





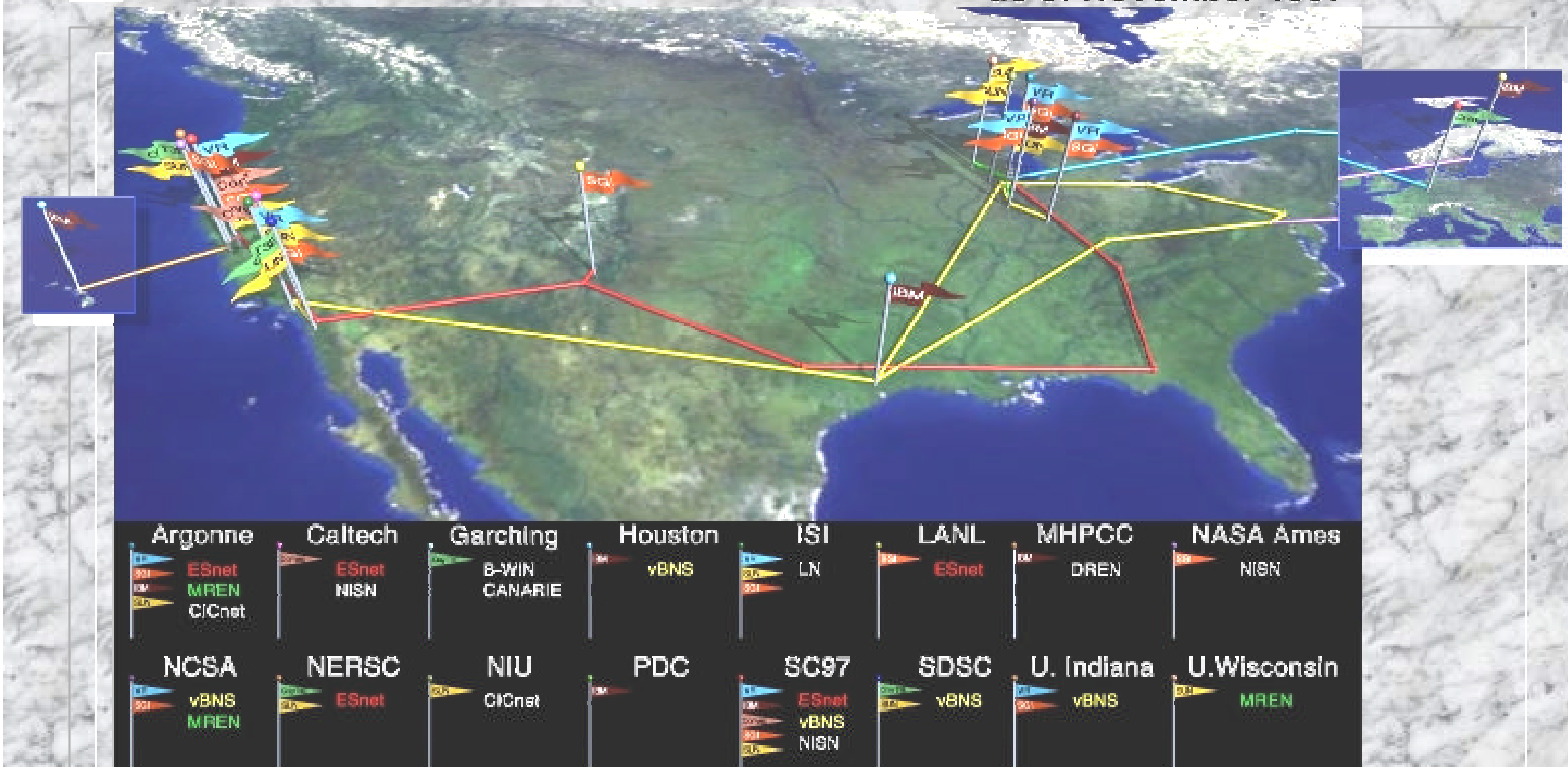
"200 GB memory, 100 BIPs"

- Performance issues
  - computational structure
  - network requirements
  - communication methods scalability

P. Messina et al., Caltech

GUSTO Computational Grid Testbed

as of November 1997



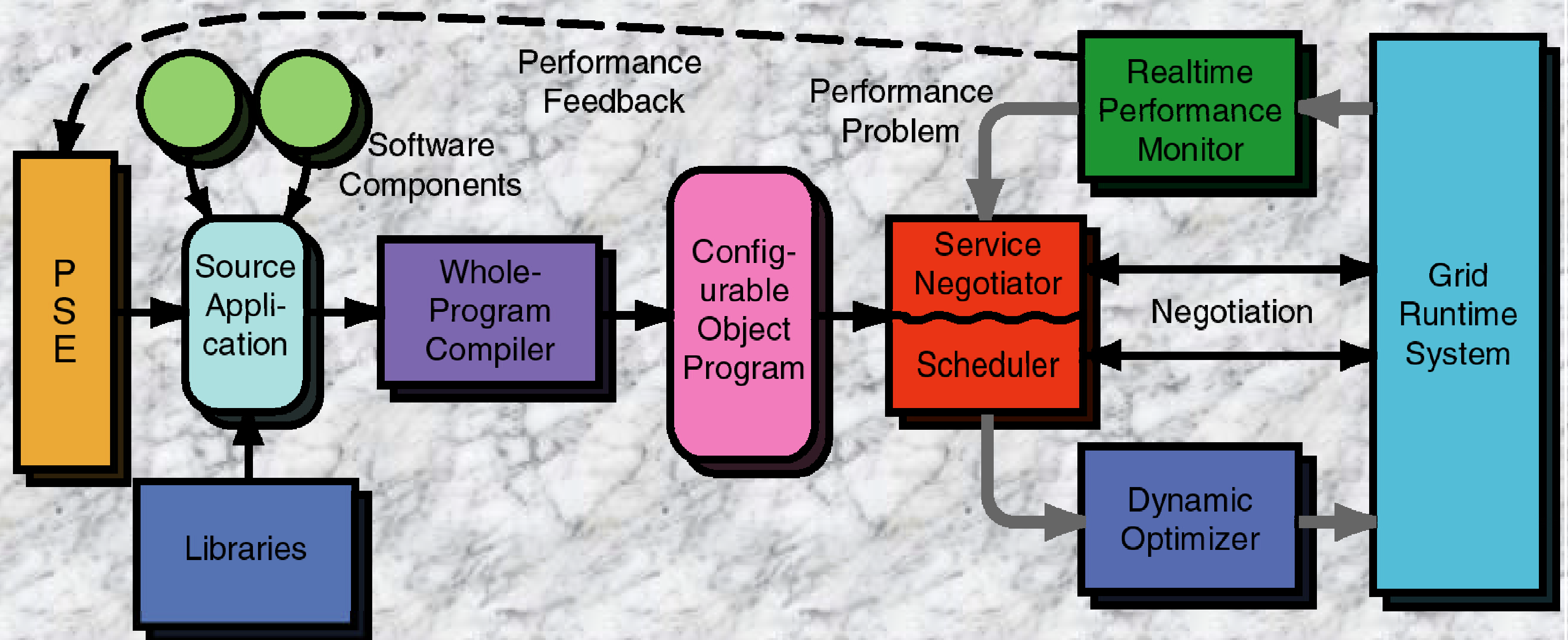
16 sites, 330 computers, 3600 nodes, 2 Teraflop/s, 10 application partners

Broader Grid Research Goals

"Provide technologies allowing programmers to develop applications that achieve high performance in environments in constant flux"

- Via an integrated treatment
  - compilers, languages, libraries and algorithms
  - problem solving environments
  - runtime systems and scheduling
  - performance and other tools

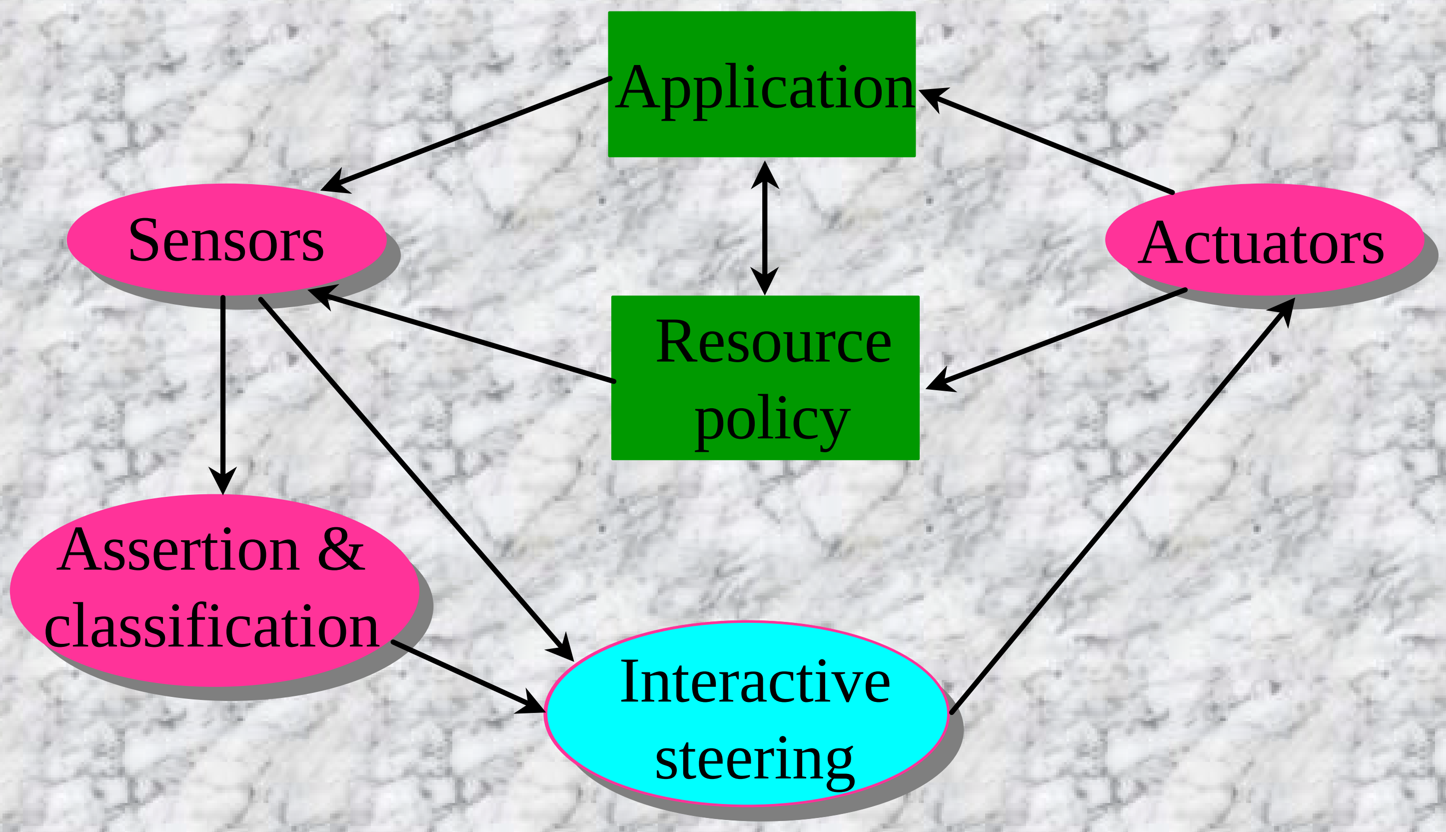
Grid Application Development



Grid Implications

- Dynamic optimization requires
  - resource models
  - scheduling
  - real-time measurement and control
  - wide-area infrastructure
  - multiple execution modes
- *Components from projects at this workshop*

Autopilot Adaptive Infrastructure



Participants

- Fran Berman
- Keith Cooper
- Jack Dongarra
- Ian Foster
- Dennis Gannon
- Ken Kennedy
- Carl Kesselman
- Lennart Johnsson
- Dan Reed
- Linda Torczon