UCRL-PRES-231061

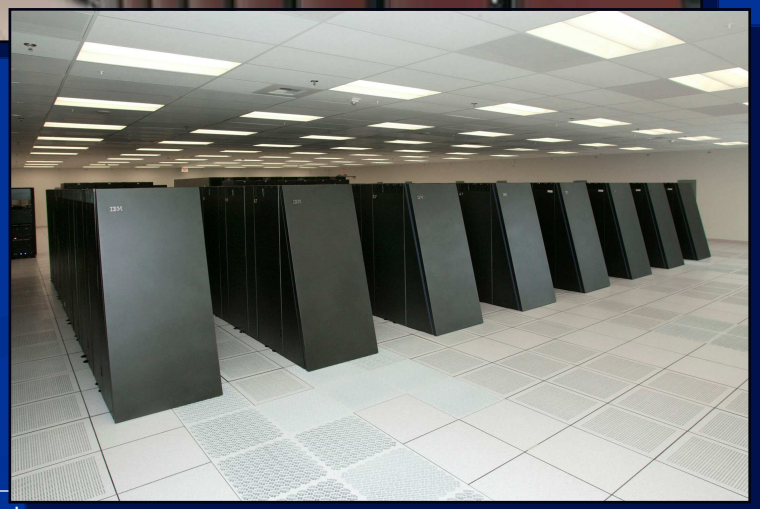
Developing New Tool Strategies for Scalable HPC Systems

Martin Schulz
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

Systems Research Seminar
NC State University, May 21st 2007

Trend towards Petascale

- Growing HPC Systems
  - *Many system at or above 10,000 cores*
  - *Multi/Many-core chips*
- TOP 500 list
  - *Nov 2005: **140** machines at or above 1024 cores*
  - *Nov 2006: **302** machines at or above 1024 cores*
- Petascale era is coming



Blue Gene/L

(compare this with a 1988 Cray YMP/8 at 2.7 GF/s)

~11mm

Compute Chip

2 processors
700 MHz
2.8/5.6 GF/s
4 MB eDRAM

Compute Card
I/O Card

FRU (field replaceable unit)
25mmx32mm
2 nodes (4 CPUs)
(2x1x1)
5.6/10.2 GF/s
1024 MB Mem

Node Card

16 compute cards
0-2 I/O cards
32 nodes
(64 CPUs)
(4x4x2)
89.6/179.2 GF/s
16 GB Mem

Cabinet

2 midplanes
1024 nodes
(2,048 CPUs)
(8x8x16)
2.9/5.7 TF/s
512 GB Mem
15-20 kW

System

64 cabinets
65,536 nodes
(131,072 CPUs)
(32x32x64)
183.5/367 TF/s
32 TB Mem
1.2 MW
2,500 sq.ft.
MTBF 6.16 Days

#1 of TOP 500 list

Multiscale Simulations

Time Scale

s

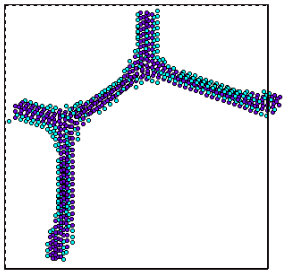
ms

µs

ns

ps

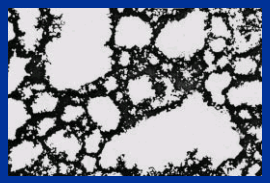
Atomic Scale

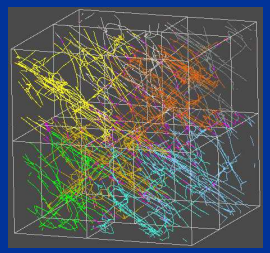


Molecular Dynamics

Unit mechanisms of defect mobility and interaction

Microscale





Dislocation Dynamics

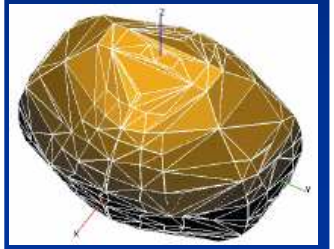
Collective behavior of defects, single crystal plasticity

Mesoscale

Aggregate Materials

Aggregate grain response, poly-crystal plasticity

Continuum



Finite Element

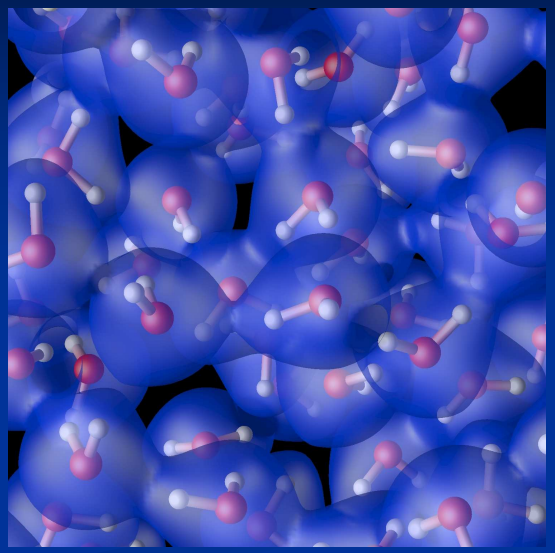
Plasticity of complex shapes

BlueGene/L simulations bring *qualitative* change to ASC material and physics modeling and engineering

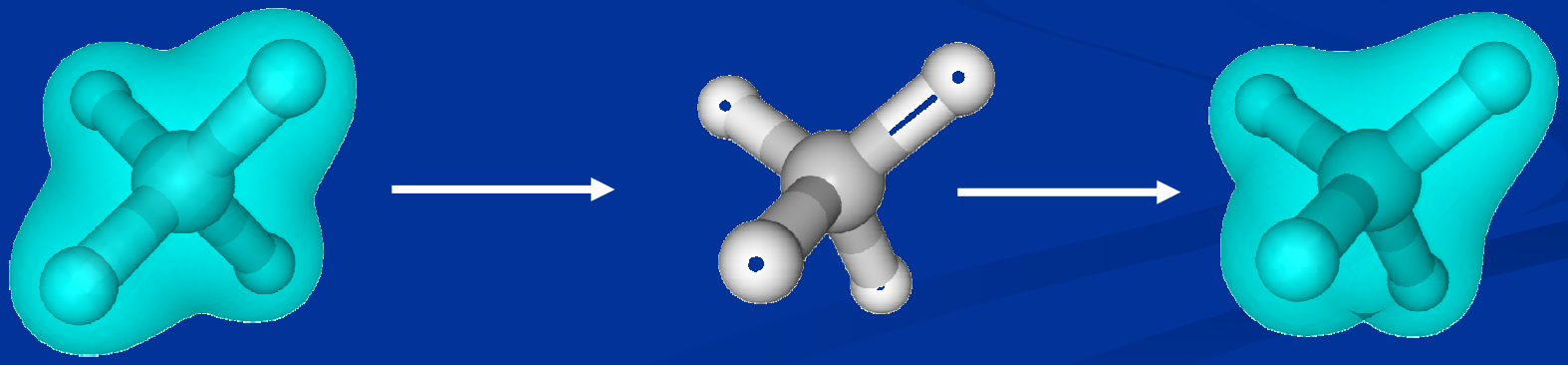
nm µm mm Length Scale

Example: QBox

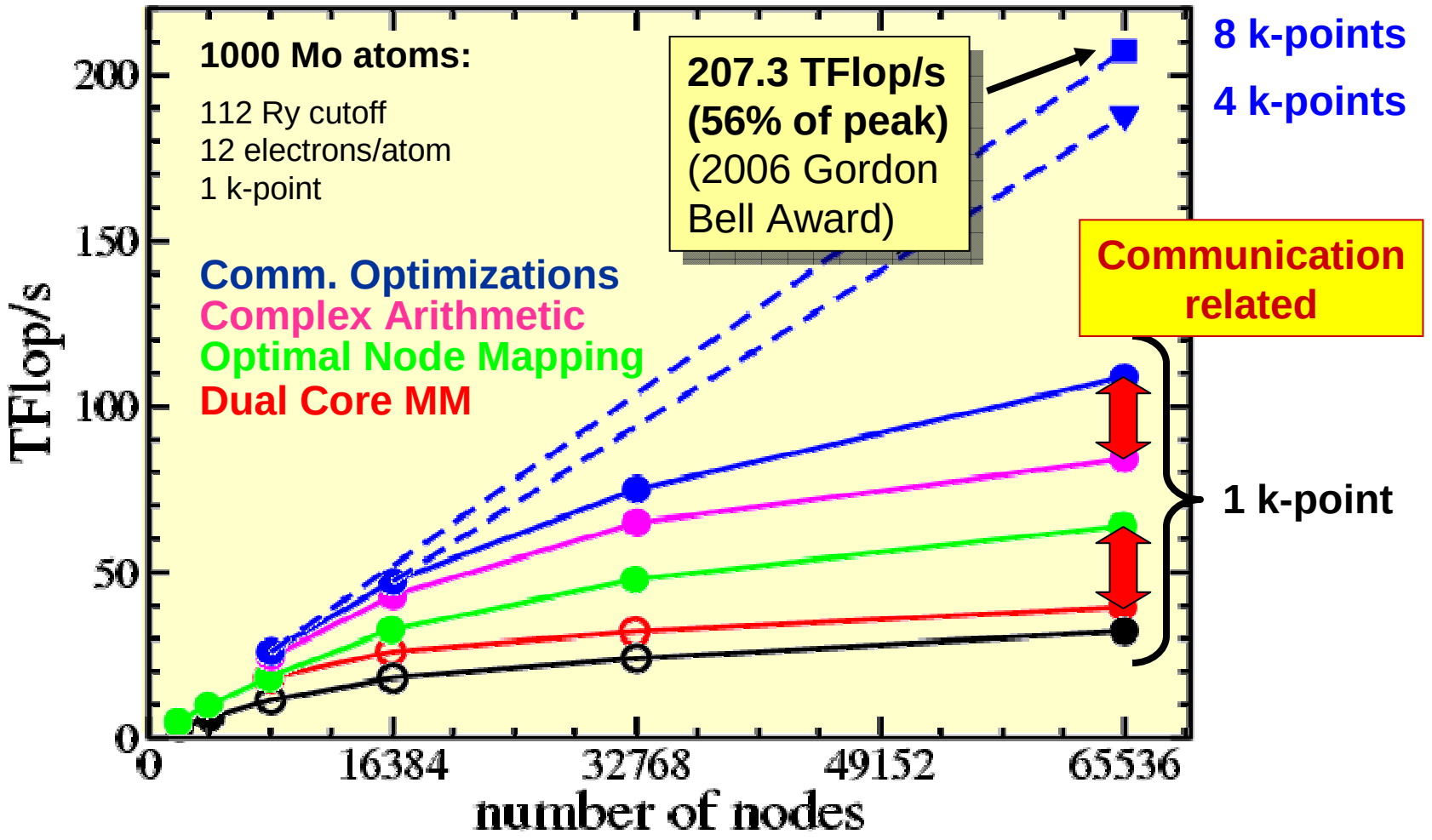
- Material Simulation
- First Principles Method
  - *No empirical parameters*
  - *Chemically dynamic*
    - *Iterative process*
    - *Computationally intensive*



Electron density surrounding water molecules, calculated from first-principles

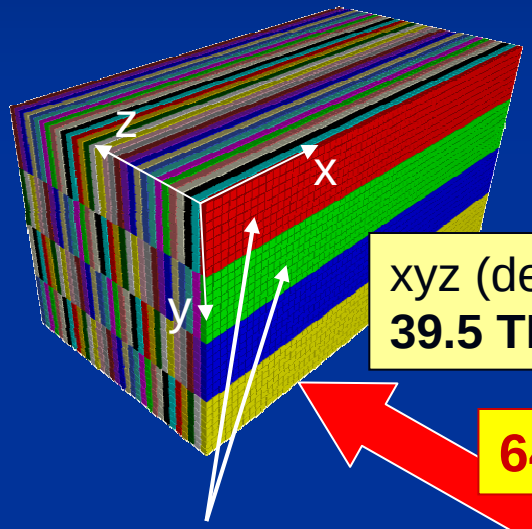


QBox Performance



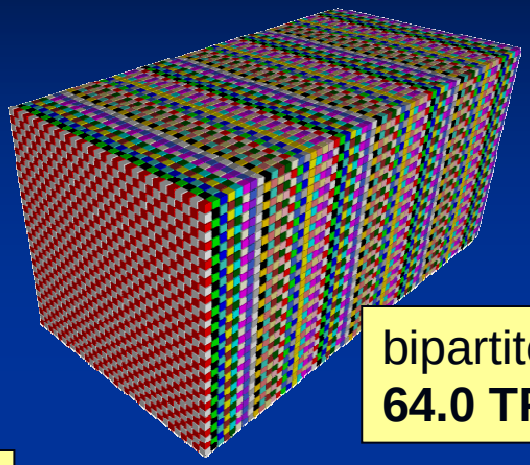
Topology Impact

65536 nodes, in a 64x32x32 torus

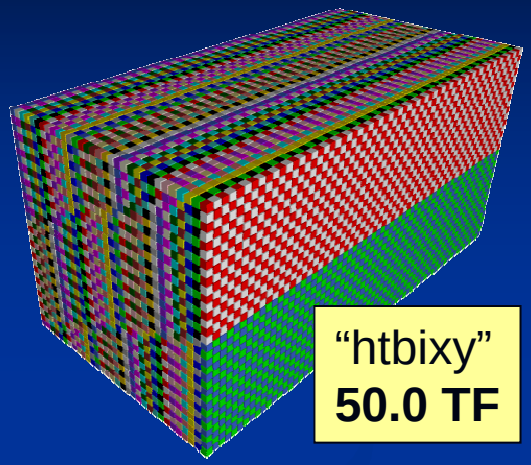


xyz (default)
39.5 TF

512 tasks per MPI subcommunicator

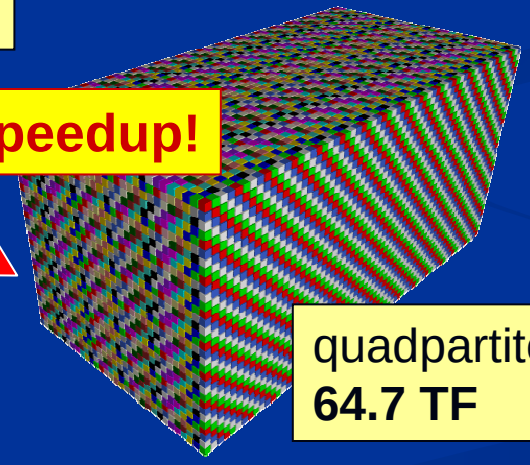


bipartite
64.0 TF

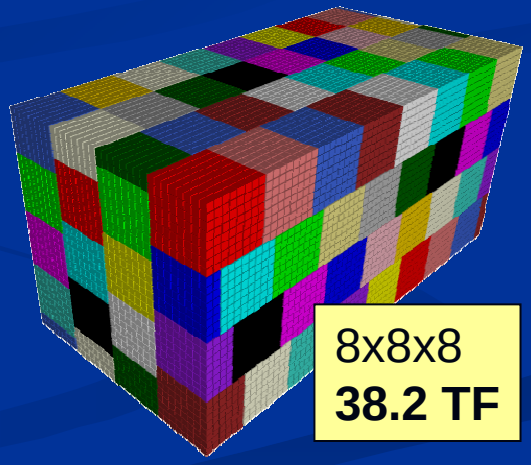


“htbixy”
50.0 TF

64% speedup!



quadpartite
64.7 TF



8x8x8
38.2 TF

Physical task distribution can significantly affect performance

Need for Scalable Tools

- Support complete development cycle
  - *Debugging*
  - *Performance analysis*
  - *Optimization/transformation*
- New challenges with scalability
  - *Large volumes of data to store and analysis*
  - *Central processing/control infeasible*
  - *Light-weight kernel*
- New tool strategies
  - *Scalable infrastructures*
  - *Application specific tool support*
  - *Flexible and interoperable toolboxes*

Outline

- ASC Tools Projects
- Scalable Debugging
  - *Challenges*
  - *Stack Trace Analysis*
- MPI Tool Infrastructure
  - *Layering MPI Tools*
  - *Communicator Specific Profiling*
- Lessons Learned & Future Work
- Conclusions

ASC Tools: Common Thread

- Scalability
  - *Tree-based data collection/processing*
  - *Intelligent data storage & compression*
- Tool components for quick prototyping
  - *Quickly assess new situations*
  - *Application specific tools*
- Ease-of-Use
  - *Little to no code modifications*
  - *Reuse existing, well-known tools*
  - *Keep learning curve low*
- Open source / tool integration

ASC Tool Projects

- Debugging/Correctness
  - *Stack Trace Analysis*
  - *Code Coverage*
  - *MPI Correctness*
  - *TotalView Collaboration*
- Performance Tools
  - *Open|SpeedShop*
  - *Tool Gear*
  - *MPI Profiling*
  - *Extended gprof*
- Static Analysis
  - *ROSE*
- Memory Tools
  - *ValGrind Collaboration*
  - *Memory Tracing*
- Fault Tolerance
  - *(MPI) Checkpointing*
  - *Soft Error Analysis*
- Infrastructures
  - *DPCL successor*
  - *P^NMPI*
  - *Trace Compression*
  - *Open Trace Format*

Debugging Challenges

TotalView on BG/L – 4096 Processes	
Operation	**Latency**
Single step	~15-20 secs.
Breakpoint Insertion	~30 secs.
Stack trace sampling	~120 secs.

Typical debug session includes many interactions

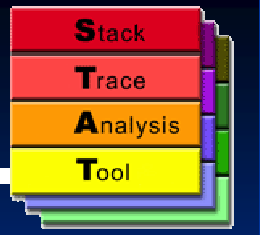
4096 is only 3% of BG/L!

Scalable Debugging

- Large volumes of debug data
- Single frontend for all node connections
- Centralized data analysis
- Vendor licensing limitations

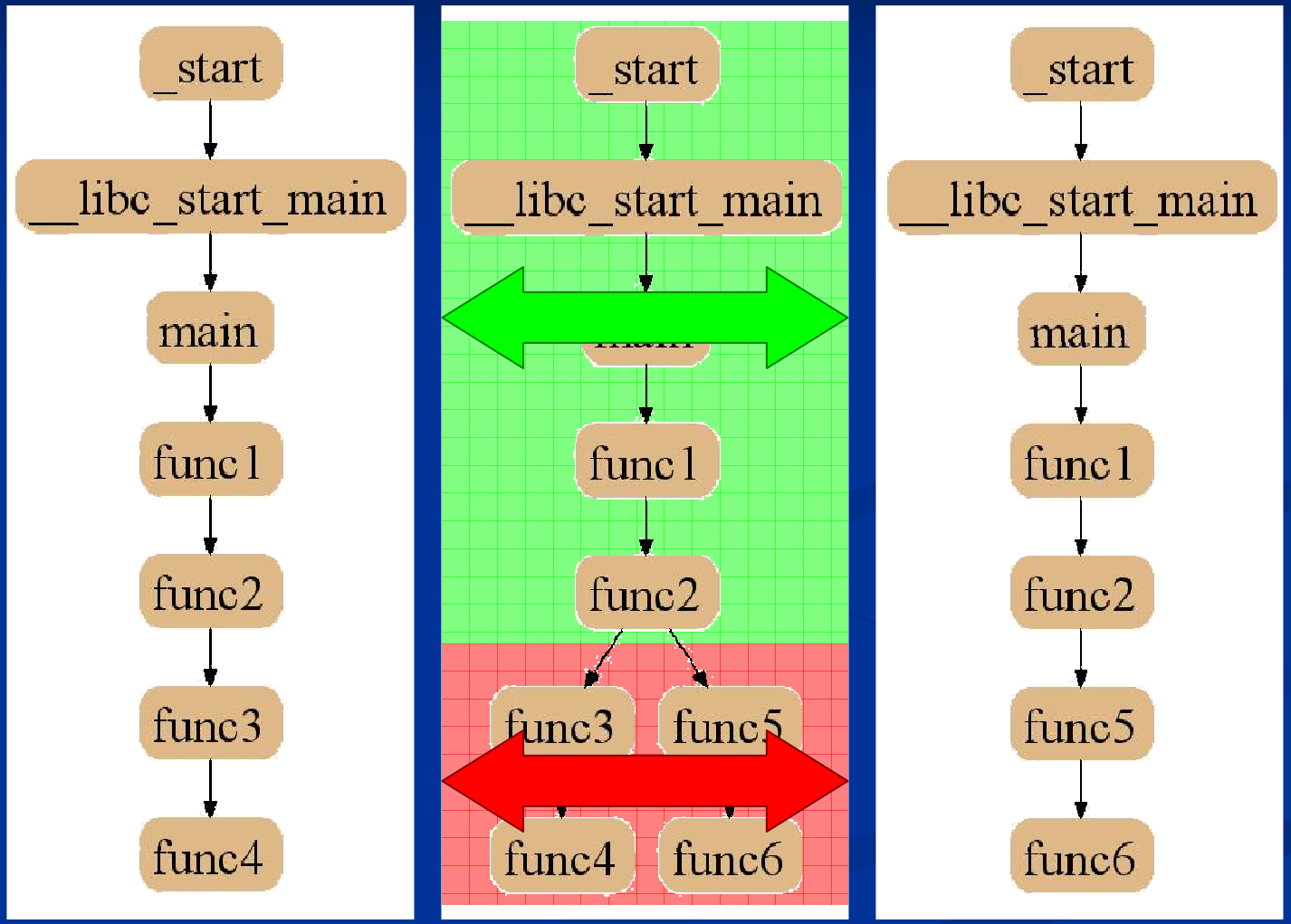
- Approach: scalable, lightweight debugger
  - *Discover equivalent process behavior*
  - *Reduce exploration space to small subset*
  - *Full-featured debugger for deeper digging*
  - *Analysis in tree-based reduction network*

STAT Approach

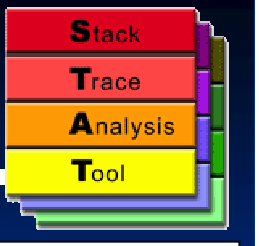


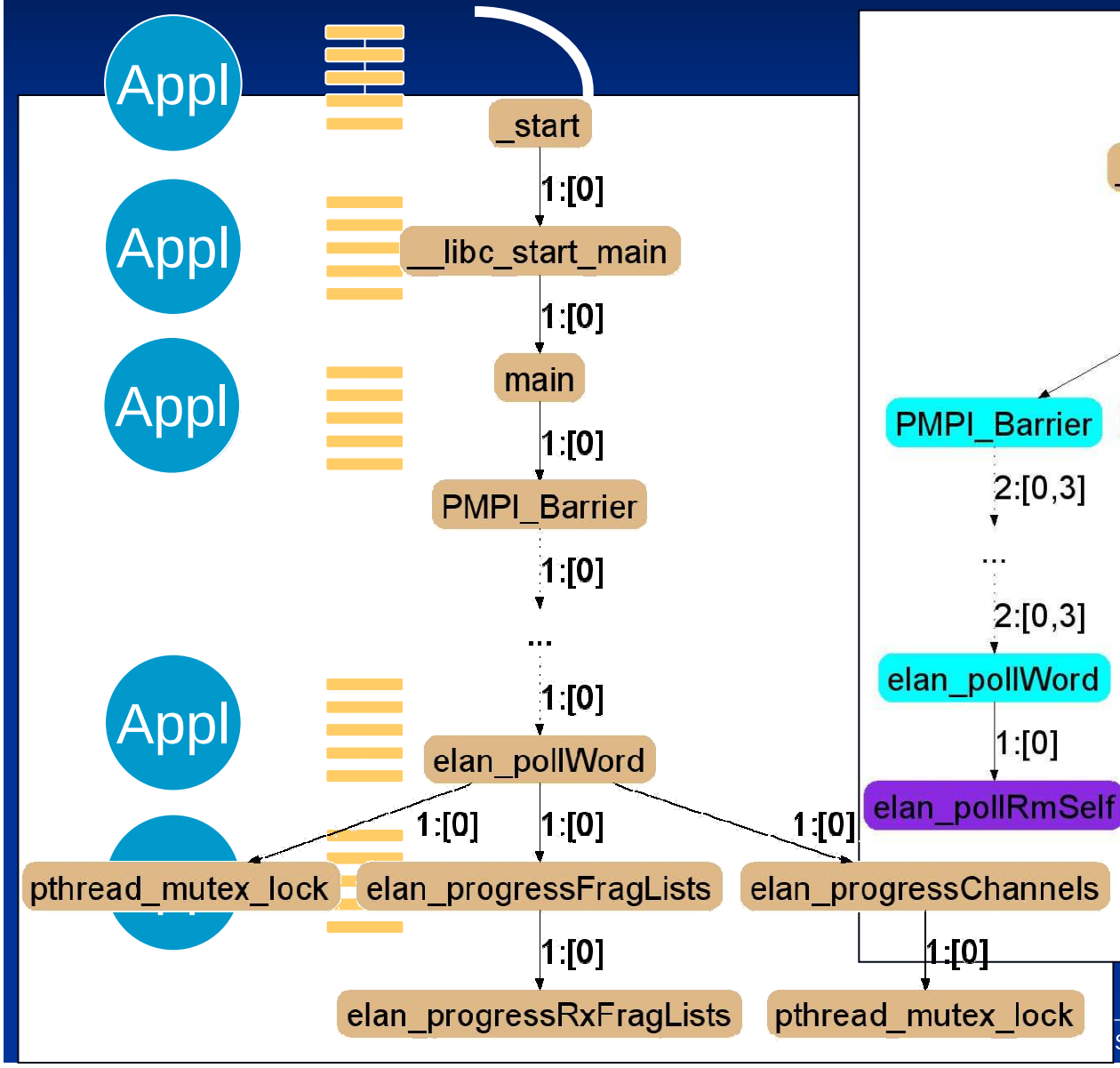
- Sample application stack traces
  - *Multiple snapshots across time and space*
  - *Through third party (DynInst) interface*
  - *Stored in graph representation*
- Create call graph prefix tree
  - *Only merge nodes with identical stack backtrace*
  - *Retains context information*
  - *First merge per node, then across nodes*
- Export in widely used graph formats
  - *Use existing viewer*

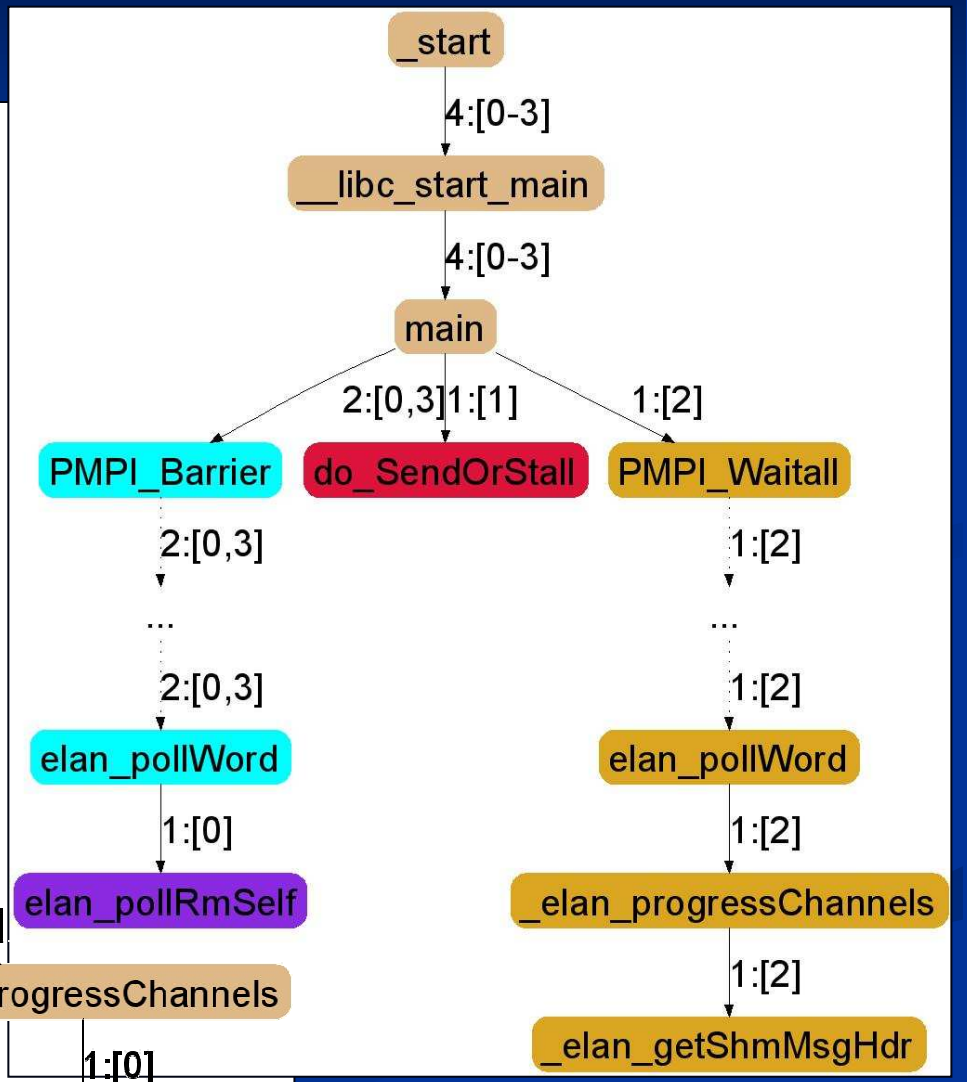
Stack Prefix Trees



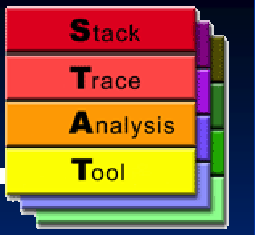
2D Trace Analysis







3D-Trace/Space/Time Analysis



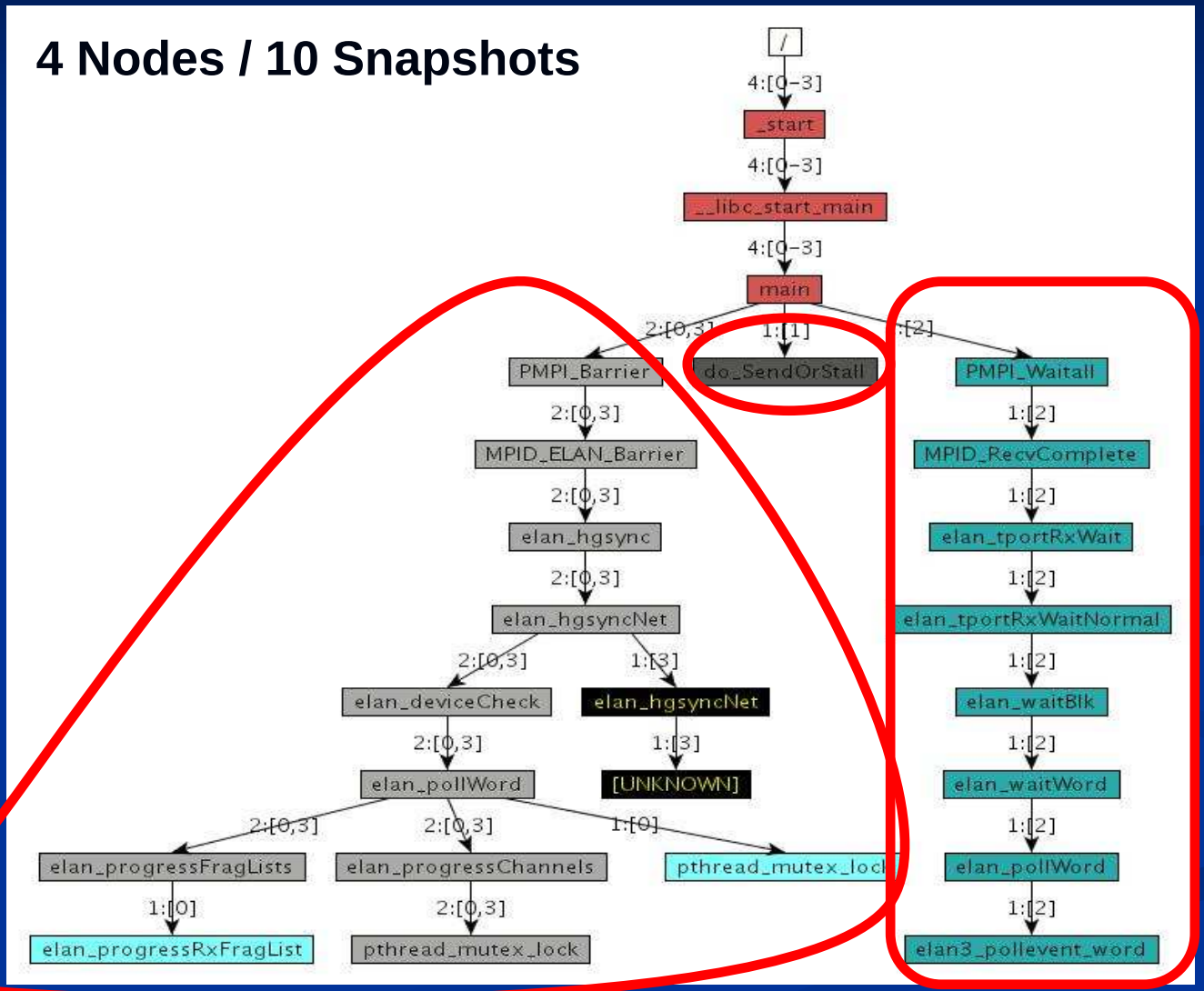
Appl

Appl

Appl

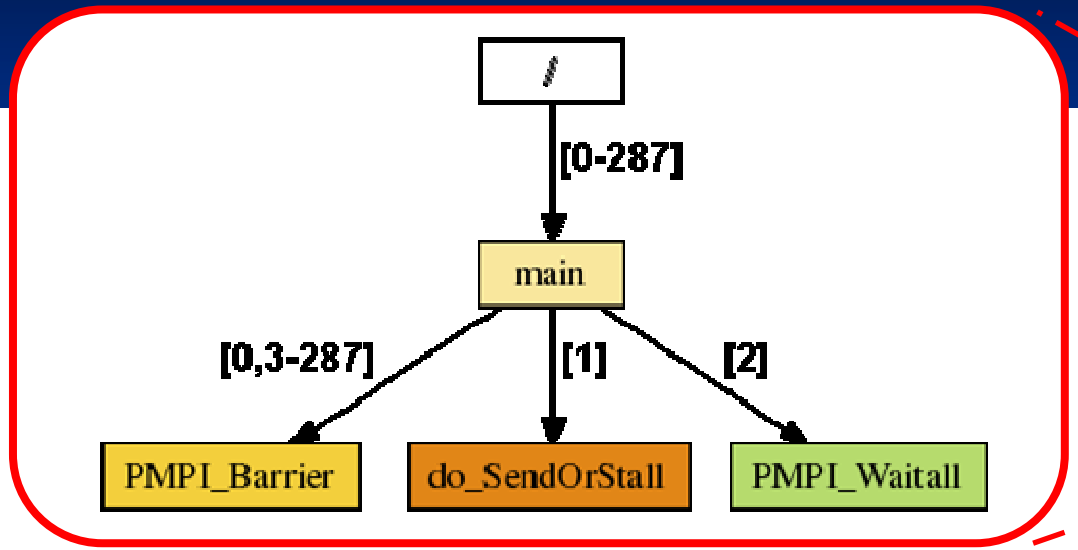
Appl

Appl

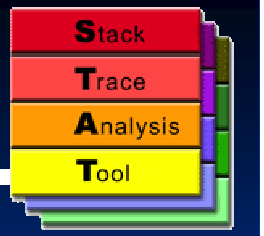


3D-Trace/Space/Time Analysis

Stack Trace Analysis Tool

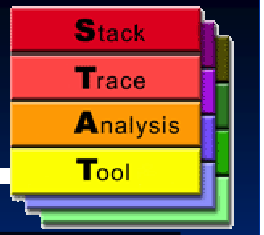


Achieving Scalability

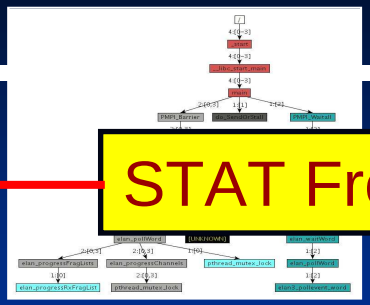


- Lower overhead than full debugger, but …
  - *Need to aggregate across all nodes*
  - *Bottleneck on tool front end*
  - *Traditional designs will not scale*
- Tree based reduction
  - *Merge partial trees on the fly*
  - *Implementation using MRNet*
- STAT components
  - *Backend (BE) daemons gathering traces*
  - *Communication processes merging prefix trees*
  - *Frontend (FE) tool storing the final graph*

Work and Data Flow



trace(count, freq.)



FE

STAT Frontend

Tree Merge

CP

CP

MRNet Communication Process

CP

CP

STAT Tool Daemon

BE

M M M MPI

BE

M M M MPI

. . .

BE

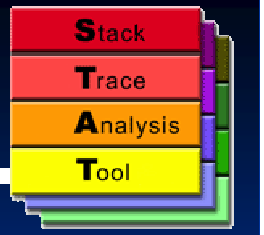
M M M MPI

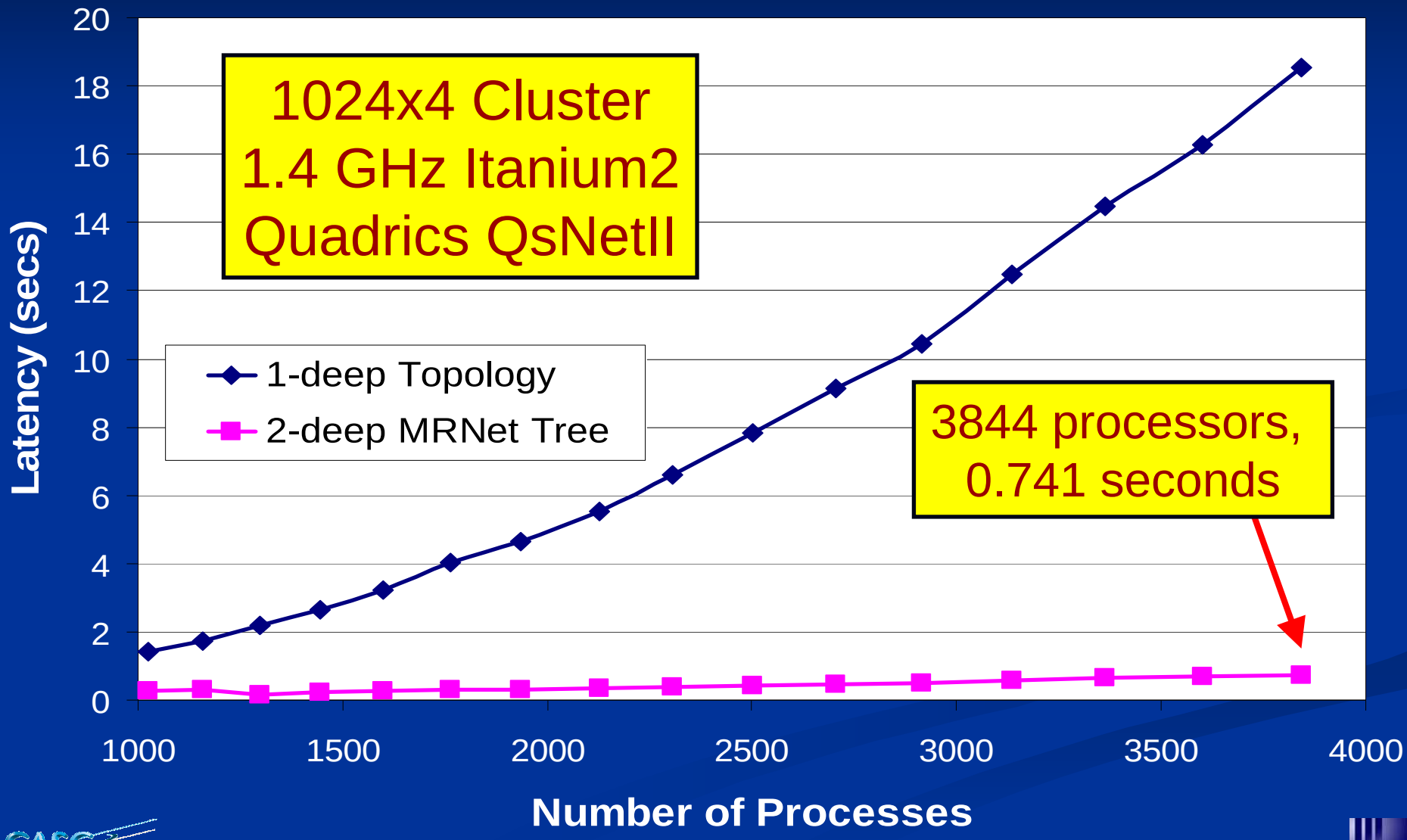
BE

M M M MPI

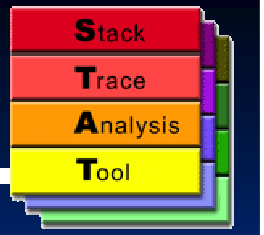
Application Processes

STAT Performance





STAT Summary



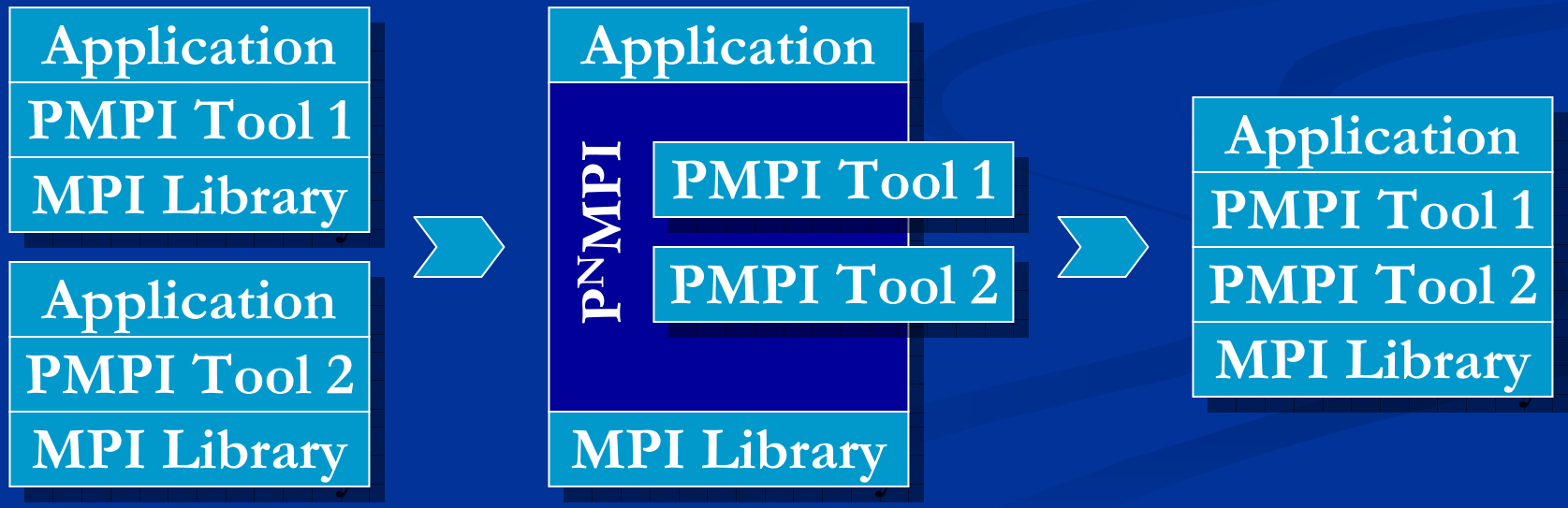
- Scalable Stacktrace Analysis
  - *Lightweight tool to identify process classes*
  - *Aggregation in Time and Space*
  - *Guide the use of full featured debuggers*
- Reduction network
  - *Based on merged callgraph prefix trees*
  - *Process merge operation inside the tree*
- *More Information:*
  - Stack Trace Analysis for Large Scale Debugging *Arnold, Ahn, de Supinski, Lee, Miller, Schulz IPDPS 2007*
  - *http://www.paradyn.org/STAT*

General Tool Infrastructures

- Application Specific Tools
  - *Adjust to special scenarios*
  - *Analyze and Optimize one target code*
  - *Prototypes for more general tools*
- Need the ability for quick prototypes
  - *Reuse existing components*
  - *Dynamically assemble tools*
  - *Specialize existing tools*
- MPI Tools
  - *Successful interface: PMPI*
  - *No support for cooperation or integration*
  - *All tools have a global scope*

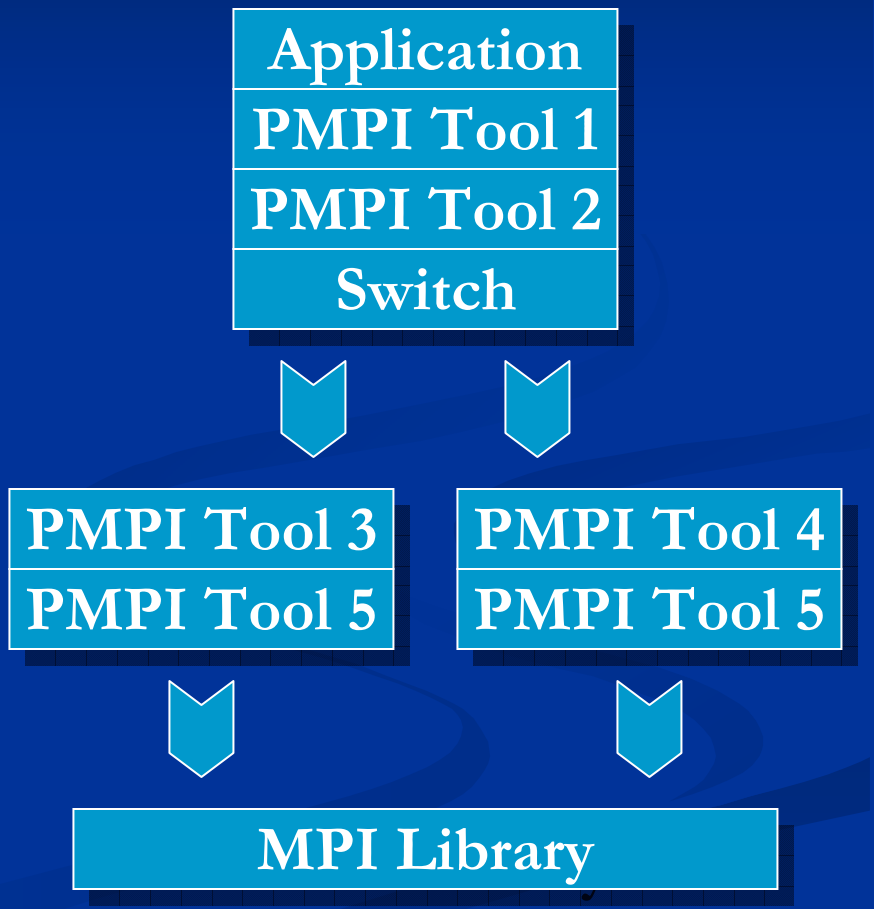
PNMPI Infrastructure

- MPI Tool Infrastructure
  - *Maintain compatibility with PMPI interface*
  - *Dynamic creation of tool stacks*
  - *Transparent to end & tool user*
  - *Plug-ins binary compatible with PMPI tools*

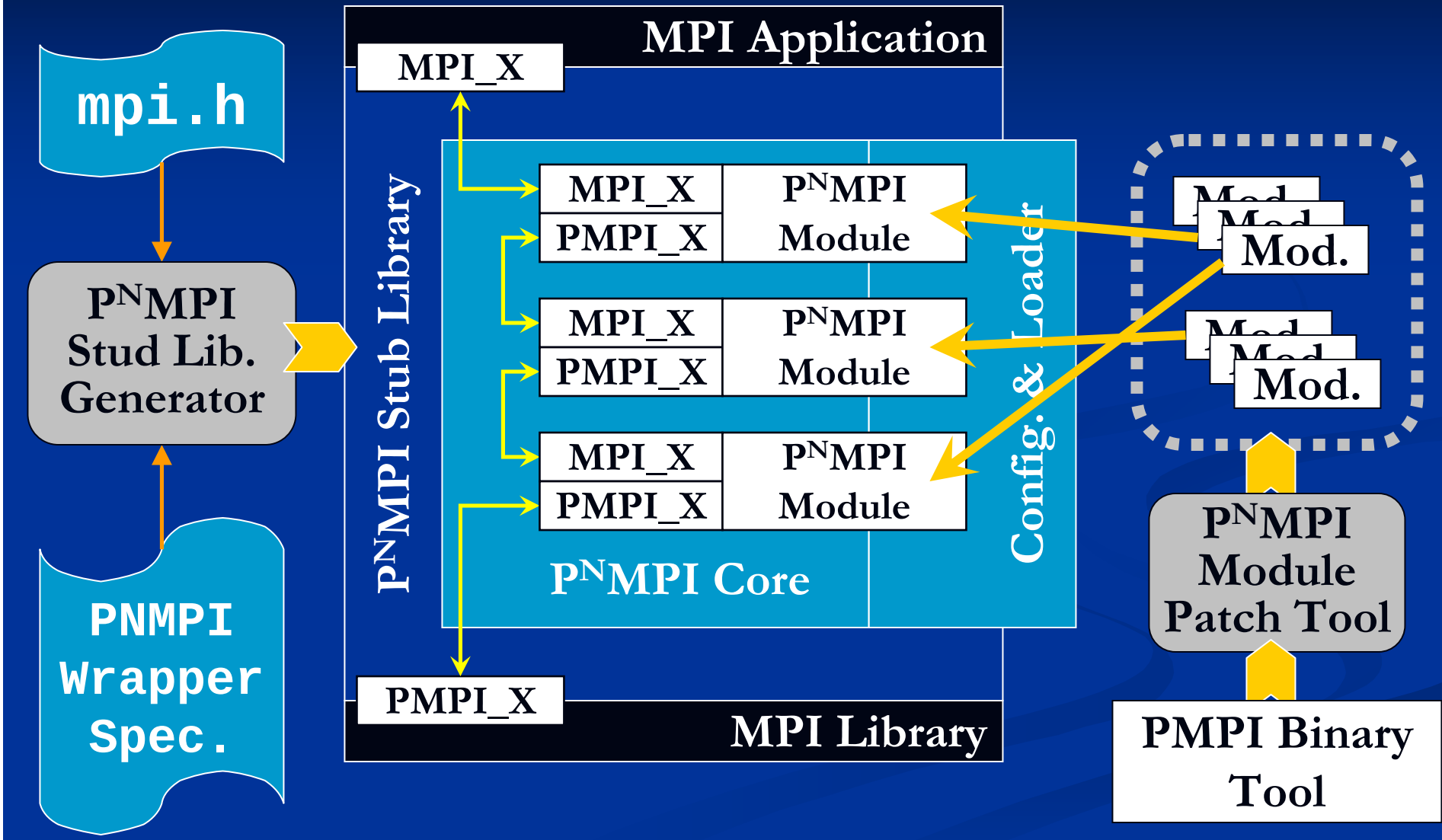


PNMPI Switch Modules

- Multiple tool stacks
  - *Defined independently*
  - *Initial tool stack called by application*
- Switch modules
  - *Dynamic stack choice*
  - *Based on arguments or dynamic steering*
- Duplication of tools
  - *Multiple contexts*
  - *Separate global state*

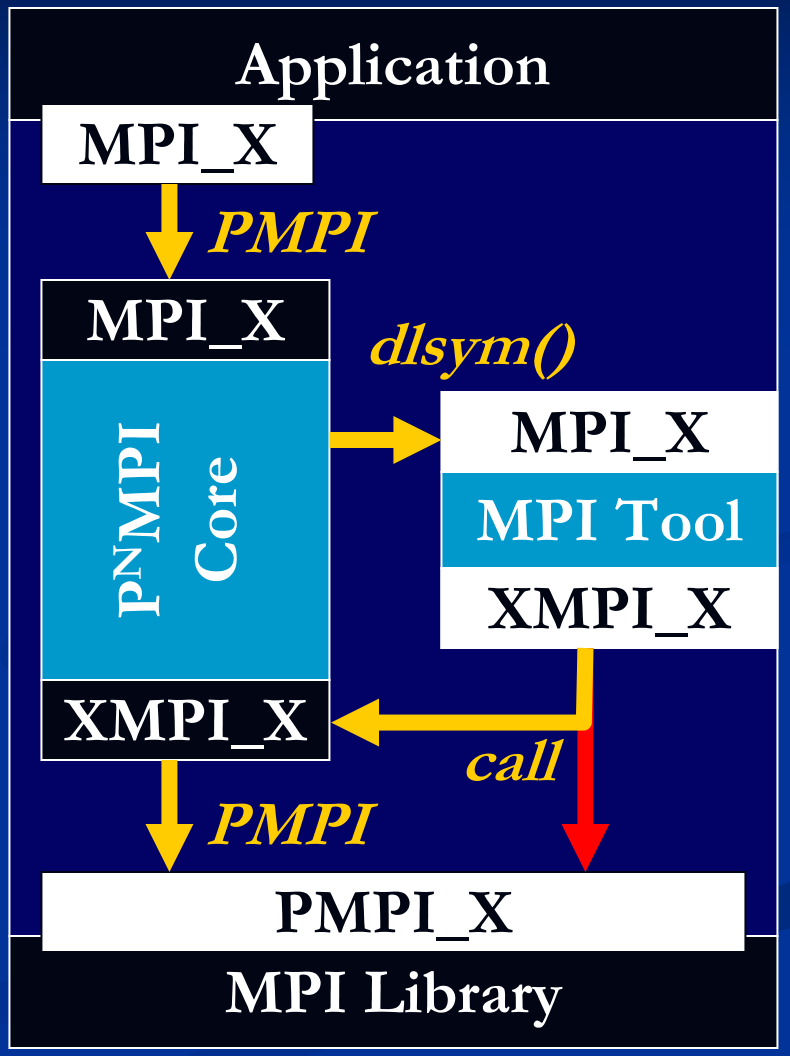


Architecture



Module Creation

- Reuse existing tools
  - *PMPI binaries*
  - *Transparency*
- Prevent PMPI calls
  - *Patch binary to rename all PMPI calls*
  - *Provide routine with patched name in P^{N}MPI*
- Core gains control after module invocation



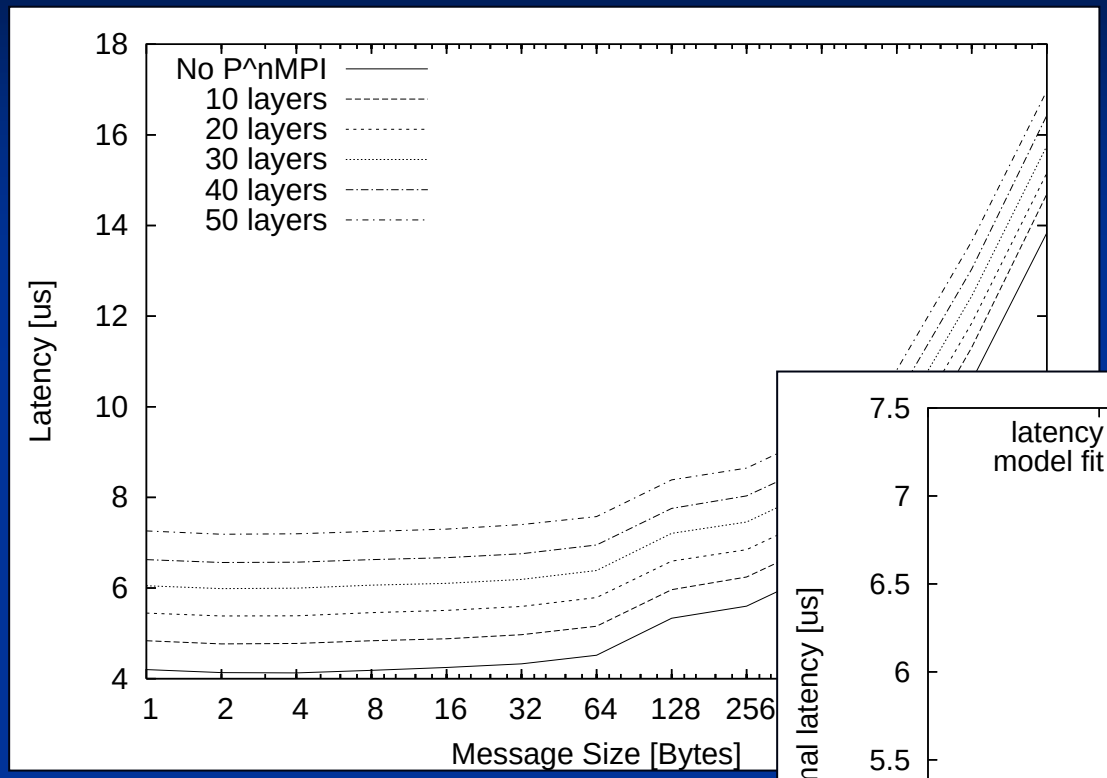
PNMPI Services

- Registration
  - *Make tool module visible to other tools*
  - *Process module arguments*
- Publish/Subscribe Services
  - *Offer callbacks to services*
  - *Query services in other modules*
  - *Type signatures*
- Pcontrol
  - *Control selected modules*
  - *PNMPI specific Pcontrol syntax*

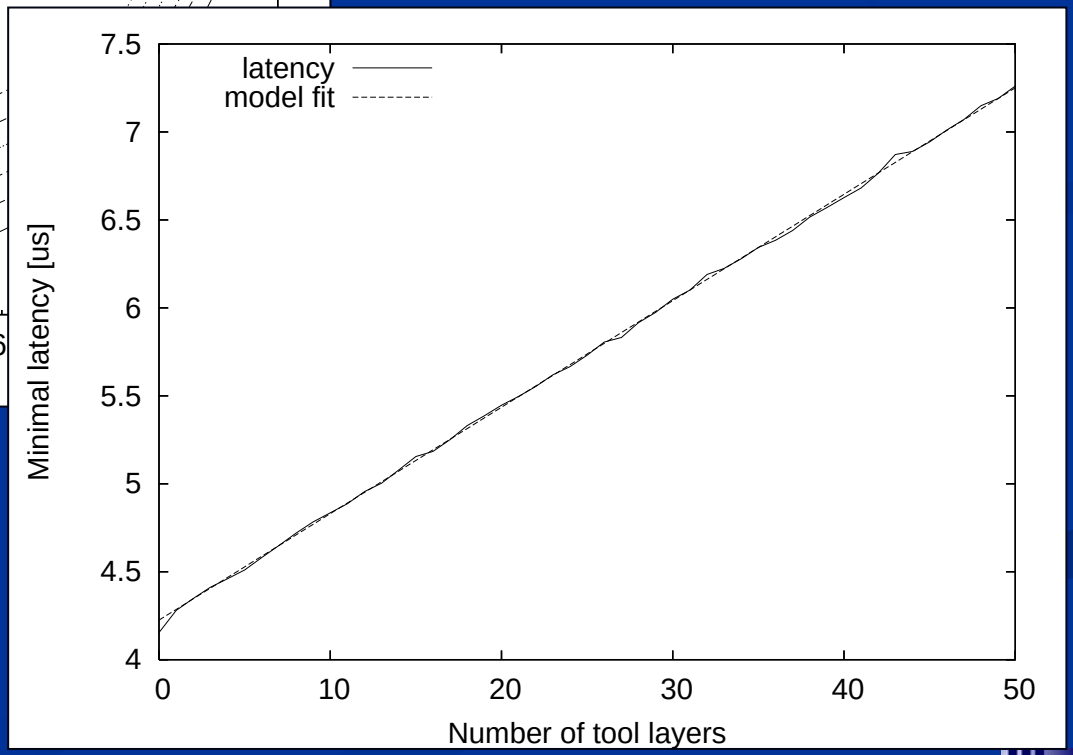
Setup & Configuration

- P^{N}MPI configuration file
  - *Define tool stacks*
  - *Set tool arguments*
  - *Evaluated at program start*
- Static version available
  - *Prelink configuration of tool stack*
  - *Support for machines like BG/L*
- Experimental Setup
  - *Atlas cluster: 44 TFlop/s cluster at LLNL*
  - *1152 nodes with 8 Opteron cores each*
  - *Mellanox Infiniband Interconnect*

Overhead



Stacking Empty Tool Layers



Overhead ~ # tools

Independent of # tasks

Usage Scenarios

- Concurrent Execution of Transparent Tools
  - *Tracing and Profiling*
  - *Message Perturbation and MPI Checker*
- Tool Cooperation
  - *Encapsulate common tool operations*
  - *Examples: datatype walking, request tracking*
  - *Application level MPI checkpointing*
- Tool Multiplexing
  - *Apply tools to subsets of applications*
  - *Run concurrent copies of the same tool*
- MPI job virtualization

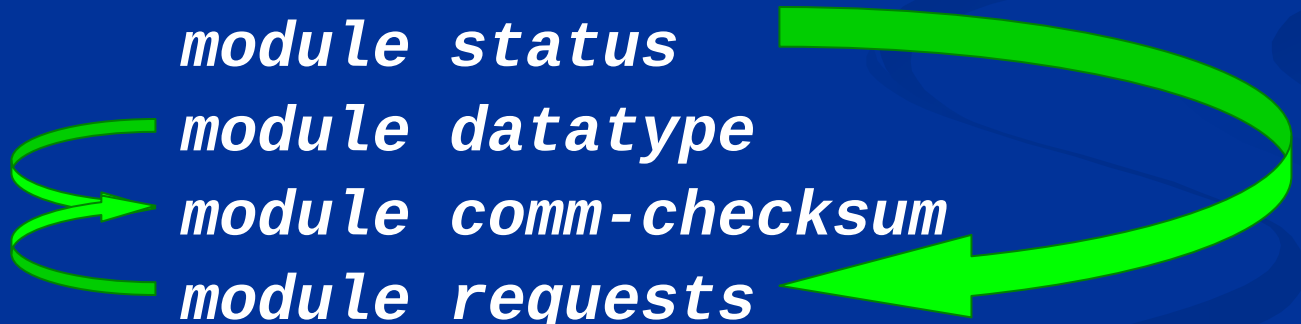
Checksums

- Goal: checksums for each message
  - *Compute at SEND*
  - *Piggyback checksum*
  - *Check at RECV*
- Detects message corruptions
  - *Message buffer corruptions*
  - *Incorrect MPI implementations*
- Requires many typical tasks
  - *Walk arbitrary datatypes*
  - *Track requests for asynchronous messages*
  - *Intercept all communication events*

Implementation

- Encapsulate each task in one module
  - *Module to capture all MPI datatypes*
  - *Replace MPI Request and Status objects*
  - *Extend generic communication callback module*
- Configuration file

```
module status
module datatype
module comm-checksum
module requests
```



- Test application: SMG 2000

Checksum Performance

Active Modules	16 tasks / 4 nodes Exec. Time	Overhead	64 tasks / 16 nodes Exec. Time	Overhead	256 tasks / 64 nodes Exec. Time	Overhead
No P^NMPI	29.18	—	31.35	—	34.84	—
Status	29.28	0.3%	31.45	0.3%	34.98	0.4%
Requests	29.37	0.6%	31.59	0.8%	35.18	1.0%
Datatype	29.25	0.2%	31.42	0.2%	34.98	0.4%
Comm	29.47	1.0%	31.75	1.3%	35.30	1.3%
Piggyback	29.85	2.3%	32.48	3.6%	36.77	5.5%
Checksum	34.39	17.8%	37.11	18.4%	42.67	22.5%

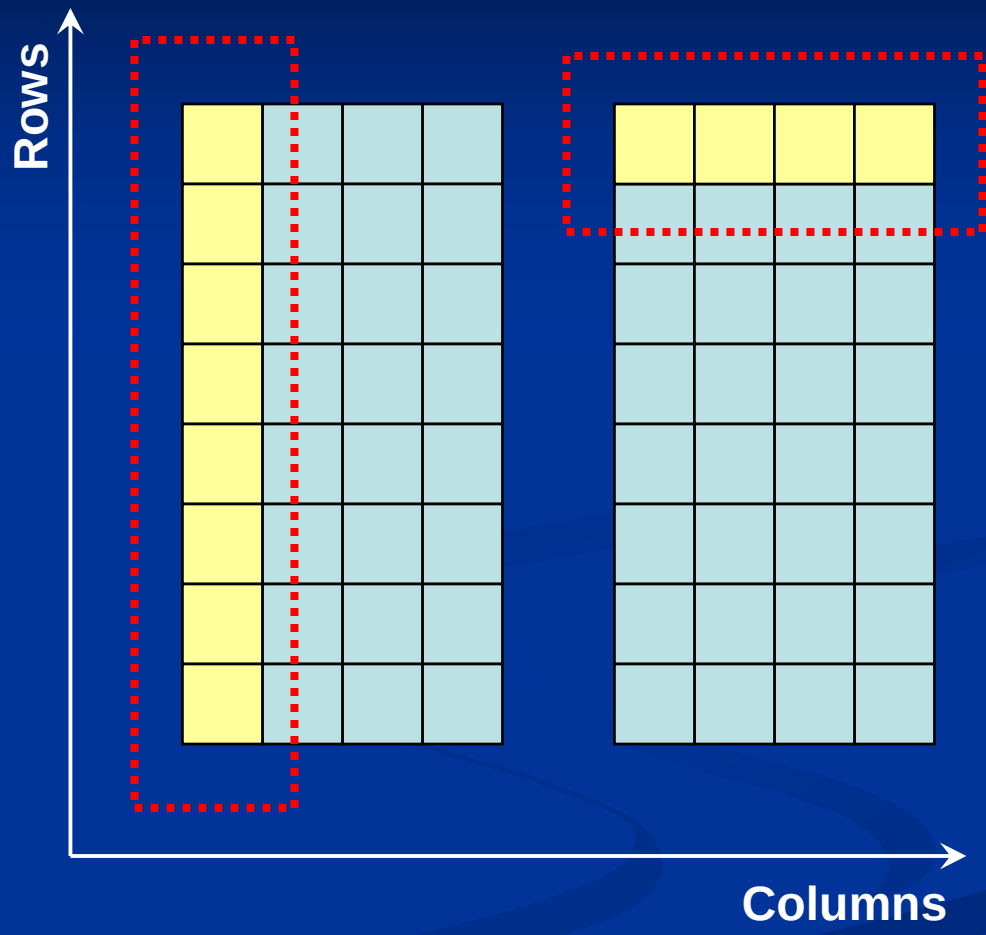
- Observations:
  - *Support modules cause only minimal overhead*
  - *Scalability*
  - *Actual overhead comes from tool itself*

Selective Profiling

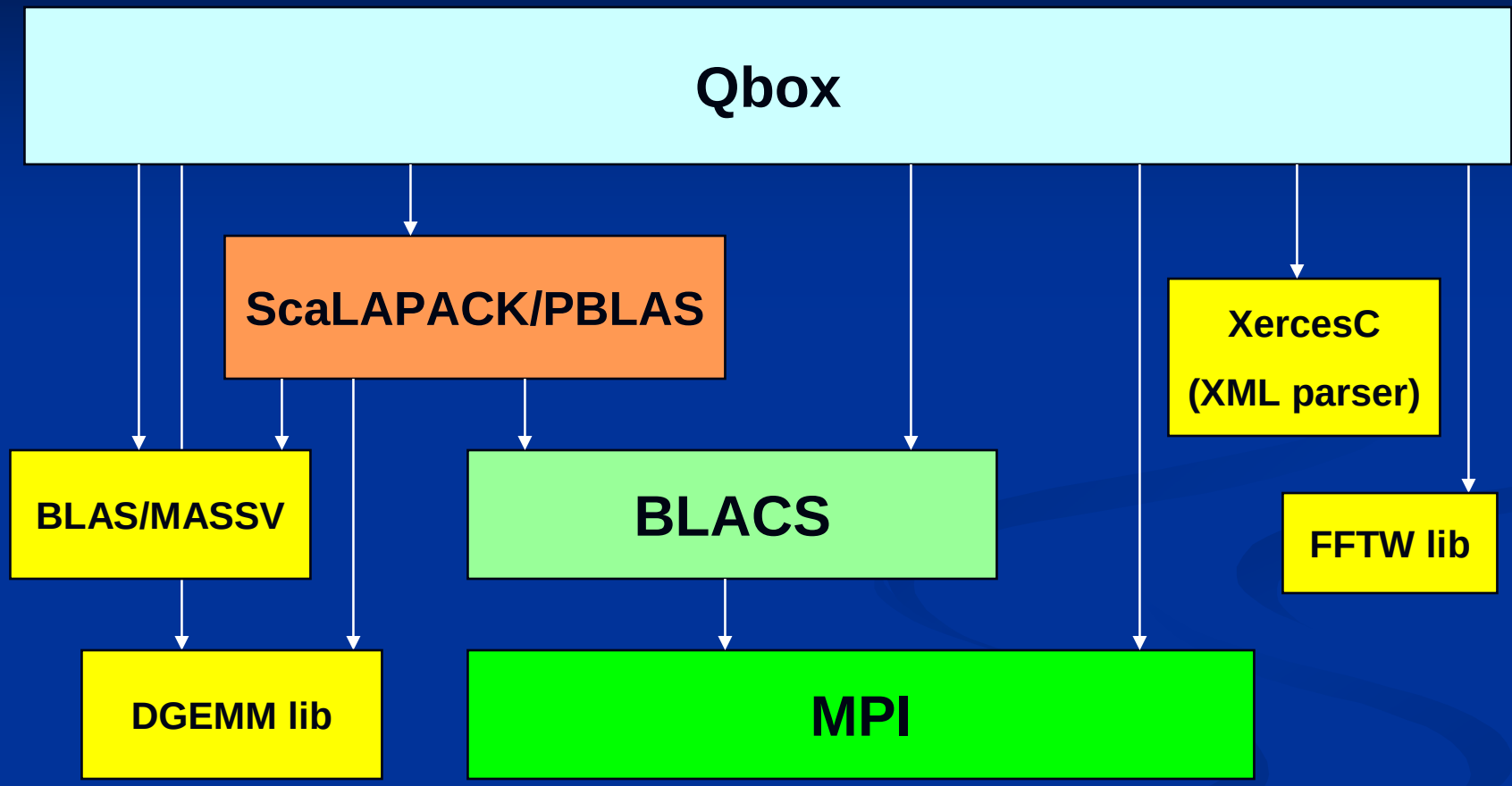
- MPI Profiling
  - *Provide aggregated view of MPI performance*
  - *Example: mpiP library*
- Disadvantage
  - *Global view of whole application*
  - *Can't distinguish communicators or groups*
- Approach with P^{N}MPI
  - *Replicate instances of unmodified profiler*
  - *Switch module to determine context*
  - *Forward MPI call to matching profiler instance*

Example: QBox

- Dense matrix
  - *Row and column communicators*
  - *Global operations*
- Communication patterns
  - *Depends on the communicator*
  - *Need to profile separately*



QBox Code Structure



- *Implicit generation of communicators*
- *Frequent creation and destruction*

Profiling Setup

- Configuration file:

Default Stack

```
module commsize-switch
argument sizes 8 4
argument stacks column row
module mpiP
```

Target Stack 1

```
stack row
module mpiP1
```

Target Stack 1

```
stack column
module mpiP2
```

Switch Module

Arguments controlling switch module

Multiple profiling instances

Results

Count	Global	Sum	COMM_WOLRD	Row	Column
Send	317365	317245	31014	202972	83259
Allreduce	319028	319028	269876	49152	0
All2allv	471488	471488	471488	0	0
Recv	379355	379265	93034	202972	83259
Bcast	401312	401042	11168	331698	58176

- Information helpful for …
  - *Evaluating interactions of libraries and MPI*
  - *Understanding impact on the network*
  - *Optimization of collectives*
  - *Node mapping*

Summary PnMPI

- PNMPI tool infrastructure
  - *Keep PMPI interface*
  - *Specify any number of tools at runtime*
  - *Dynamic creation of tool chains*
- Optional services
  - *Publish/Subscribe services*
  - *Dynamic tool stack selection*
- Extend/Specialize/Assemble tools
  - *New functionality using existing building blocks*
  - *Change tools without relinking*
  - *Fast prototyping using generic tool services*

Lessons for Petascale Tools

- Tools are essential in Petascale efforts
  - *Need to debug at large scale*
  - *Performance optimization to exploit machines*
- Centralized infrastructures will not work
  - *Tree-based aggregation schemes*
  - *Distributed storage and analysis*
  - *Node count independent infrastructures*
- Need for flexible and scalable toolboxes
  - *Integration and interoperability*
  - *Comprehensive infrastructures*
  - *Community effort necessary*

Future Work

- Scalable performance tools
  - *Utilization of tree-based communication (MRNet)*
  - *Platform: Open|SpeedShop*
- Tool integration & infrastructures
  - *Leverage or establish community standards*
  - *Encapsulate common tasks (e.g., MPI launcher)*
- New capabilities
  - *Automatic MPI pattern extraction*
  - *Memory scalability analysis*
  - *"Performance Cook Books"*

Conclusions

- System size growing towards Petascale
  - *Tools must scale with systems and codes*
  - *New concepts and infrastructures necessary*
- Scalable debugging with STAT
  - *Lightweight tool to narrow search space*
  - *Tree-based stack trace aggregation*
- Dynamic MPI tool creation with P^NMPI
  - *Ability to quickly create application specific tools*
  - *Transparently reuse and extend existing tools*
- Tool interoperability increasingly important