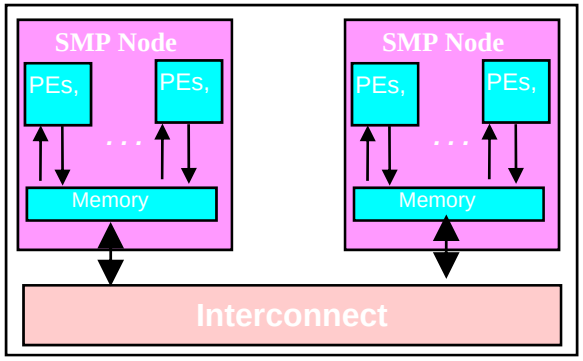
Petascale Challenges and Solutions

Kevin Gildea
gildeak@us.ibm.com

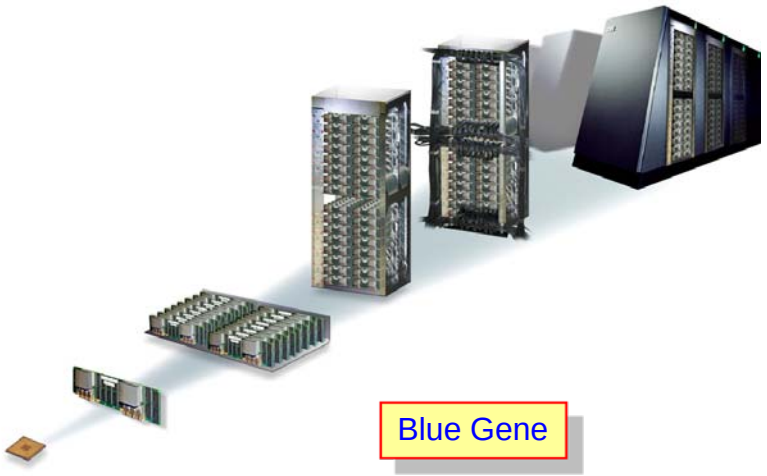
Agenda

- Petascale Landscape and Challenges
- HPCS and PERCS Solution
- PERCS Hardware Innovations
- PERCS Software and Productivity

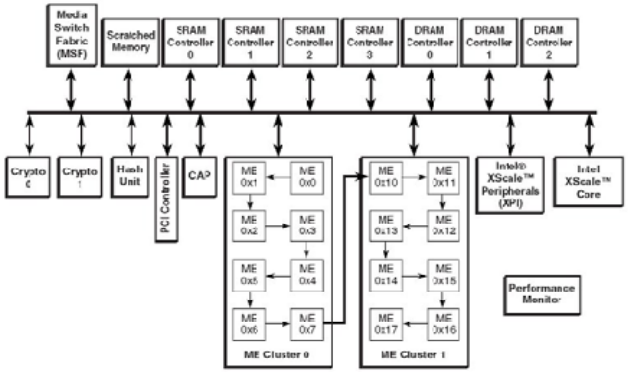
The current architectural landscape



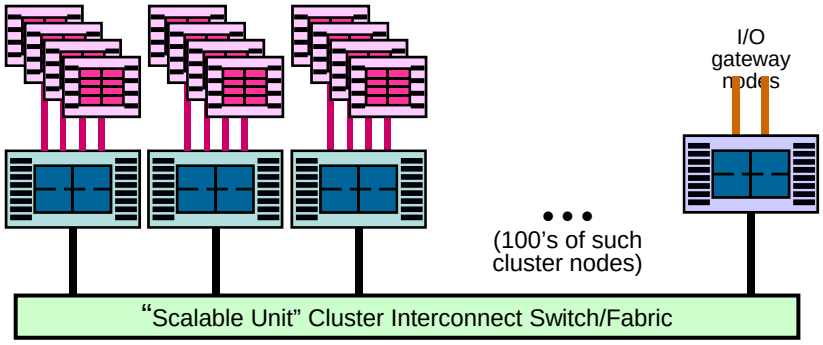
Power6 Clusters



Blue Gene



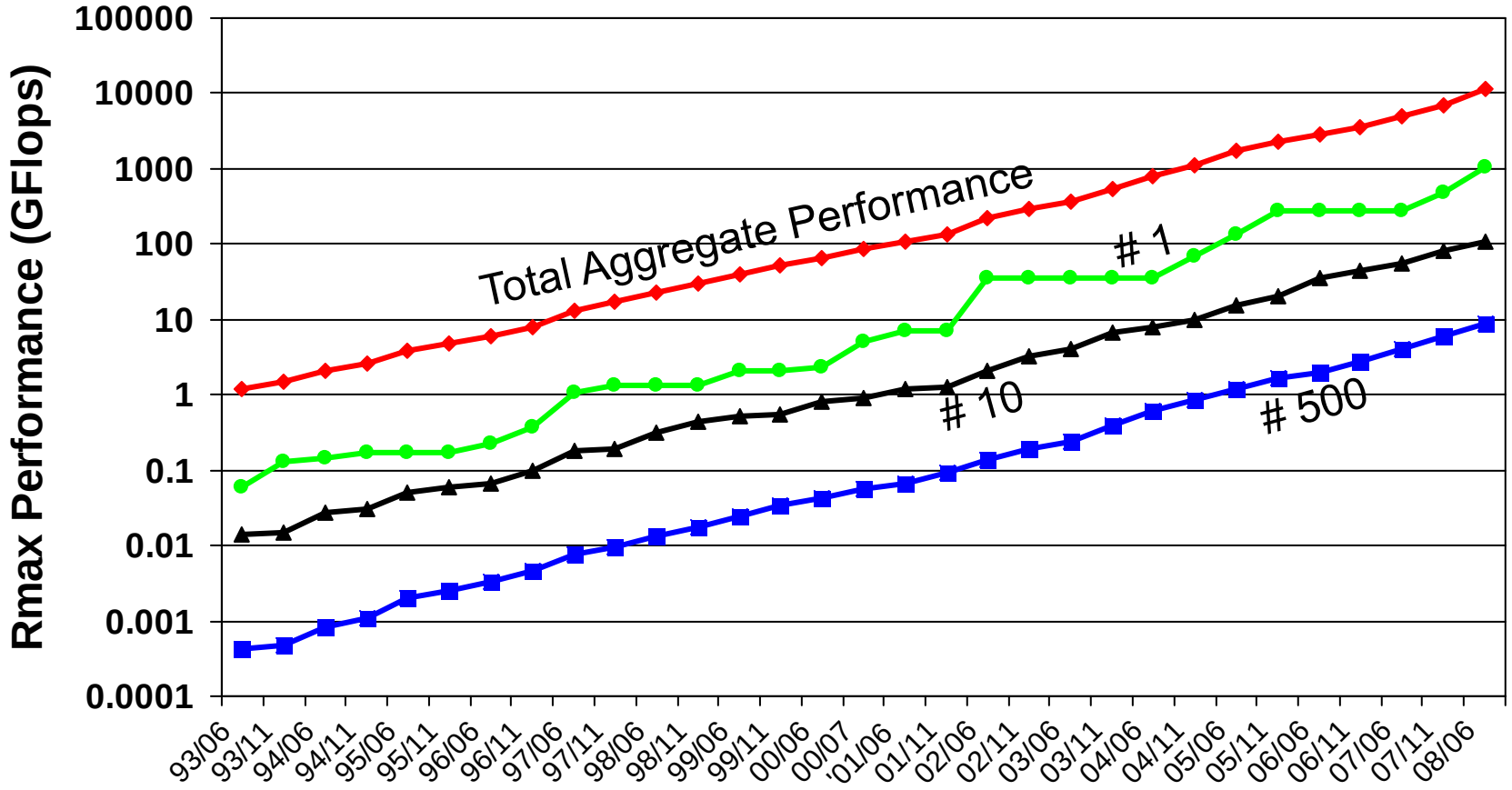
Multi-core w/ accelerators *(IXP 2850)*



Road Runner: Cell-accelerated Opteron

TOP500 Performance Trend

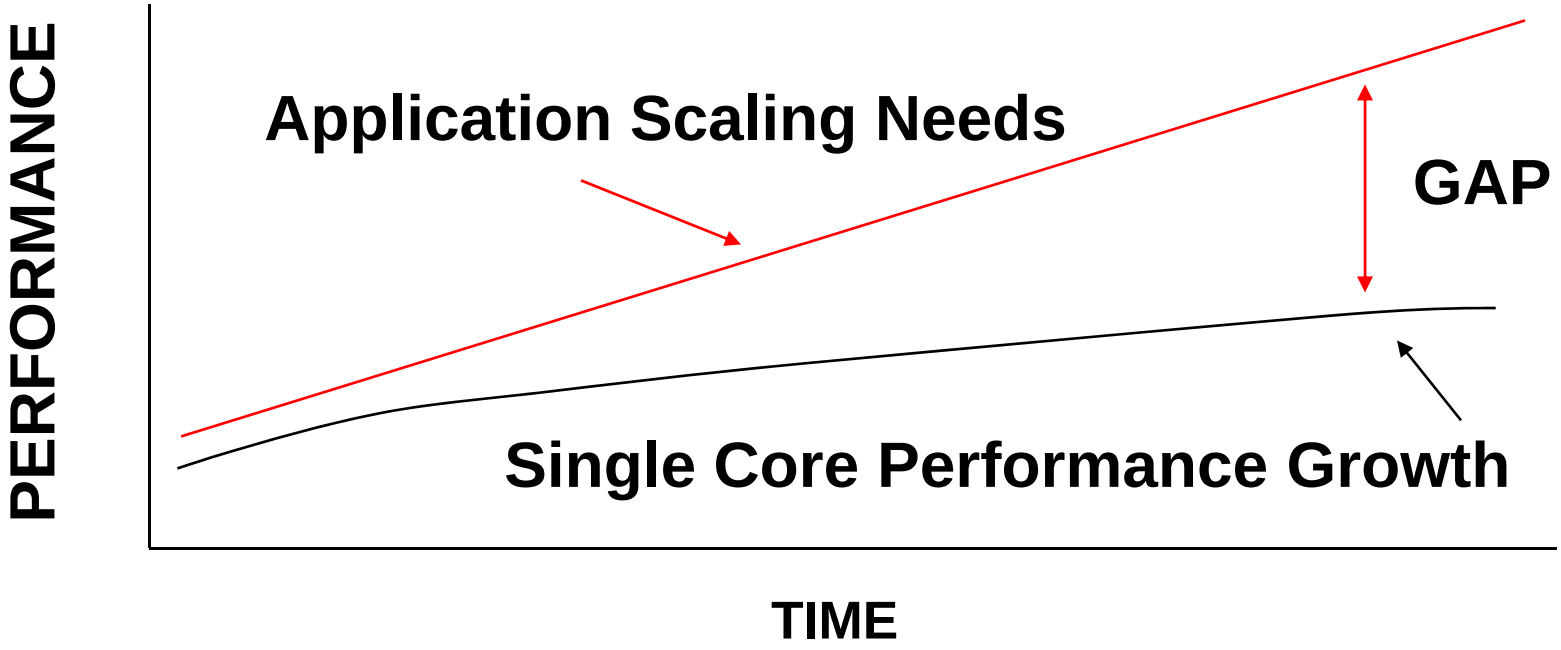
Even though there is some stepping of the performance of the #1 system. The #500 clip level, #10 clip level and Total Aggregate performance all are virtual straight line trends when plotted on log scale (~ 96% CGR)



How are all these systems programmed?

- Automatic parallelization of sequential codes
  - Polaris, xlc –qsmp=auto, etc.
  - Successful for a limited application domain and relatively small scale
- MPI (+ OpenMP or pthreads)
  - The dominant model today, scales well to large numbers of processors
  - Increasingly considered too complex to program
- Parallel libraries
  - Parallel ESSL, PLAPACK, ScaLAPACK, STAPL, HTA, Intel TBB
  - Composability
- Explicit parallel languages or parallel language extensions
  - OpenMP – small scale (hundreds of threads)
  - PGAS: UPC, CoArray Fortran, Titanium, X10, Chapel
  - Fortress

Programmer Productivity



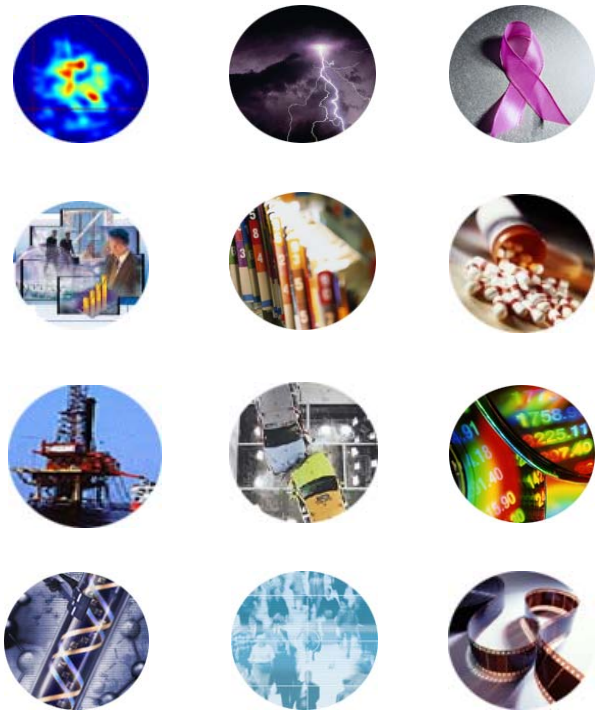
Key Problem: Frequency Improvements Do Not Match App Needs

Increasing Burden On The Application Design

Objective: Provide Tools to allow Scientists to Bridge the Gap

What are the key challenges to advancing Technical Computing?

- Productivity: ***How do make this massive compute power more <u>consumable</u> and <u>reduce time-to-insight?</u>***



- Performance: ***How do we, at the same time, provide <u>sustained growth in application level performance</u> in the face of a technology discontinuity?***

High Productivity Computing Systems Overview

Critical to National Security

- Develop a **new generation** of **economically viable** high productivity computing systems for national security and industrial user communities (2011)
- **Ensure U.S. lead, dominance, and control** in this critical technology

Phase III Vendors: IBM, CRAY

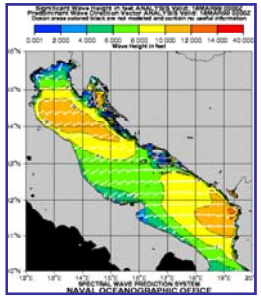
Mission Partners: NNSA, NSF, Office of Science

Impact:

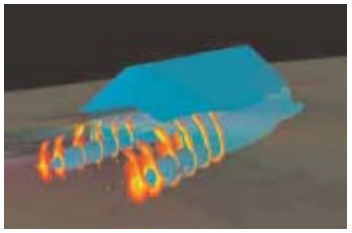
- **Performance** (time-to-solution): speedup by **10X to 40X**
- **Programmability** (idea-to-first-solution): dramatically reduce cost and development time
- **Portability** (transparency): insulate software from system
- **Robustness** (reliability): continue operating in the presence of localized hardware failure, contain the impact of software defects, and minimize likelihood of operator error

Applications:

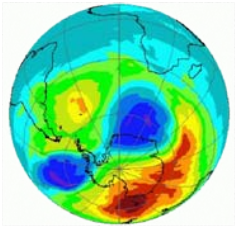
Weather Prediction



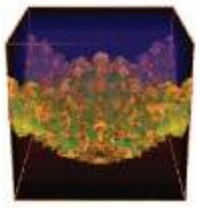
Ocean/wave Forecasting



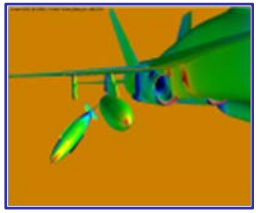
Ship Design



Climate Modeling



Nuclear Stockpile Stewardship



Weapons Integration

PERCS – Productive, Easy-to-use, Reliable Computing System is IBM's response to DARPA's HPCS Program

PERCS Productivity Domains

Programmer

- Develop Applications
- Debug Applications
- Tune Applications

System Operational Efficiency

- Maximize System throughput
- Maximize Enterprise Efficiency
- Ensure System Balance

Administrator

- Storage Management
- Network Management
- Install, Upgrades
- System Monitoring

Reliability and Serviceability

- Continuous Operation
- Problem Isolation
- First Failure Data Capture
- Serviceability

PERCS Productivity Solutions

Programmer

- Eclipse IDE
- Compiler Enhancements
- UPC and X10 Languages
- Automated Performance Tuning

System Operational Efficiency

- Resource & Workload Management
- Protocol Optimization and Acceleration
- Co-scheduling
- Dynamic Page Size Assignments

Administrator

- Automated Discovery
- Automated Configuration
- Diskless Boot
- Rolling Updates

Reliability and Serviceability

- Concurrent and Rolling Update
- Checkpoint/Restart
- Server, Network, & Storage Monitoring
- Declustered RAID

Compiler Focus

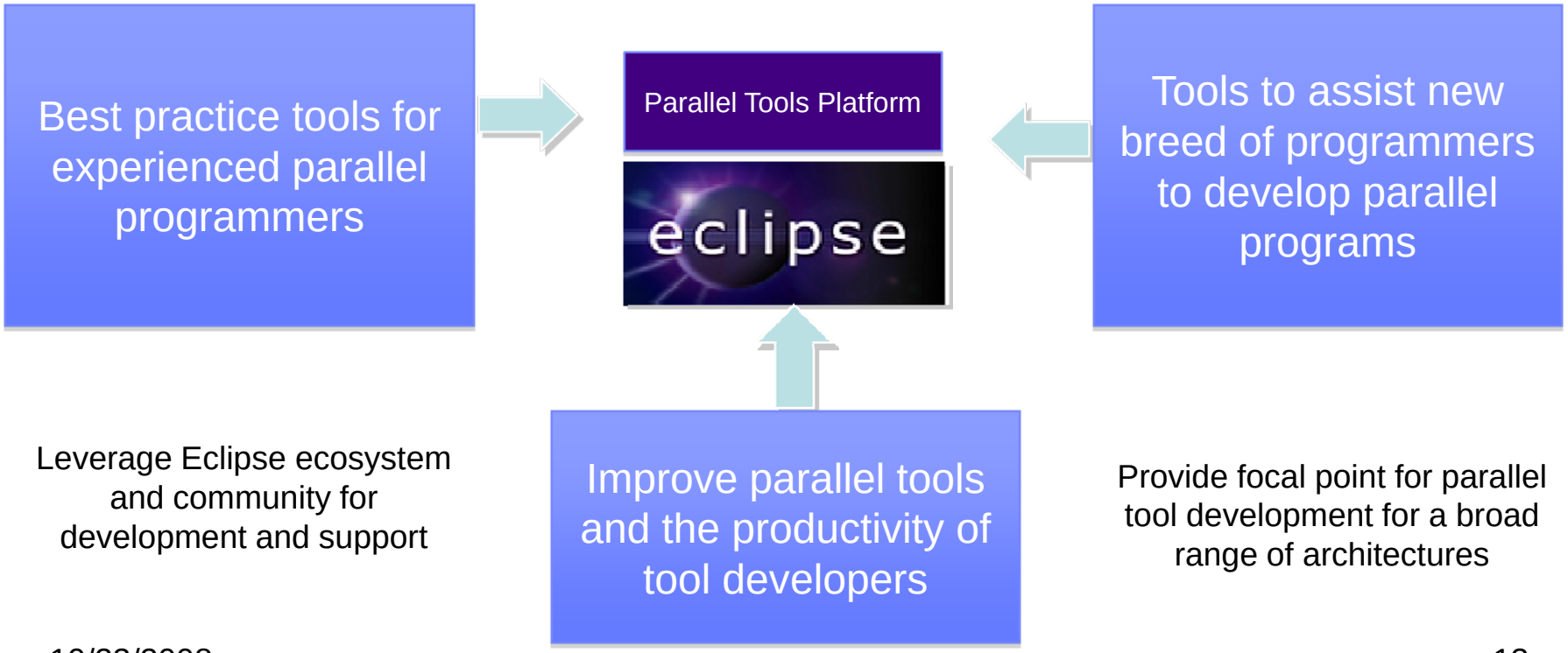
- Performance
  - Automatically exploit POWER 7 hardware characteristics
  - Address key memory wall issues
  - Automatically exploit SIMDization (double precision).
  - Effectively handle parallelization and scaling issues
- Productivity
  - Hide system complexity from programmers
  - Automatically fine tune optimizations for the applications using profile feedback information.
  - Generate transformation reports to help programmers fine tune their source code.
  - Support for legacy applications on new hardware

XL C,C++,Fortran Compilers

- Advanced Memory Optimizations
  - Address memory wall issues, hide system complexity by tuning and improving memory sub-system performance automatically
- XL Compilers Transformation Reports
  - Generate XML enabled reports to help users fine tune their applications.
- Polyhedral framework for Automatic Parallelization
  - Help scaling to large number of threads
  - Exploit multi-level parallelism provided by POWER 7 hardware
- Assist Threads
  - Deploy the available multiple SMT threads and cores to increase single thread performance.

Parallel Tools Strategy
Eclipse-based Parallel Tools Platform

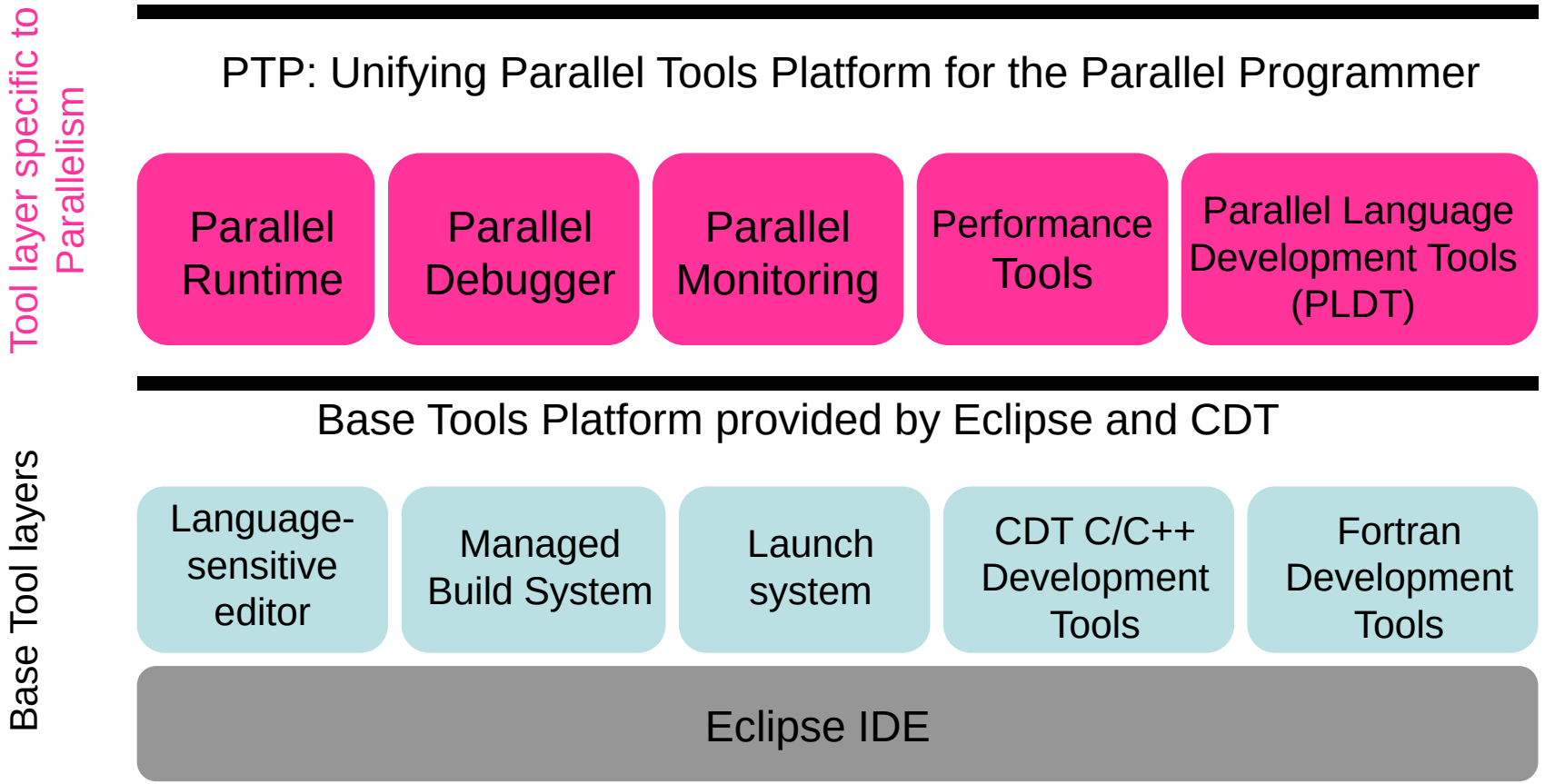
- **Bring richness of commercial IDEs to the HPC programmer**
  - –Grow HPC ecosystem around common IDE
  - –Address the needs of HPC users ranging from novice to expert parallel programmers
- **Open and extensible to encourage further development by IBM and others**



Parallel Tools Strategy

Eclipse Parallel Tools Platform (PTP)

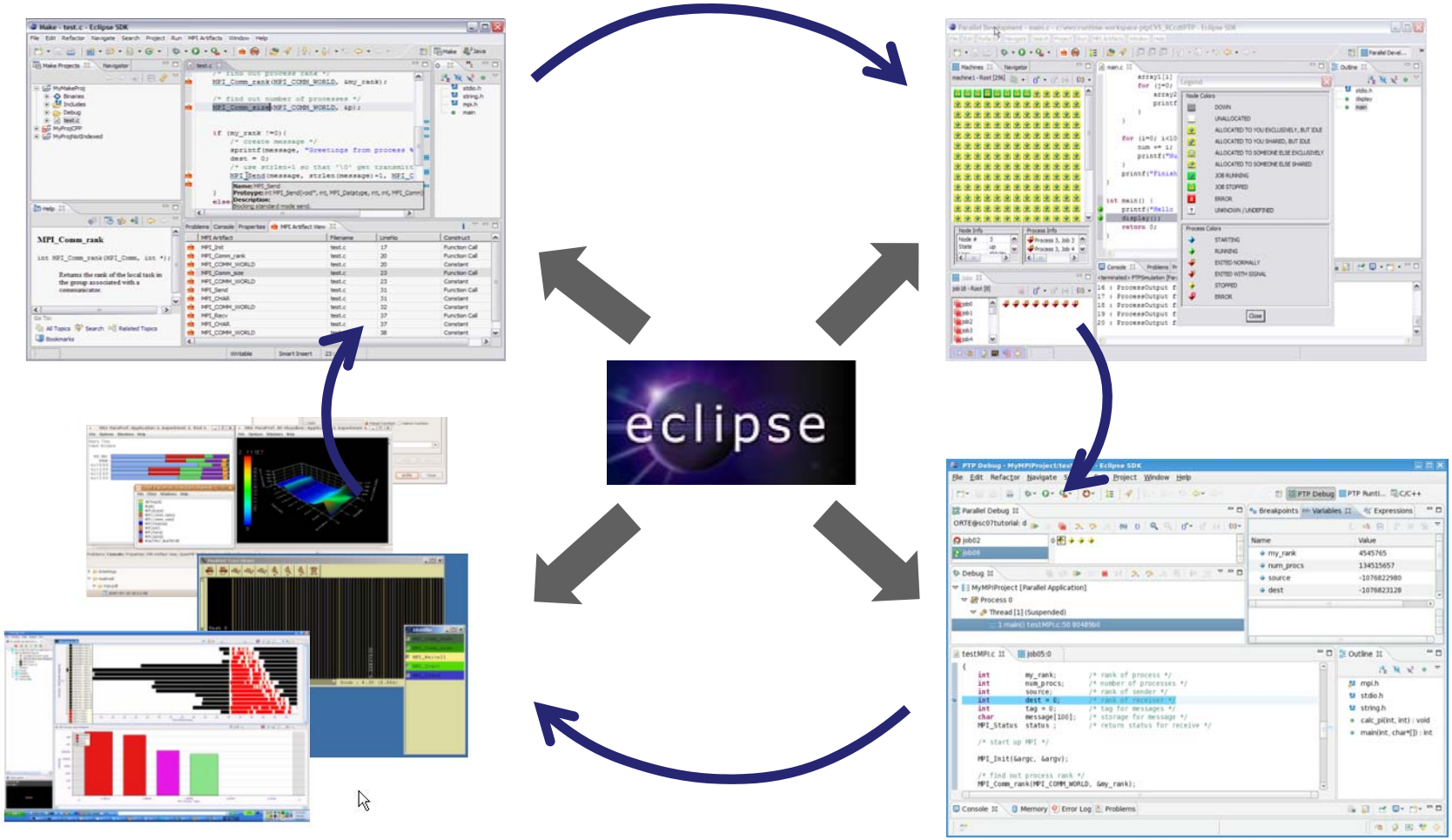
www.eclipse.org/ptp



Application Development in PTP

Coding & Analysis Tools

Launching & Monitoring Tools

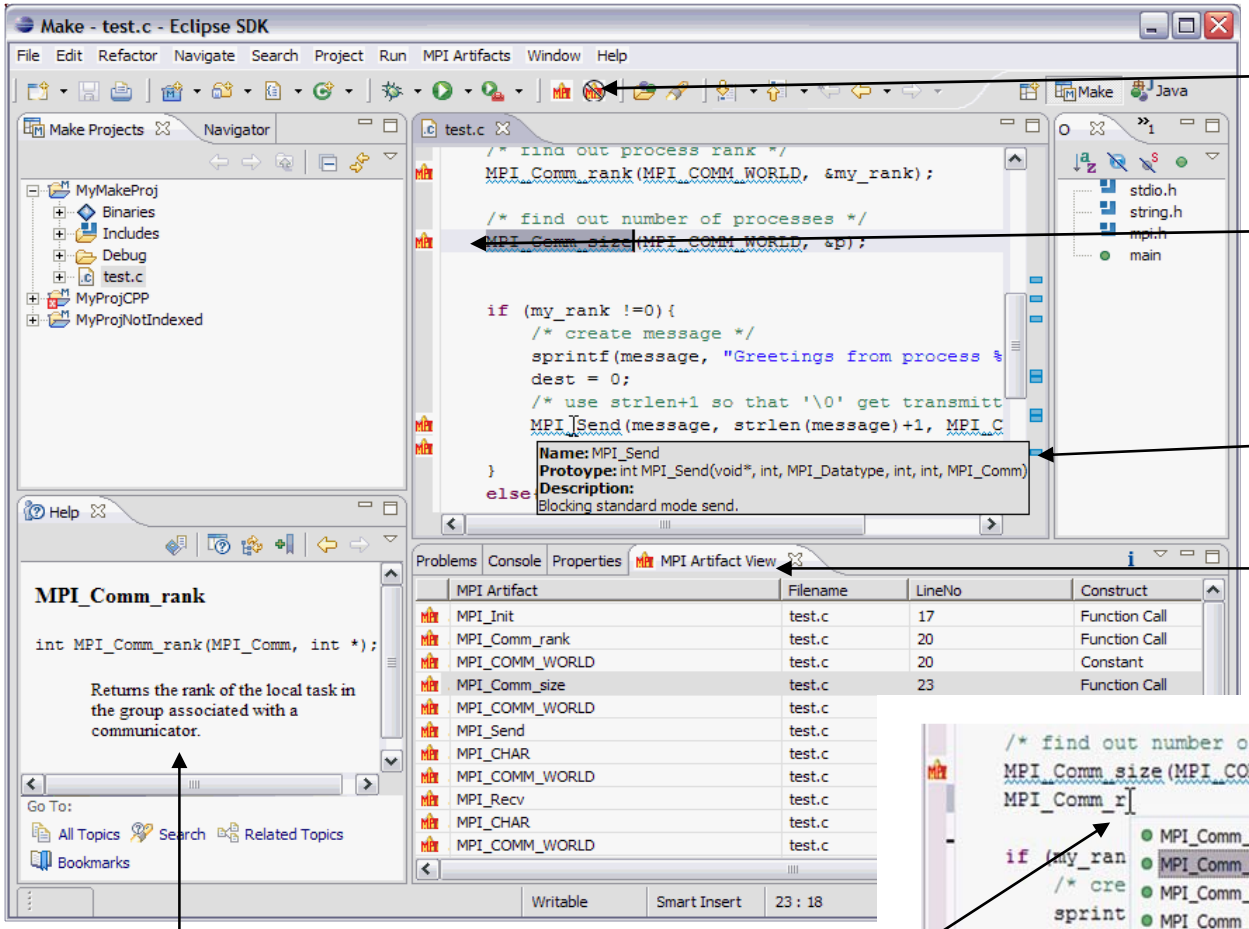


Performance Tuning Tools

Debugging Tools

Parallel Language Development Tools:

MPI Assistance Tools (similar Tools available for OpenMP, and UPC)

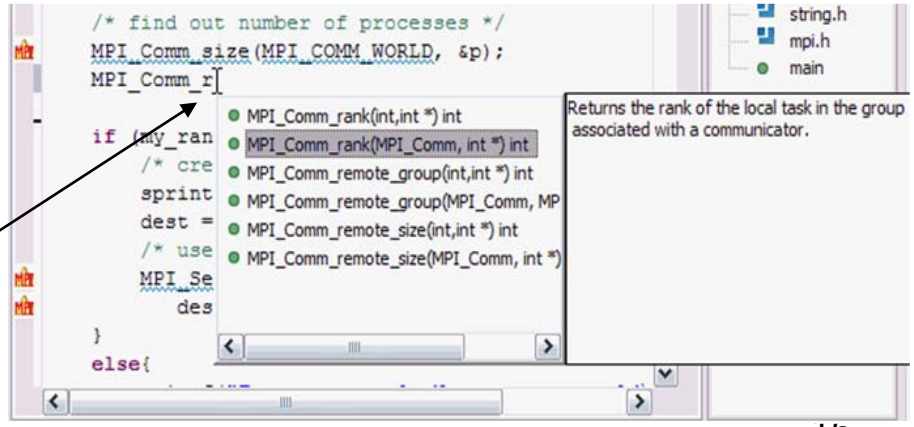


Actions to find MPI Artifacts

Source Markers for Navigation & ID

Mouse hover Help

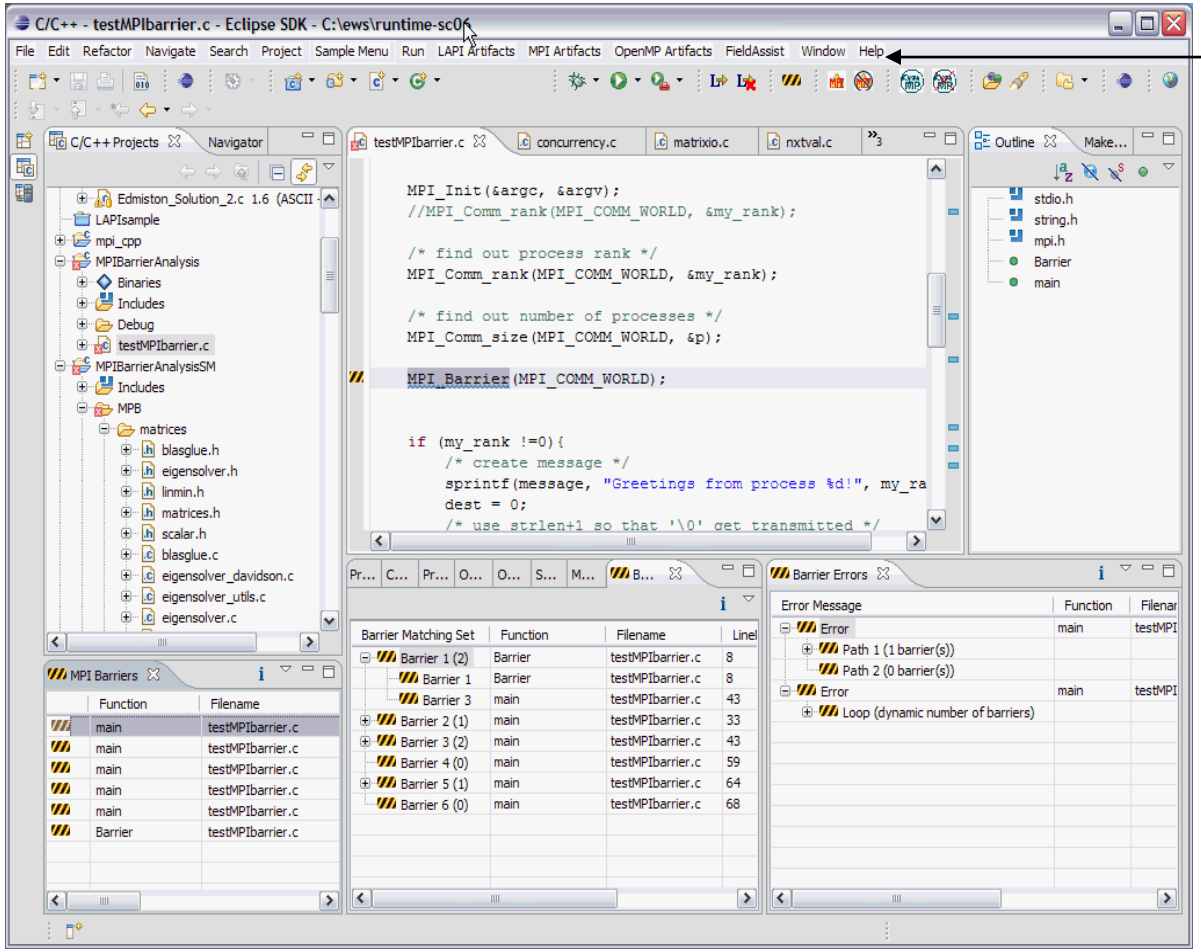
MPI Artifacts listing



Context Sensitive Help:
(F1) provides API info

Content Assist:
Ctrl-space
suggest completions

Parallel Language Development Tools:
Advanced Static Analysis: MPI Barrier Verification Tool

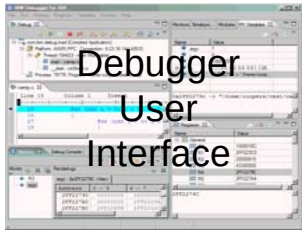


Action to run Barrier Verifier

Verify barrier synchronization in C/MPI programs

- Synchronization errors lead to deadlocks and stalls.
- Programmers may have to spend hours trying to find the source of a deadlock
- The MPI Barrier Verification Tool detects potential barrier deadlocks/stalls before the program executes

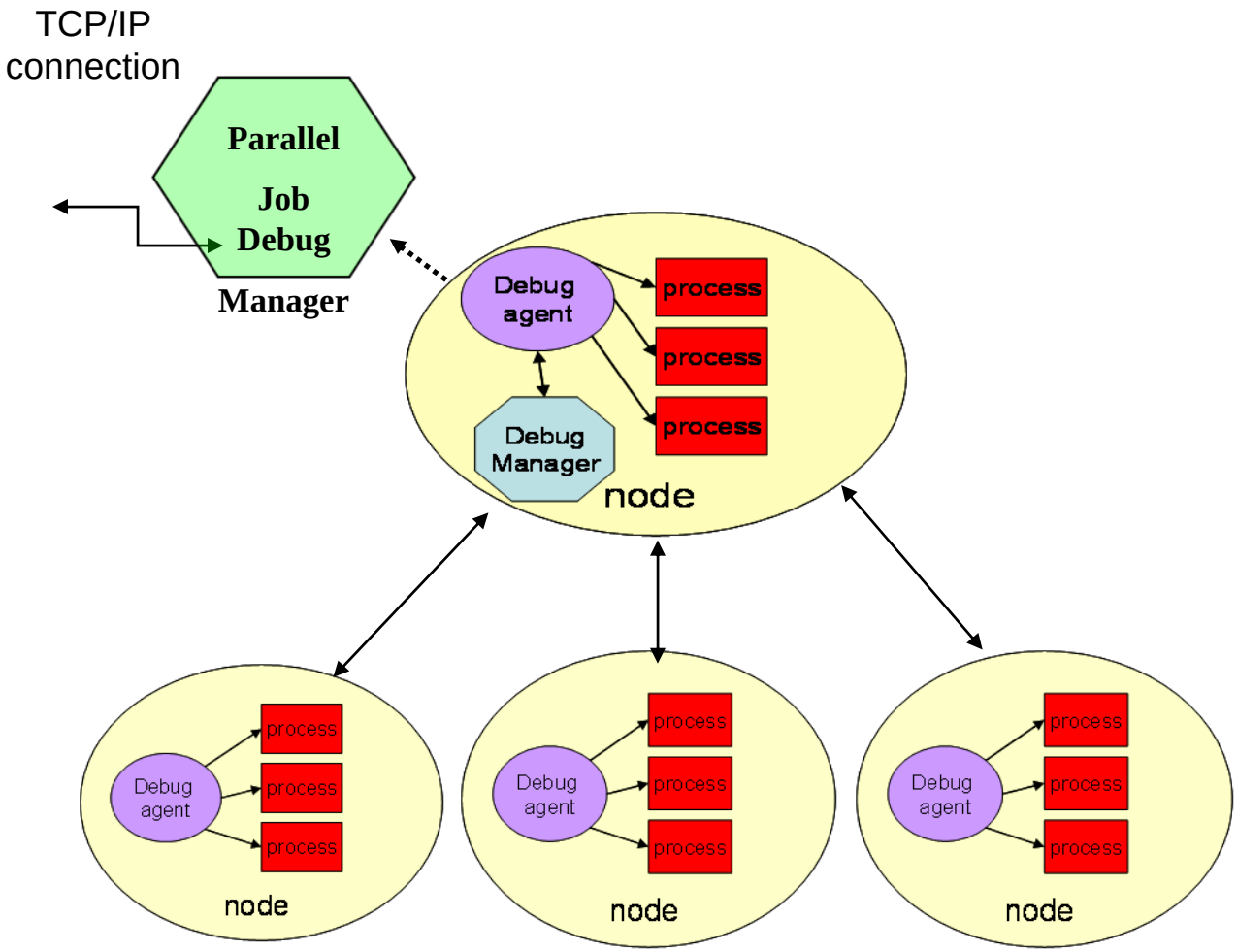
Parallel Debugger Architecture



Eclipse Debug Adaptor

Eclipse PTP

May run on local laptop



PTP Performance Tools Framework

Integration Framework:

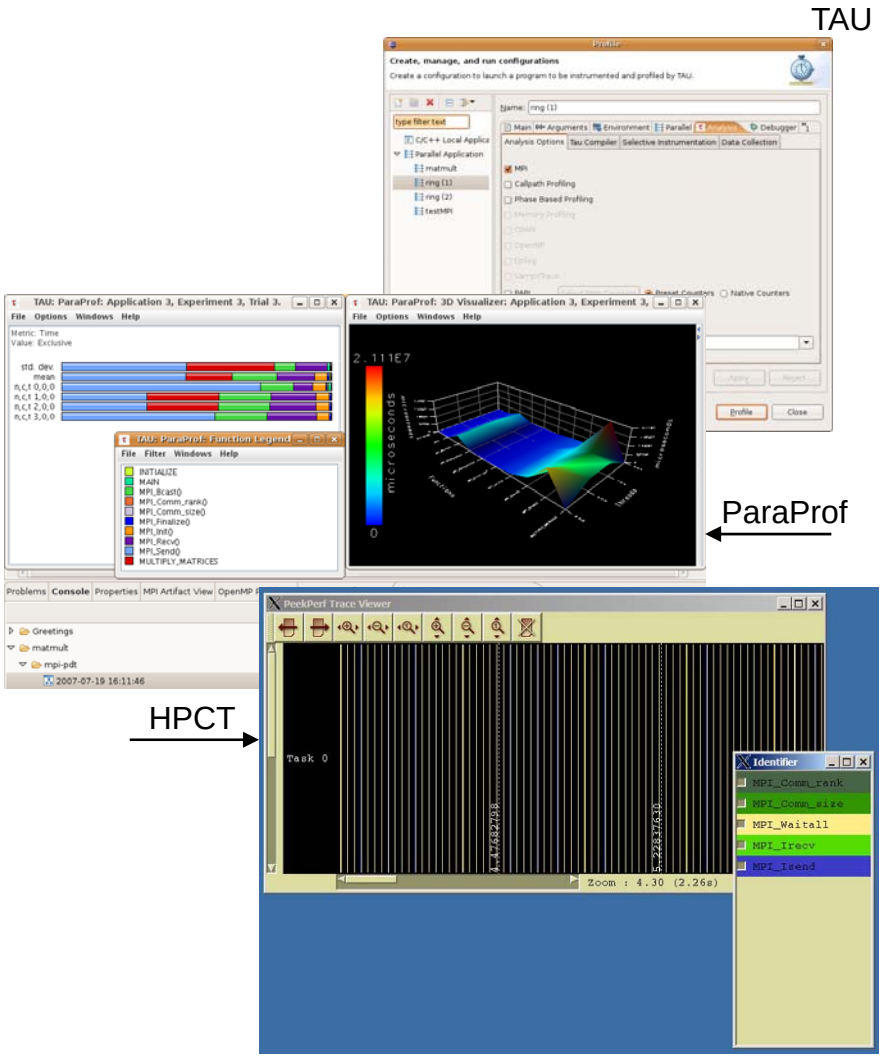
- Facilitate integration of existing performance tools into PTP
- Provide consistent & uniform user interfaces to simplify tool operation
- Reduce the “Eclipse plumbing” necessary to integrate these tools

Provide Eclipse integration for **<u>instrumentation, measurement</u>**, and **<u>analysis</u>**

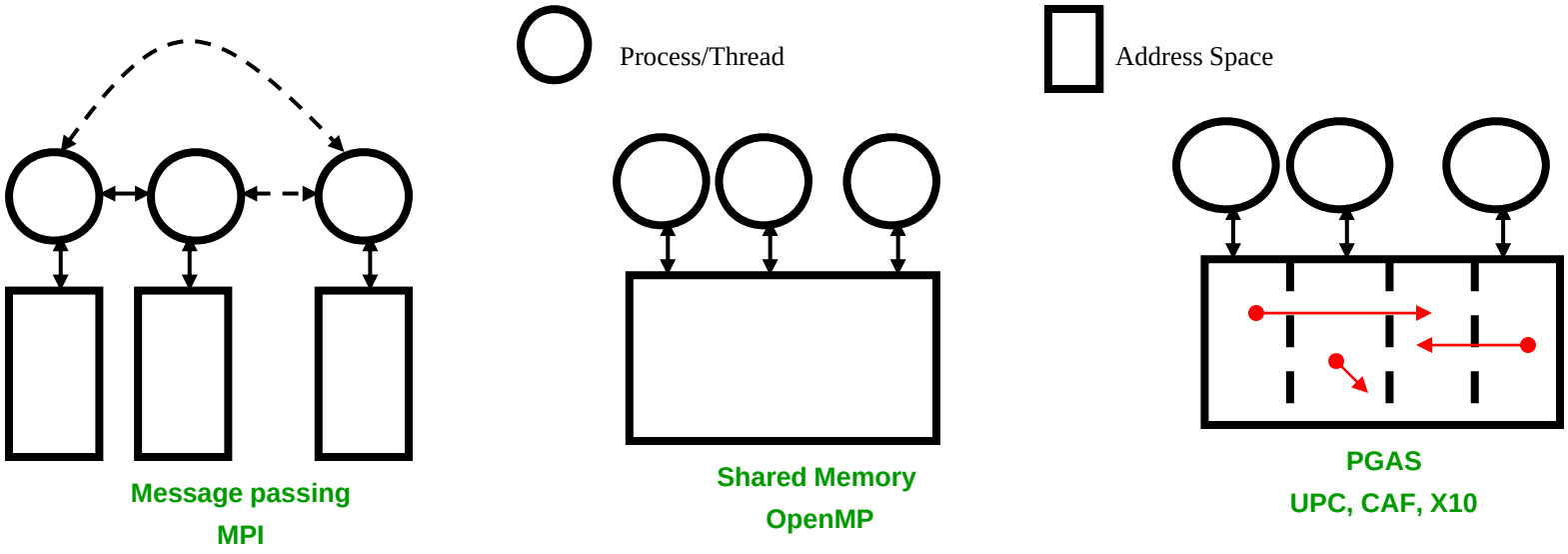
- Tools and tool workflows are specified in an XML file
- Tools are selected and configured by users in the launch configuration window
- Output is generated, managed and analyzed as specified in the workflow

Integration of HPCS Toolkit

- **Automated rules-based perf analysis**

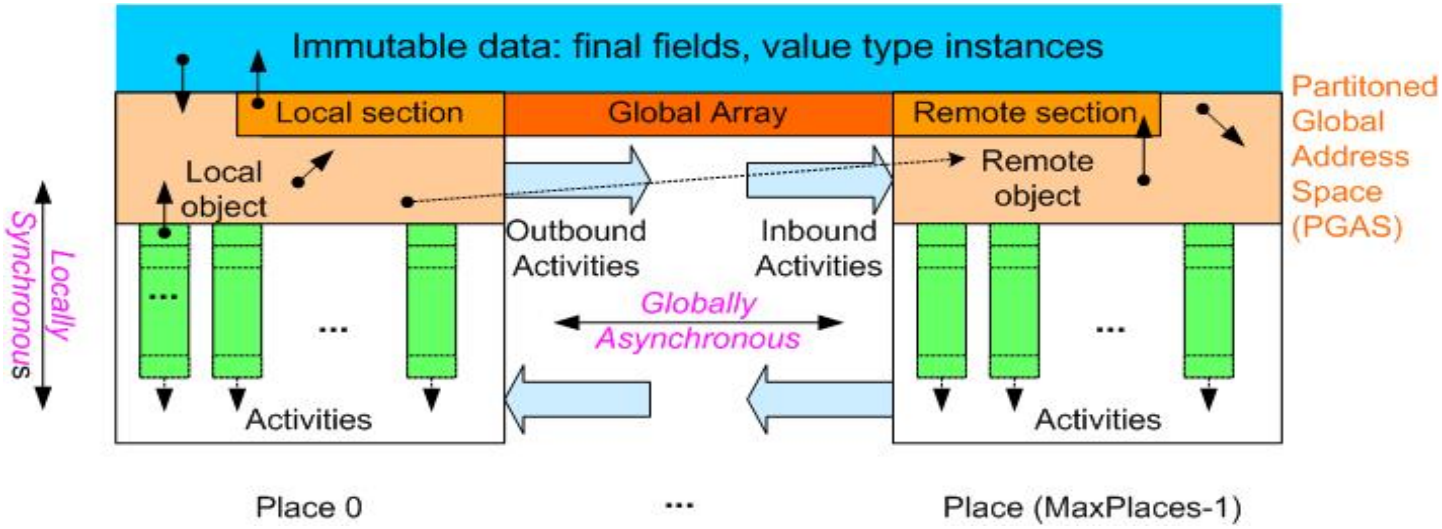


What is Partitioned Global Address Space (PGAS)?



- Computation is performed in multiple places.
- A place contains data that can be operated on remotely.
- Data lives in the place it was created, for its lifetime.
- A datum in one place may reference a datum in another place.
- Data-structures (e.g. arrays) may be distributed across many places.
- Places may have different computational properties

Asynchronous PGAS



- Asynchrony
  - Simple explicitly concurrent model for the user: **async (p) S** runs statement S "in parallel" at place p
  - Controlled through **finish**, and local (conditional) **atomic**
- Used for active messaging (remote asyncs), DMAs, fine-grained concurrency, fork/join concurrency, do-all/do-across parallelism
  - SPMD is a special case

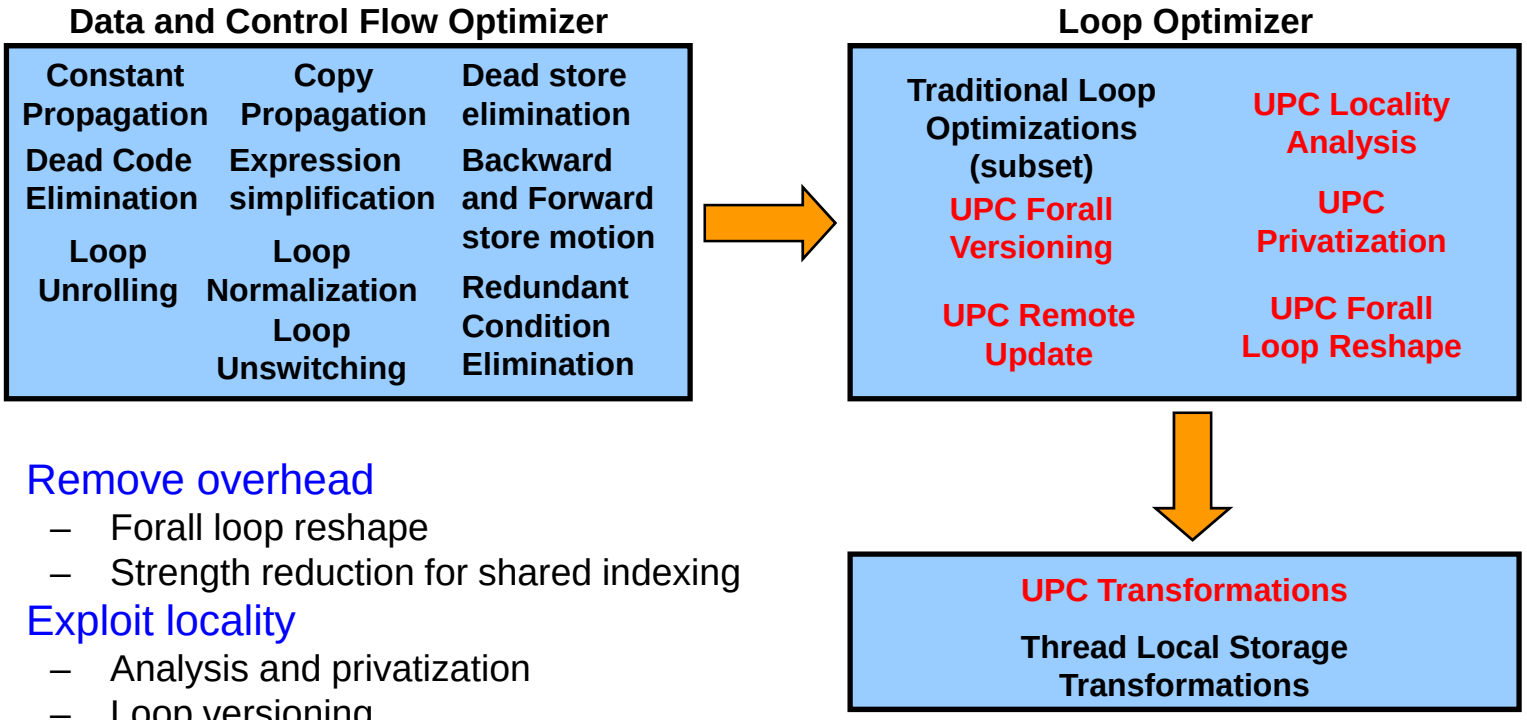
Concurrency is made explicit and programmable.

UPC Performance Gaps

- Data distributions
  - Express data locality and distribution
- Efficient single thread performance
  - Exploit existing, optimized serial libraries
  - Compiler optimizations: parallel loop, privatization
- Efficient and scalable communication
  - Collective operations
  - Compiler optimizations: communication scheduling and aggregation, hw exploit
- Fine grain threading for load balancing
- Synchronization
- Parallel I/O

Combination of system, runtime and compiler opts.

UPC Compiler Optimizations



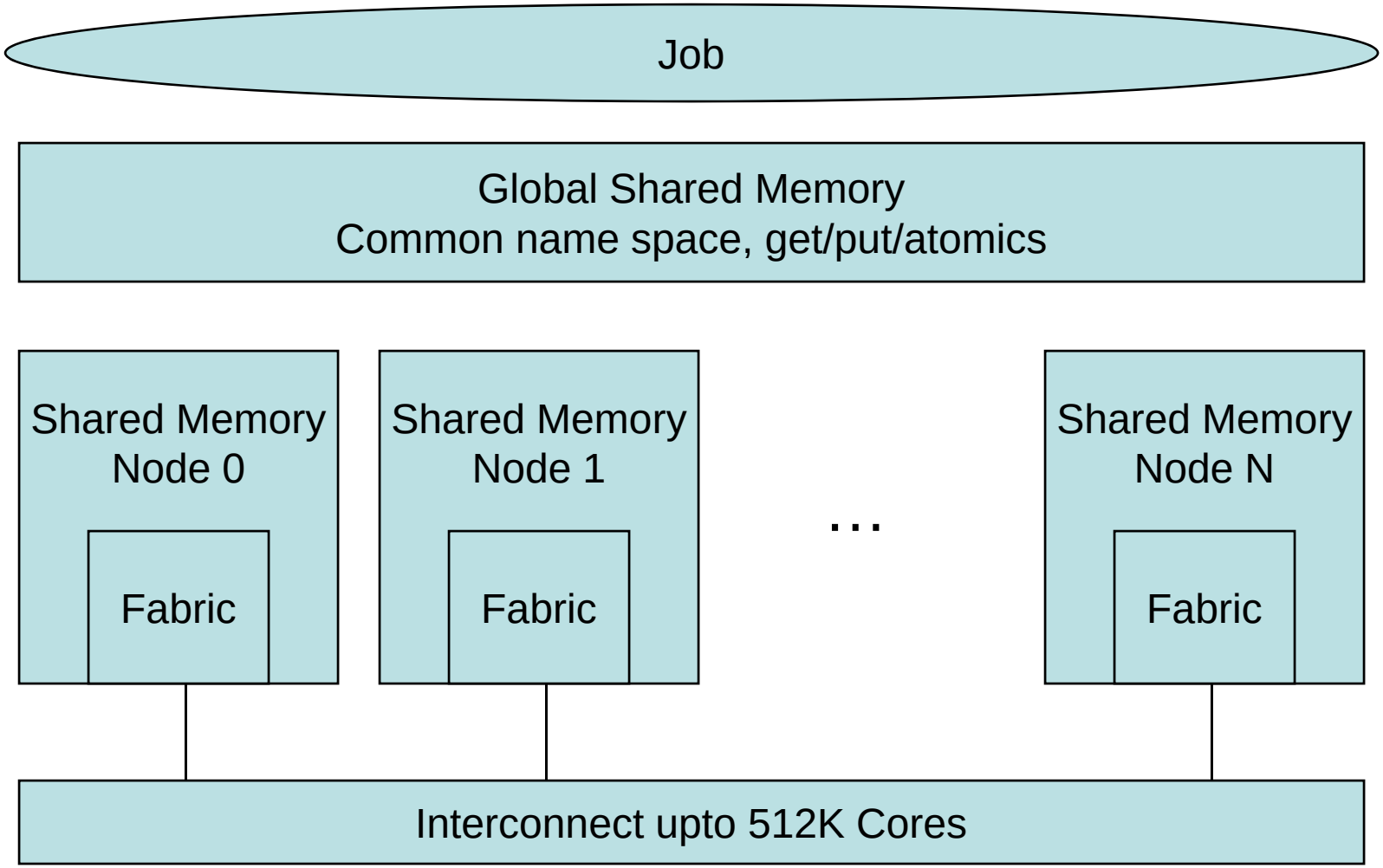
- Remove overhead
  - Forall loop reshape
  - Strength reduction for shared indexing
- Exploit locality
  - Analysis and privatization
  - Loop versioning
- Exploit hardware assist
  - GSM for remote update
  - Collectives hardware assist
- Reduce communication
  - Comm. aggregation and scheduling

Optimizer infrastructure applicable to other PGAS languages (Co-Array Fortran)

PERCS Hardware Innovations

- General Purpose POWER7
  - Common with commercial systems
- Integrated Storage and Networking
  - SAS2 disk enclosures and links
  - 10GigE links for direct connection to IP backbones
- Advanced HPC Inteconnect
  - Low diameter fabric with ultra low latency and high bi-section bandwidth
    - Single hop between groups of 1024 cores
    - Three hop routes between all 512K cores
  - Collective acceleration
  - Global shared memory access and atomics

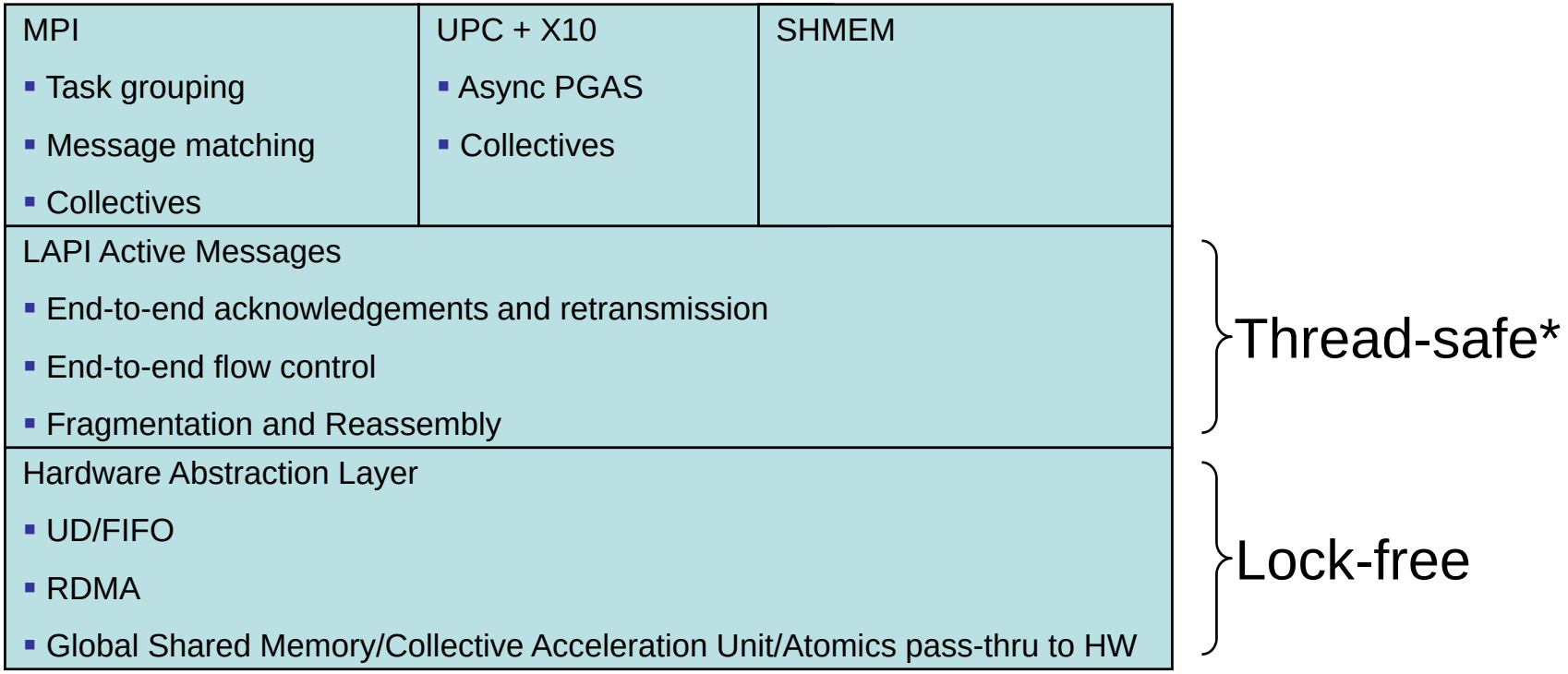
GSM Overview



Protocol Enhancements for Sustained Performance

- Communication latency:
  - Burst MMIO
  - Cache injection
  - Lock overhead reduction – lock-free option
  - Exploitation of Global Shared Memory
- Collective Communication overheads:
  - Collective Acceleration Unit
  - RDMA exploitation
- Memory latency:
  - Drive towards zero cache miss execution in the latency critical paths
- OS Jitter minimization:
  - Exploitation of Global Counters
  - OS hooks for scheduling low-priority threads and interrupts on secondary SMT threads
  - Co-scheduler to synchronize high-priority and low-priority windows

Communication Protocol Layers



* Lock-free and semi-reliable options under investigation

OS Enhancements for Sustained Performance

- Dynamic Variable Page Size Support:
  - OS support for multiple page sizes
  - Dynamically change page size for a running application's need
- APIs to Control System Resources
  - Control application memory usage
  - Control CPU allocation
- 64-bit I-node:
  - Enable OS to support trillions of files per file system
- OS Jitter Minimization:
  - OS hooks to scheduling non critical threads to secondary SMT threads
- Checkpoint/Restart Support:
  - Creating lightweight container technology
    - Called WPAR in AIX
    - Working on adding virtualization hooks into Linux kernel
- Help Define/Configure Lightweight Compute OS
  - Provide a list of non essential daemon/services to turn off on compute nodes
- APIs to Hardware Counters
  - System
  - Network