On Utilizing Experiment Data Repository for Performance Analysis of Parallel Applications

Hong-Linh Truong[1], Thomas Fahringer[2]

[1] Institute for Software Science,
University of Vienna, Austria
truong@par.univie.ac.at

[2] Institute of Computer Science,
University of Innsbruck, Austria
Thomas.Fahringer@uibk.ac.at

www.par.univie.ac.at/project/scalea

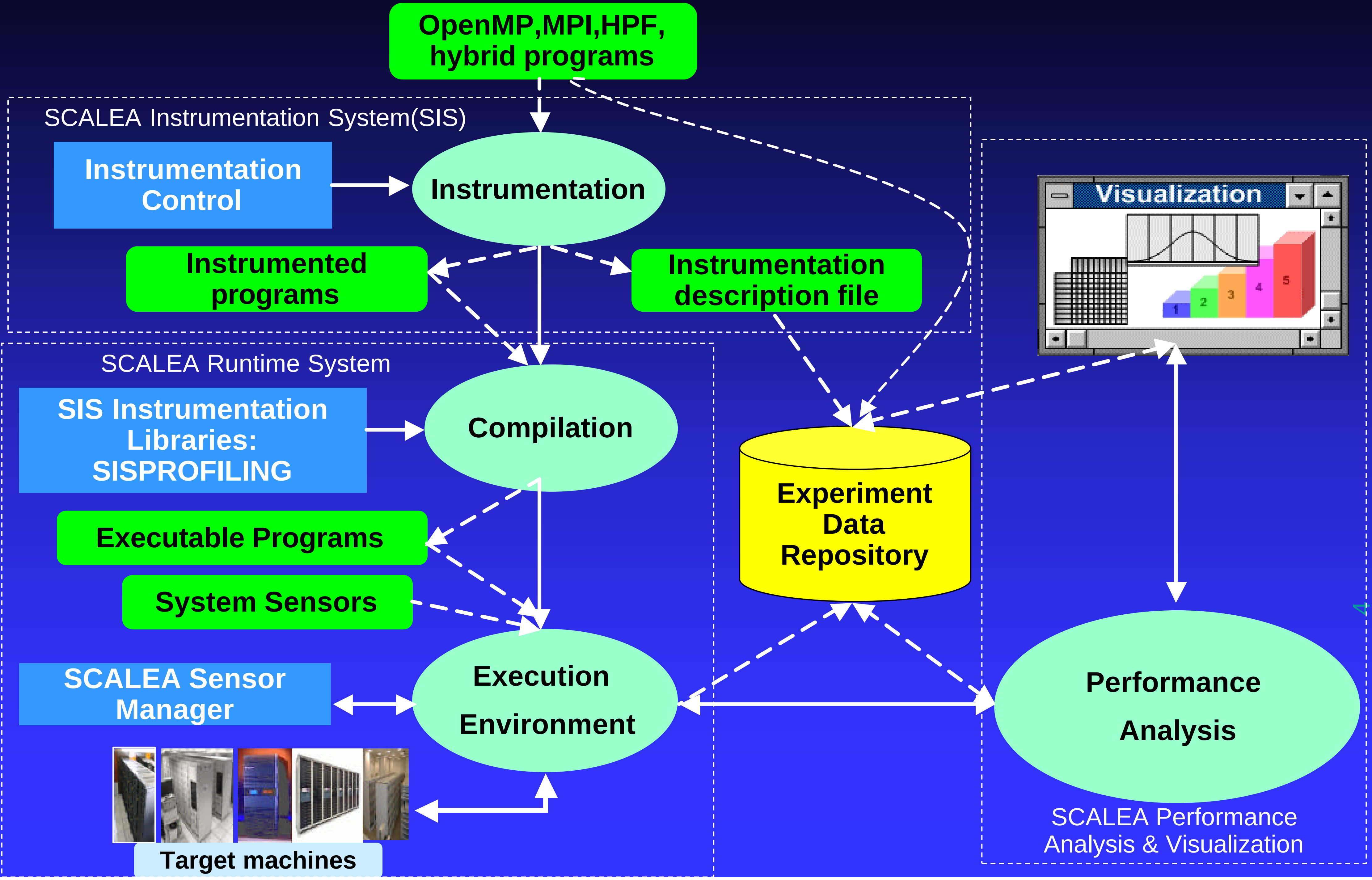
Outline

- Motivation
- SCALEA
- Experiment Data Repository
  - Experiment-related Data
  - Implementation Overview
  - Experiment-related APIs
- Achievements
  - Search and Filter Performance Data
  - Multi-Experiment Analysis
  - Tools Integration
- Conclusions and Future Work

Motivation

- The need to collect and archive performance data for
  - Multi-experiment analysis
  - Performance comparison
- Limitation on supporting code complexity:
  - Handle large-scale performance data
  - Basic search on performance data
- Lack of capabilities to export, share, and exchange performance data and tools interoperation

SCALEA (SC01, Euro-Par02, PVMMPI02)



OpenMP,MPI,HPF, hybrid programs

SCALEA Instrumentation System(SIS)

Instrumentation Control

Instrumentation

Instrumented programs

Instrumentation description file

SCALEA Runtime System

SIS Instrumentation Libraries: SISPROFILING

Compilation

Executable Programs

System Sensors

Experiment Data Repository

SCALEA Sensor Manager

Execution Environment

Target machines

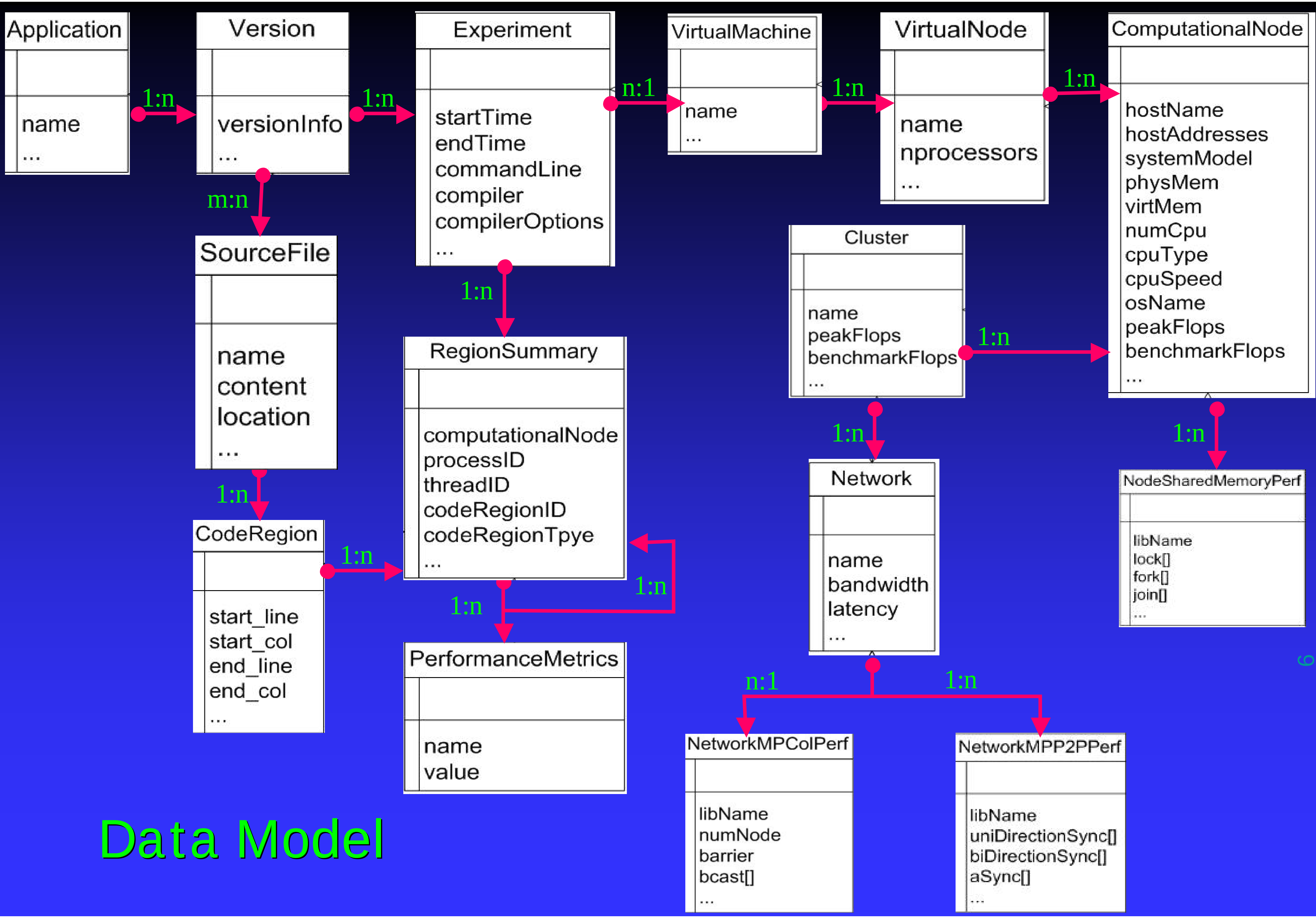
Visualization

Performance Analysis

SCALEA Performance Analysis & Visualization

Experiment Data Repository

- For comparing performance across experiments with large amounts of data
  - Need to archive performance data
  - Require a very strong and flexible database system
- Data repository is first step required by mining, knowledge discovery in performance data
- Experiment Data Repository:
  - Stores information about application, source code, machine information, and performance data
  - Associates performance results with source code and machine information
  - Supports sophisticated analysis and easy integration with other tools



Application
name
...

Version
versionInfo
...

Experiment
startTime
endTime
commandLine
compiler
compilerOptions
...

VirtualMachine
name
...

VirtualNode
name
nprocessors
...

ComputationalNode
hostName
hostAddresses
systemModel
physMem
virtMem
numCpu
cpuType
cpuSpeed
osName
peakFlops
benchmarkFlops
...

SourceFile
name
content
location
...

RegionSummary
computationalNode
processID
threadID
codeRegionID
codeRegionTpye
...

Cluster
name
peakFlops
benchmarkFlops
...

CodeRegion
start_line
start_col
end_line
end_col
...

PerformanceMetrics
name
value

Network
name
bandwidth
latency
...

NodeSharedMemoryPerf
libName
lock[]
fork[]
join[]
...

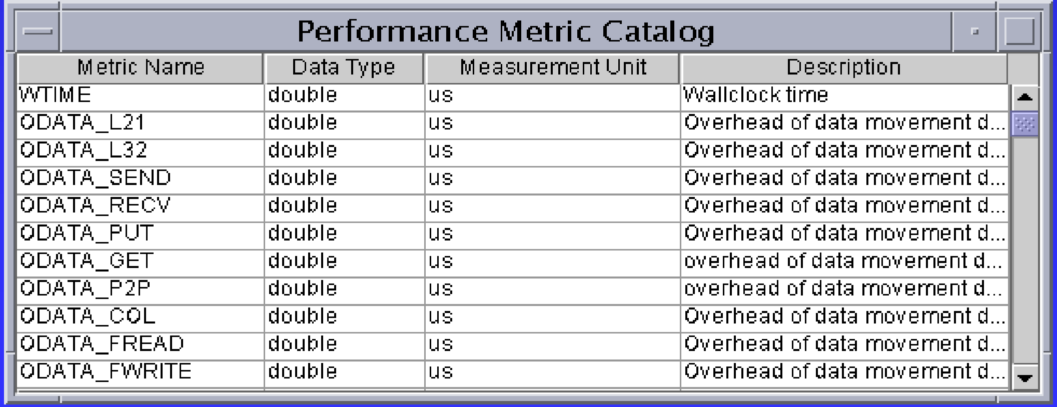
NetworkMPColPerf
libName
numNode
barrier
bcast[]
...

NetworkMPP2PPerf
libName
uniDirectionSync[]
biDirectionSync[]
aSync[]
...

Data Model

Performance Metric Catalog

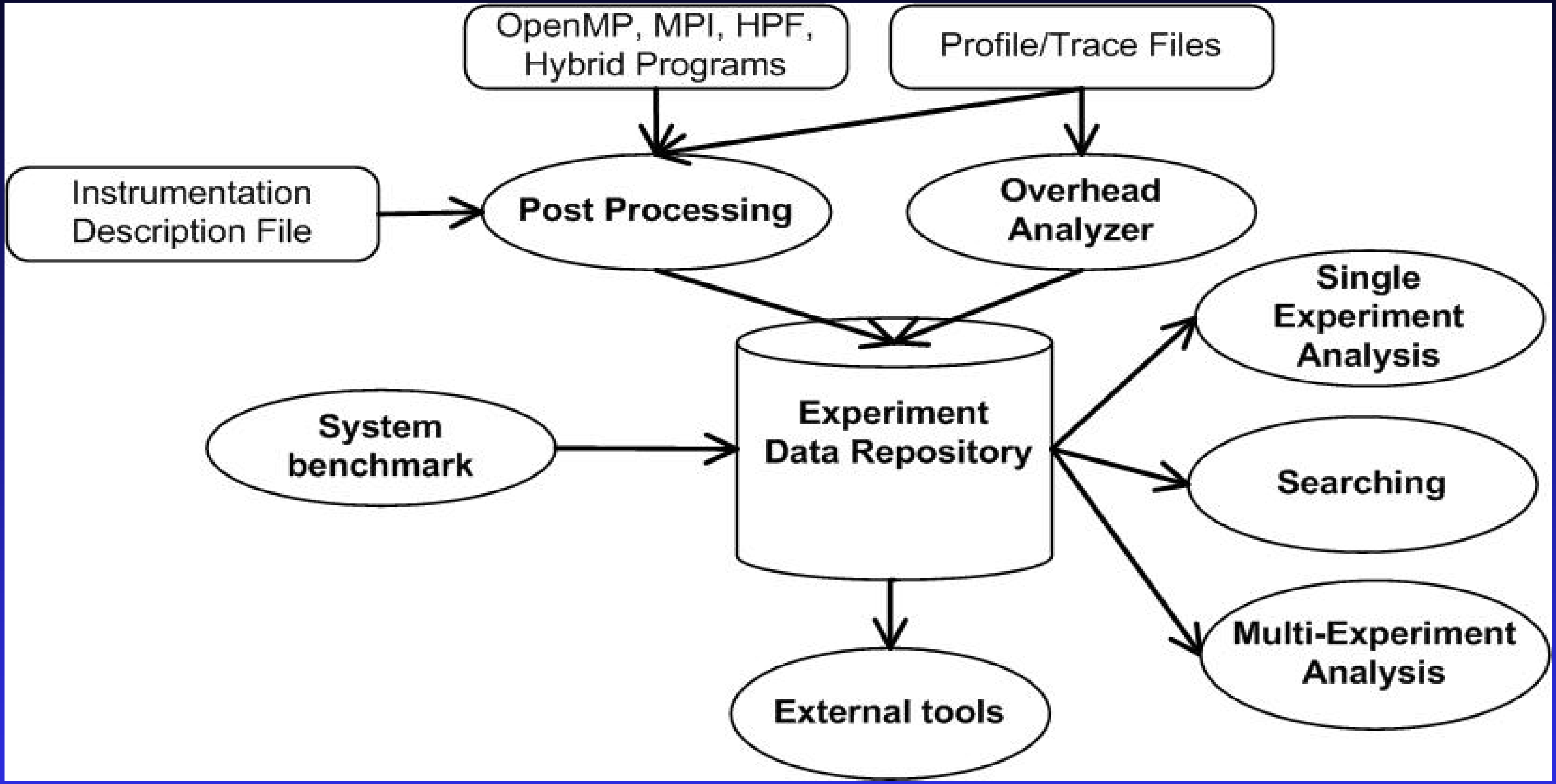
- Why performance metric catalog:
  - Documentation about metrics provided
  - Help tools/users understanding and interpreting performance data
- Performance metric has 4 properties
  - unique name is a value that distinguishes this metric from all others
  - data type describes data type of measurement which describes the value of the metric.
  - measurement unit is the unit of measurement
  - semantics (or well-defined meaning)
- All metrics are stored in a catalog in experiment data repository



Performance Metric Catalog

Metric Name	Data Type	Measurement Unit	Description
WTIME	double	us	Wallclock time
ODATA_L21	double	us	Overhead of data movement d...
ODATA_L32	double	us	Overhead of data movement d...
ODATA_SEND	double	us	Overhead of data movement d...
ODATA_RECV	double	us	Overhead of data movement d...
ODATA_PUT	double	us	Overhead of data movement d...
ODATA_GET	double	us	overhead of data movement d...
ODATA_P2P	double	us	overhead of data movement d...
ODATA_COL	double	us	Overhead of data movement d...
ODATA_FREAD	double	us	Overhead of data movement d...
ODATA_FWRITE	double	us	Overhead of data movement d...

Implementation Overview



OpenMP, MPI, HPF, Hybrid Programs

Profile/Trace Files

Instrumentation Description File

Post Processing

Overhead Analyzer

Single Experiment Analysis

System benchmark

Experiment Data Repository

Searching

Multi-Experiment Analysis

External tools

- Powering search and archive with great flexibility and robustness
  - Relational database based on PostgreSQL
- Taking advantages of portability and network capability
  - All components written in Java
  - Database connection powered by JDBC

Experiment-related Data APIs

- Well-defined APIs for accessing data in Experiment Data Repository

- Java implementation

```
public class PerformanceMetric {
  private String metricName;
  private Object metricValue;
  public String getMetricName(){…};
  public void   setMetricName(String metricName){…};
  public Object getMetricValue(){…};
  public void   setMetricValue(Object metricValue){…};
  ...
  }
```

```
public class ProcessingUnit {
  private String computationalNode;
  private int    processID ;
  private int    threadID ;
  …
  public ProcessingUnit(String node,int process, int thread) {...}
  …
  }
```

Experiment-related Data APIs (const.)

```
public class RegionSummary  {
 ...
 public PerformanceMetric[] getMetrics(){...}
 public PerformanceMetric getMetric(String metricName){...}
 ...
 }
```

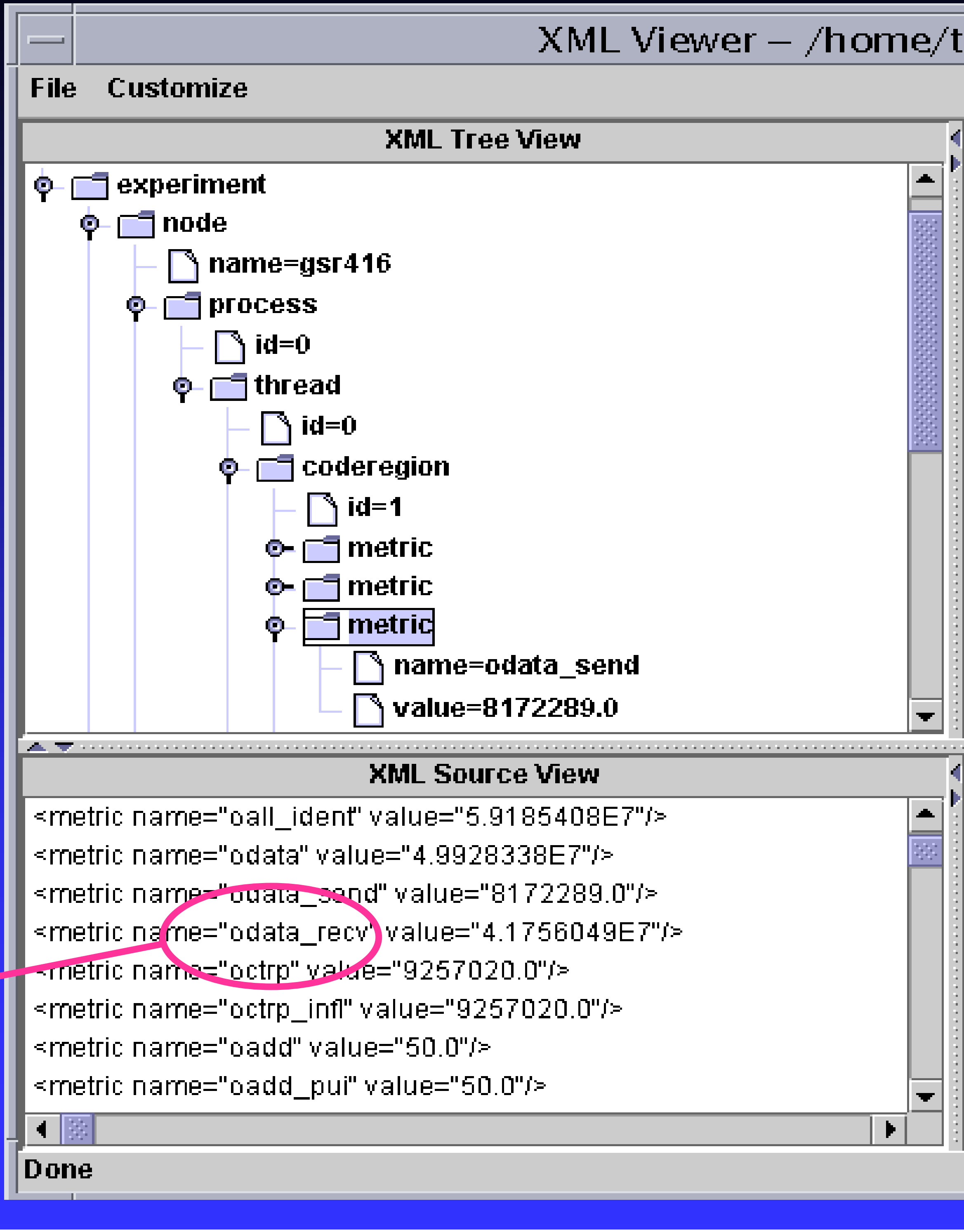
```
public class ExperimentData {
 DatabaseConnection   connection;
 ...
 public ProcessingUnit[]  getProcessingUnits(Experiment e){...}
 public RegionSummary[]   getRegionSummaries(CodeRegion cr, Experiment
 e){...}
 public RegionSummary   getRegionSummary(CodeRegion cr, ProcessingUnit
 pu, Experiment e) {...}
 public RegionSummary   getRegionSummary(CodeRegion cr, Experiment
 e,ProcessingUnit pu, RegionSummary parent) {...}   public
 RegionSummary[]   getChildOfRegionSummary(RegionSummary rs){...}
 public RegionSummary    getParentOfRegionSummary(RegionSummary
 rs){...}
 ...
}
```

Example of Using APIs

```
CodeRegion cr = new CodeRegion(“IONIZE_MOVE”);
Experiment e = new Experiment(“9Nx4P,P4”);
ProcessingUnit
   pu =new ProcessingUnit(``gsr402.vcpc.ac.at'',2,0);

ExperimentData
   ed = new ExperimentData(new DatabaseConnection(...));
RegionSummary  rs = ed.getRegionSummary (cr,pu,e);


PerformanceMetric
         overhead =rs.getMetric(''oall_ident'');
PerformanceMetric
         wtime   =rs.getMetric(``wtime'');

double overheadRatio=
   ((Double)overhead.getMetricValue()).doubleValue()/
       ((Double)wtime.getMetricValue()).doubleValue();
```

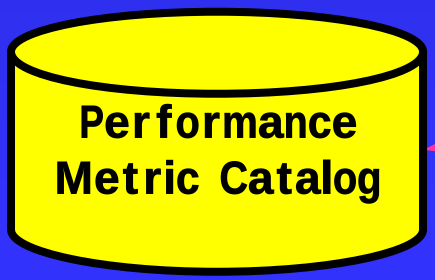
XML Data

- ➢Facilitate information exchange between tools
- ➢Allow tools interoperation in a uniform way
- ➢With/without source code
- ➢Call-graph, selective metrics



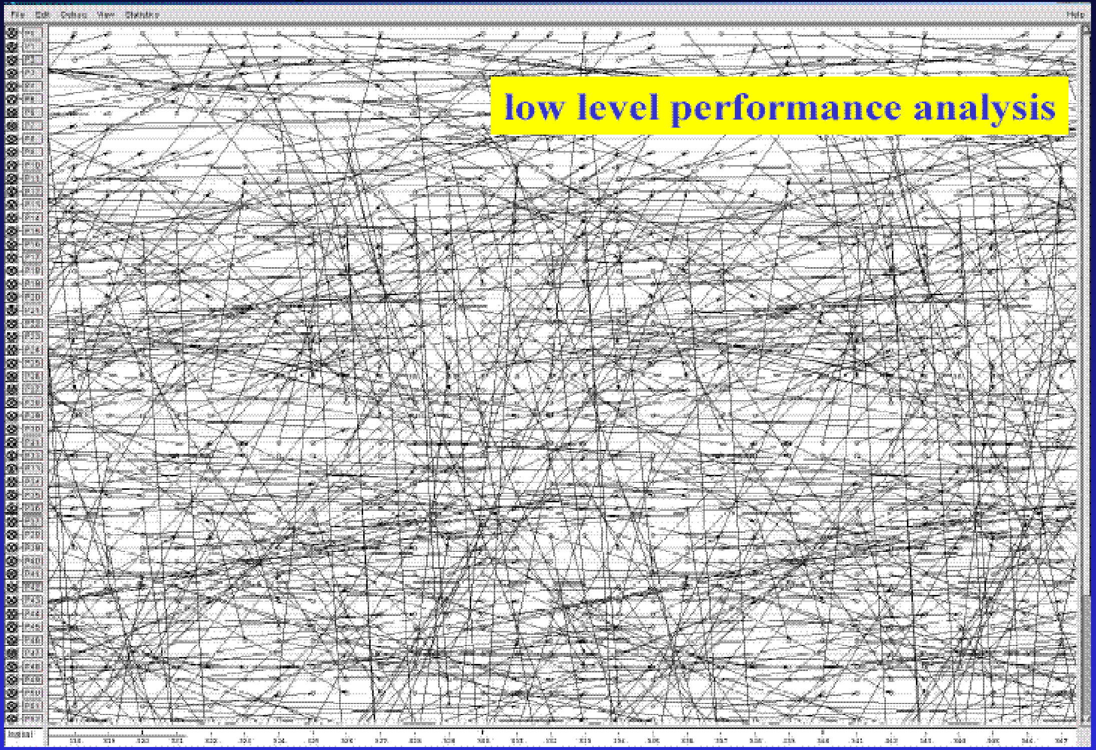
```
experiment
  node
    name=gsr416
    process
      id=0
      thread
        id=0
        coderegion
          id=1
          metric
          metric
          metric
            name=odata_send
            value=8172289.0
```

```
<metric name="oall_ident" value="5.9185408E7"/>
<metric name="odata" value="4.9928338E7"/>
<metric name="odata_send" value="8172289.0"/>
<metric name="odata_recv" value="4.1756049E7"/>
<metric name="octrp" value="9257020.0"/>
<metric name="octrp_infl" value="9257020.0"/>
<metric name="oadd" value="50.0"/>
<metric name="oadd_pui" value="50.0"/>
```



Search and Filter Capabilities

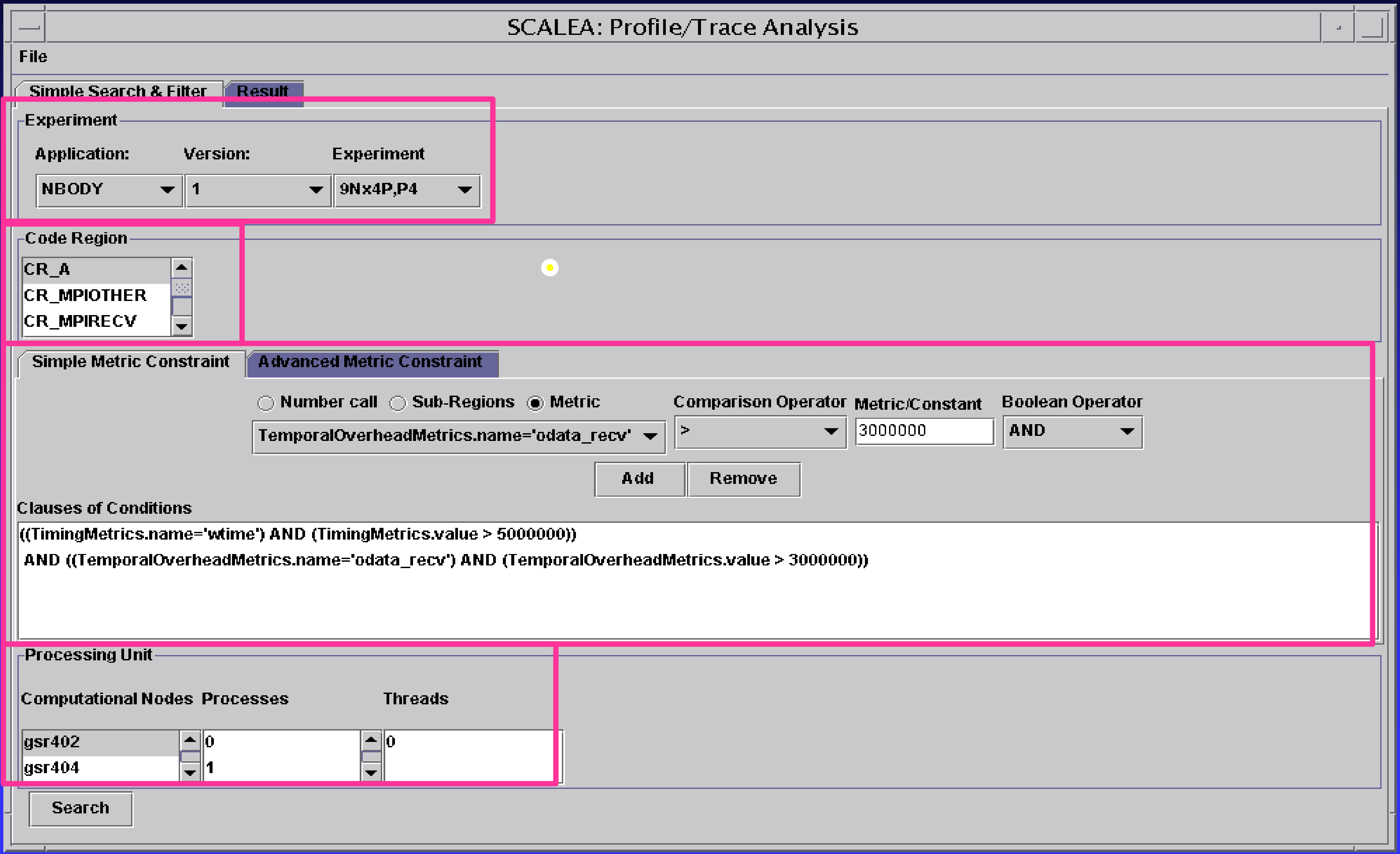
- Existing problems:
  - difficult to find interesting events via process time-lines, zooming, scrolling, call-graph
  - search and filter data based on files are not efficient, robust and fast



Picture taken from D. Kranzlmueller ´s talk

- Search and filter data based relational database and SQL: flexibility and robustness
- We support generic search and filter

Search and Filter Capabilities



SCALEA: Profile/Trace Analysis

Experiment

Application: NBODY

Version: 1

Experiment: 9Nx4P,P4

Code Region

○ Number call ○ Sub-Regions ● Metric

TemporalOverheadMetrics.name='odata_recv'

Comparison Operator: >

Metric/Constant: 3000000

Boolean Operator: AND

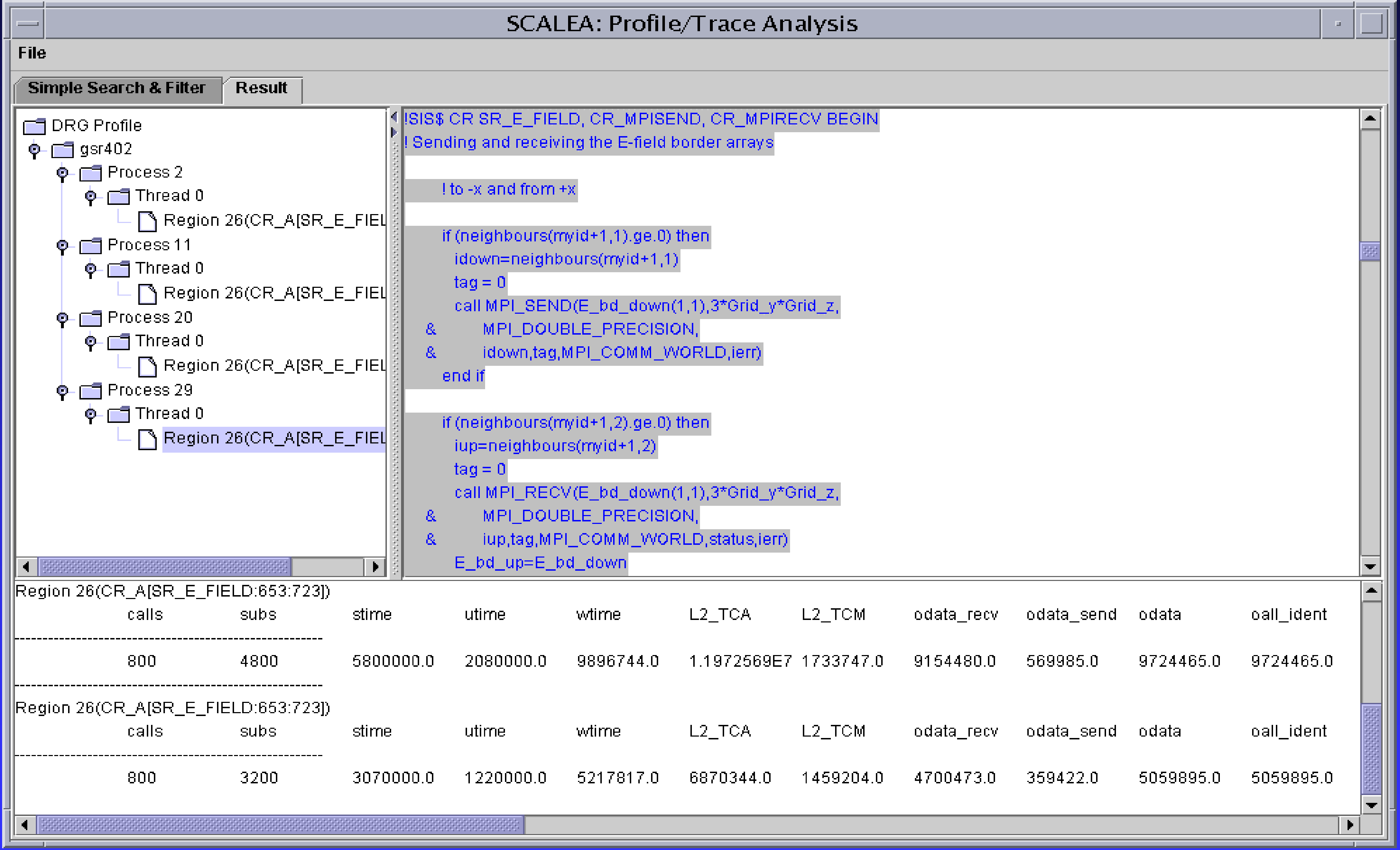
Clauses of Conditions

```
((TimingMetrics.name='wtime') AND (TimingMetrics.value > 5000000))
 AND ((TemporalOverheadMetrics.name='odata_recv') AND (TemporalOverheadMetrics.value > 3000000))
```

Processing Unit

Computational Nodes	Processes	Threads
gsr402	0	0
gsr404	1	

Search and Filter Capabilities (const.)

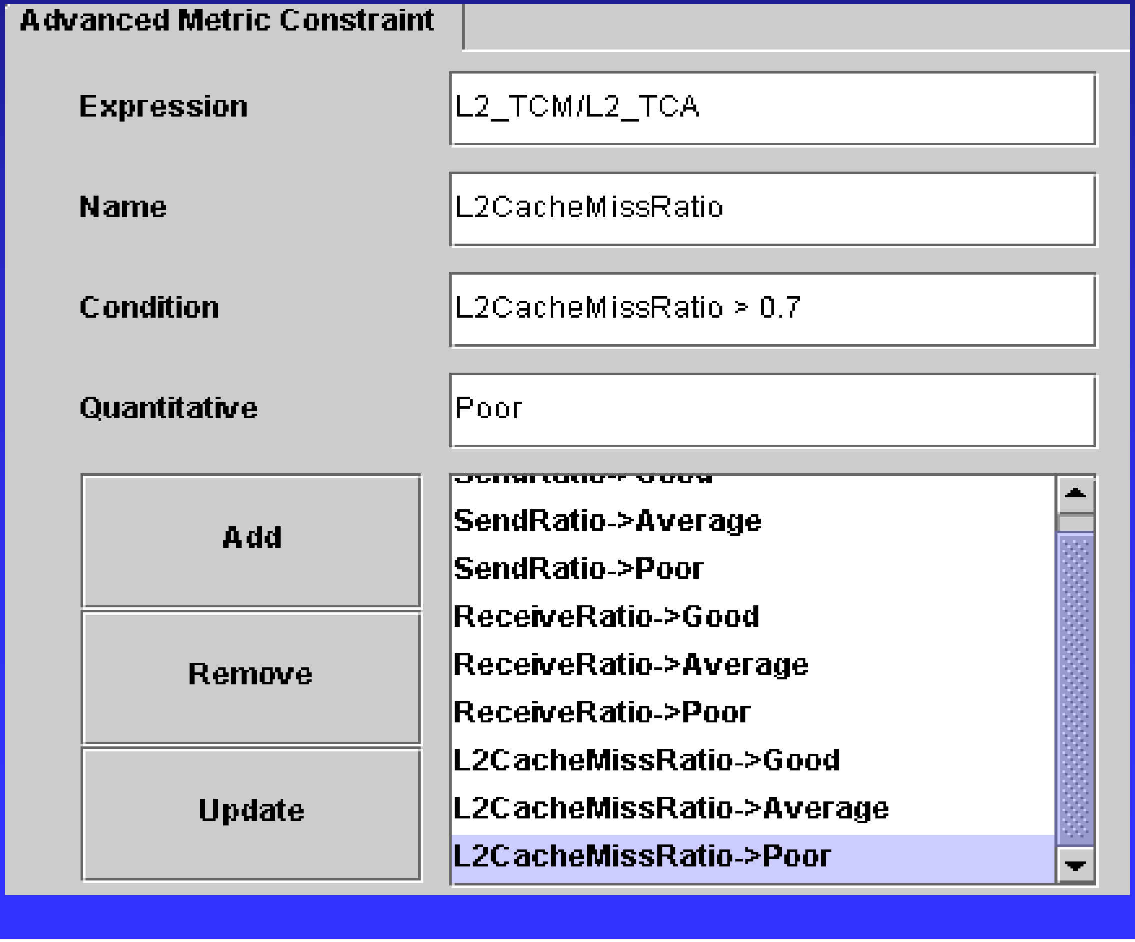


```
!SIS$ CR SR_E_FIELD, CR_MPISEND, CR_MPIRECV BEGIN
! Sending and receiving the E-field border arrays

      ! to -x and from +x

      if (neighbours(myid+1,1).ge.0) then
        idown=neighbours(myid+1,1)
        tag = 0
        call MPI_SEND(E_bd_down(1,1),3*Grid_y*Grid_z,
    &        MPI_DOUBLE_PRECISION,
    &        idown,tag,MPI_COMM_WORLD,ierr)
      end if

      if (neighbours(myid+1,2).ge.0) then
        iup=neighbours(myid+1,2)
        tag = 0
        call MPI_RECV(E_bd_down(1,1),3*Grid_y*Grid_z,
    &        MPI_DOUBLE_PRECISION,
    &        iup,tag,MPI_COMM_WORLD,status,ierr)
        E_bd_up=E_bd_down
```

```
Region 26(CR_A[SR_E_FIELD:653:723])
            calls       subs        stime        utime        wtime        L2_TCA       L2_TCM       odata_recv   odata_send   odata        oall_ident
---------------------------------------------------
            800         4800        5800000.0    2080000.0    9896744.0    1.1972569E7  1733747.0    9154480.0    569985.0     9724465.0    9724465.0
---------------------------------------------------
Region 26(CR_A[SR_E_FIELD:653:723])
            calls       subs        stime        utime        wtime        L2_TCA       L2_TCM       odata_recv   odata_send   odata        oall_ident
---------------------------------------------------
            800         3200        3070000.0    1220000.0    5217817.0    6870344.0    1459204.0    4700473.0    359422.0     5059895.0    5059895.0
```

Search and Filter Capabilities (const.)

- Defining an expression of performance metrics
- Discretizing the expression into quantitative characteristics
- Search based on selected quantitative characteristics



Advanced Metric Constraint

Expression: L2_TCM/L2_TCA

Name: L2CacheMissRatio

Condition: L2CacheMissRatio > 0.7

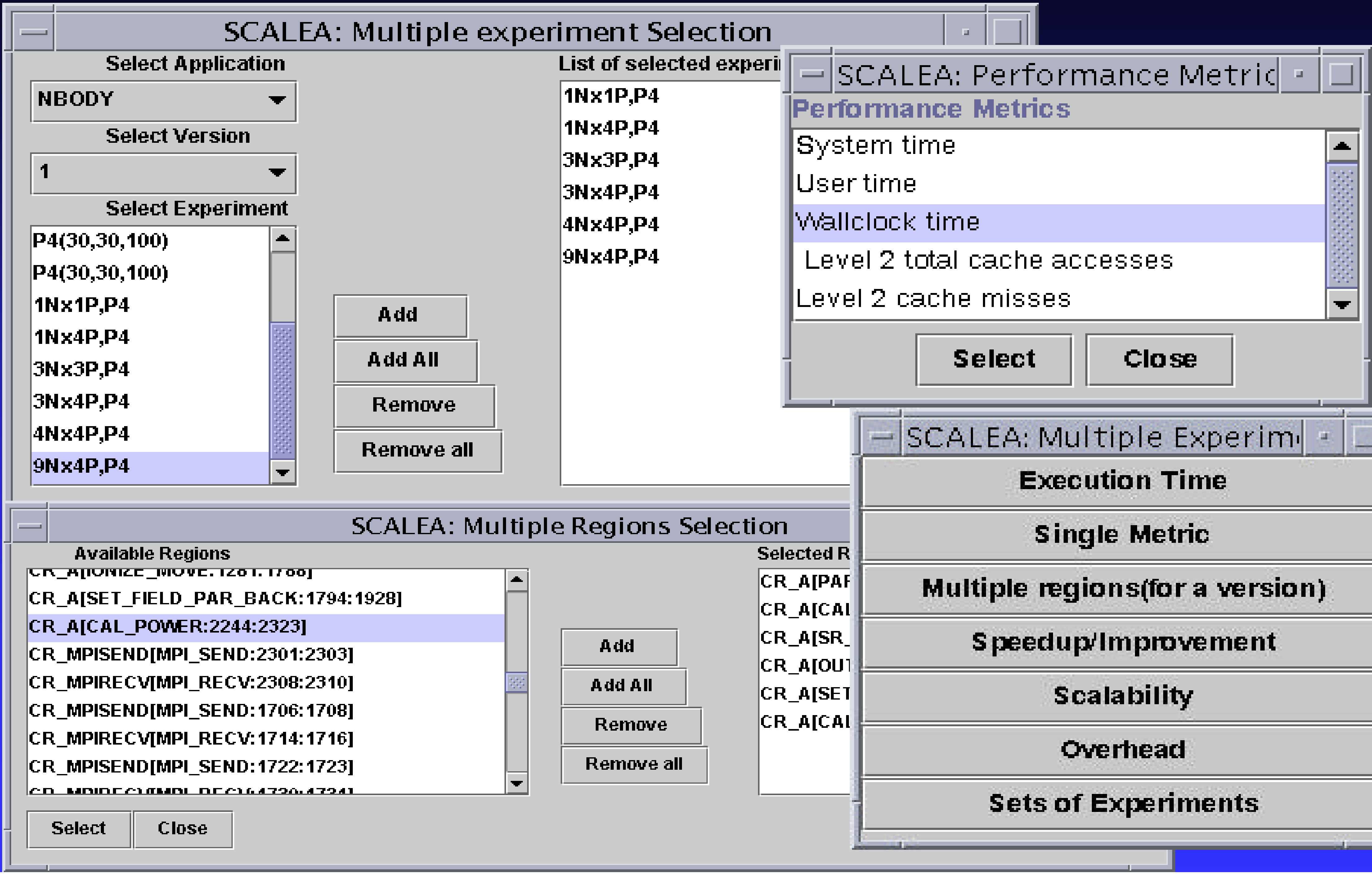
Quantitative: Poor

SendRatio->Good
SendRatio->Average
SendRatio->Poor
ReceiveRatio->Good
ReceiveRatio->Average
ReceiveRatio->Poor
L2CacheMissRatio->Good
L2CacheMissRatio->Average
L2CacheMissRatio->Poor

Multi-Experiment Analysis

- Many ways to specify what you want to analysis
  - Experiments
  - Code regions
  - Performance metrics
  - Statistic methods
- Various analyses
  - Performance comparison for different sets of experiments
  - Overhead analysis for multi-experiment
  - Performance speedup/improvement at both program and code region level

Select what to be analyzed



SCALEA: Multiple experiment Selection

Select Application

NBODY

Select Version

1

Select Experiment

P4(30,30,100)
P4(30,30,100)
1Nx1P,P4
1Nx4P,P4
3Nx3P,P4
3Nx4P,P4
4Nx4P,P4
9Nx4P,P4

List of selected experi

1Nx1P,P4
1Nx4P,P4
3Nx3P,P4
3Nx4P,P4
4Nx4P,P4
9Nx4P,P4

SCALEA: Performance Metric

Performance Metrics

System time
User time
Wallclock time
Level 2 total cache accesses
Level 2 cache misses

SCALEA: Multiple Experim

SCALEA: Multiple Regions Selection

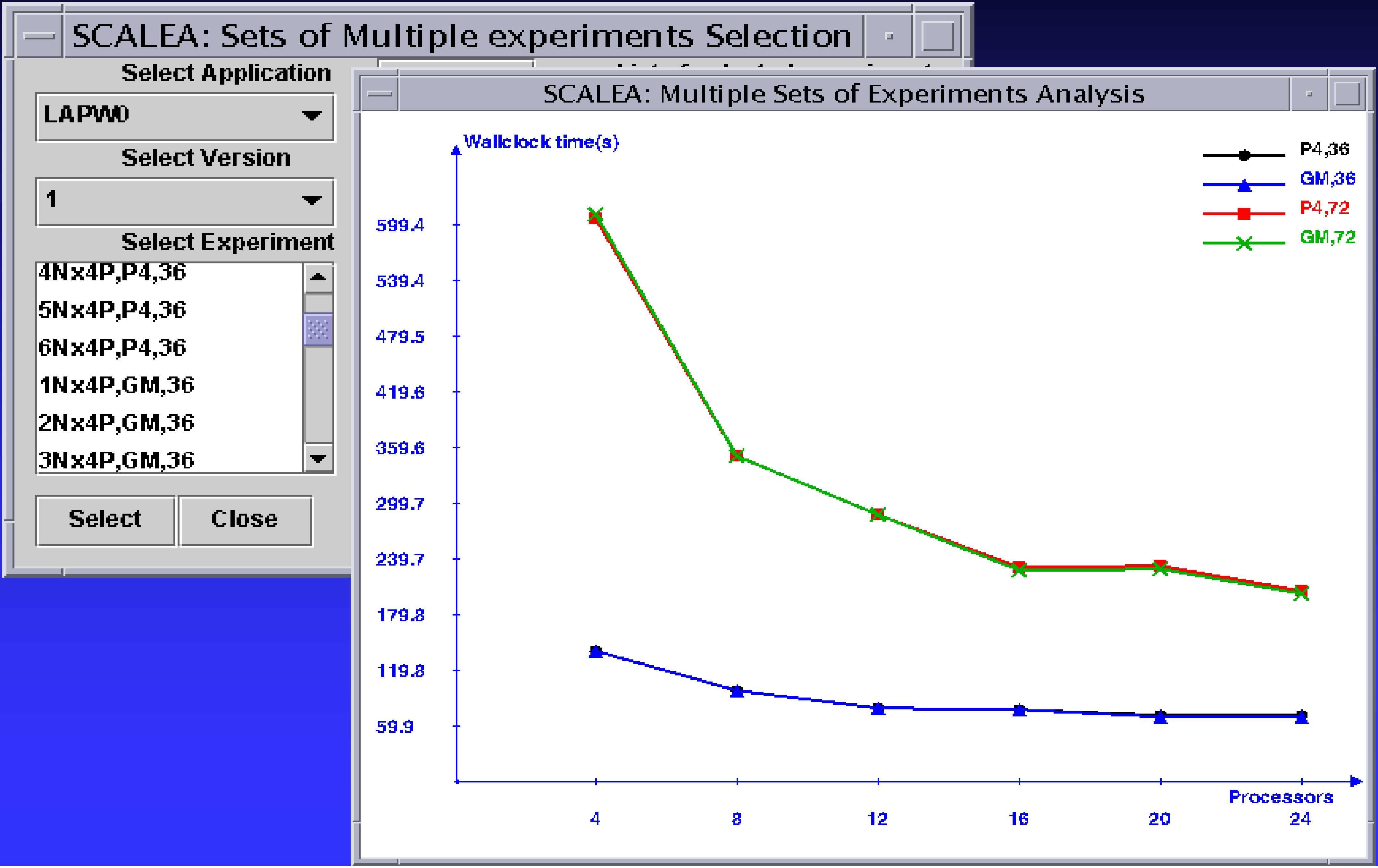
Available Regions

CR_A[IONIZE_MOVE:1281:1788]
CR_A[SET_FIELD_PAR_BACK:1794:1928]
CR_A[CAL_POWER:2244:2323]
CR_MPISEND[MPI_SEND:2301:2303]
CR_MPIRECV[MPI_RECV:2308:2310]
CR_MPISEND[MPI_SEND:1706:1708]
CR_MPIRECV[MPI_RECV:1714:1716]
CR_MPISEND[MPI_SEND:1722:1723]
CR_MPIRECV[MPI_RECV:1730:1731]

Selected R

CR_A[PAR
CR_A[CAL
CR_A[SR_
CR_A[OUT
CR_A[SET
CR_A[CAL

Different Sets of Experiments



SCALEA: Sets of Multiple experiments Selection

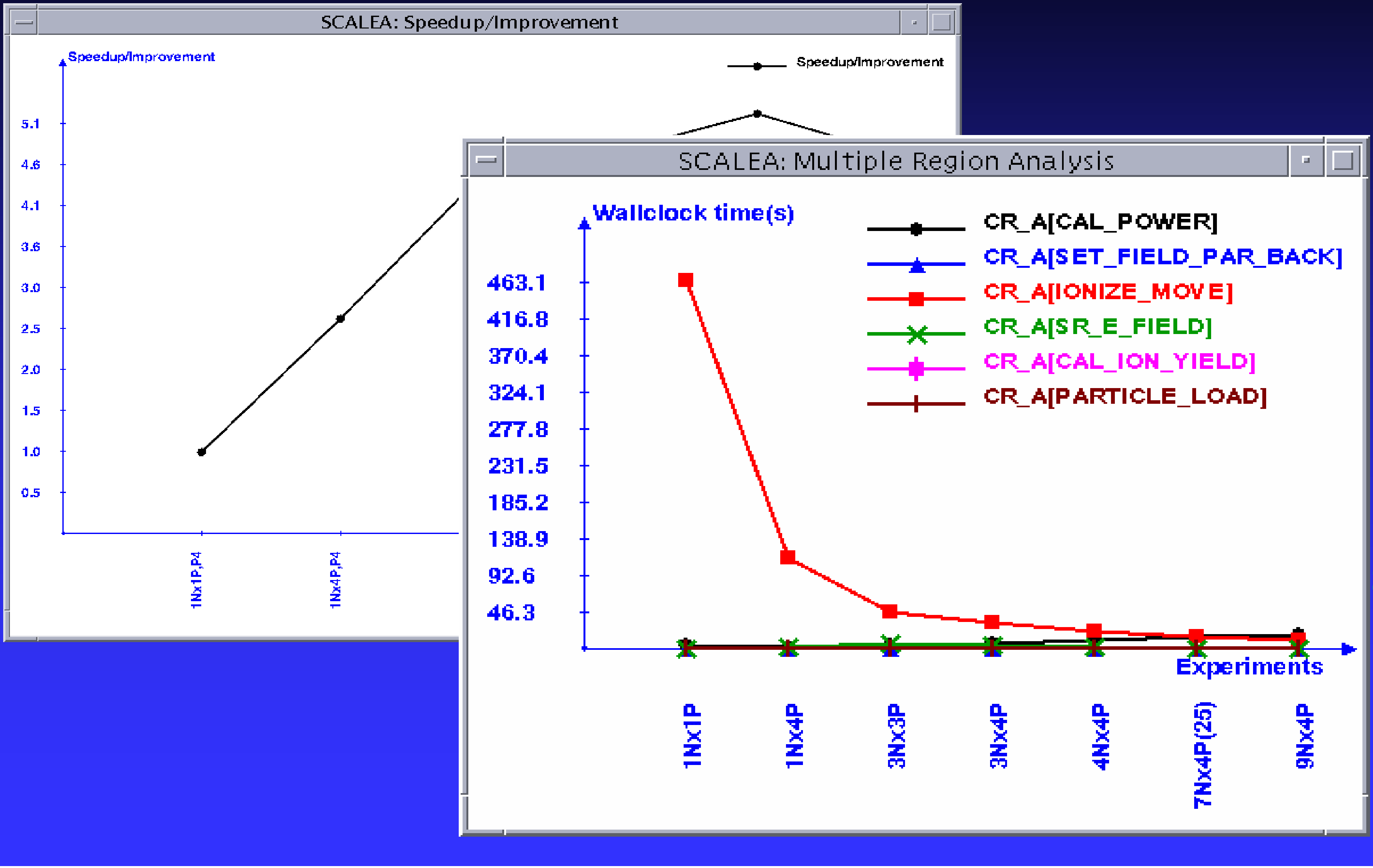
Select Application: LAPW0

Select Version: 1

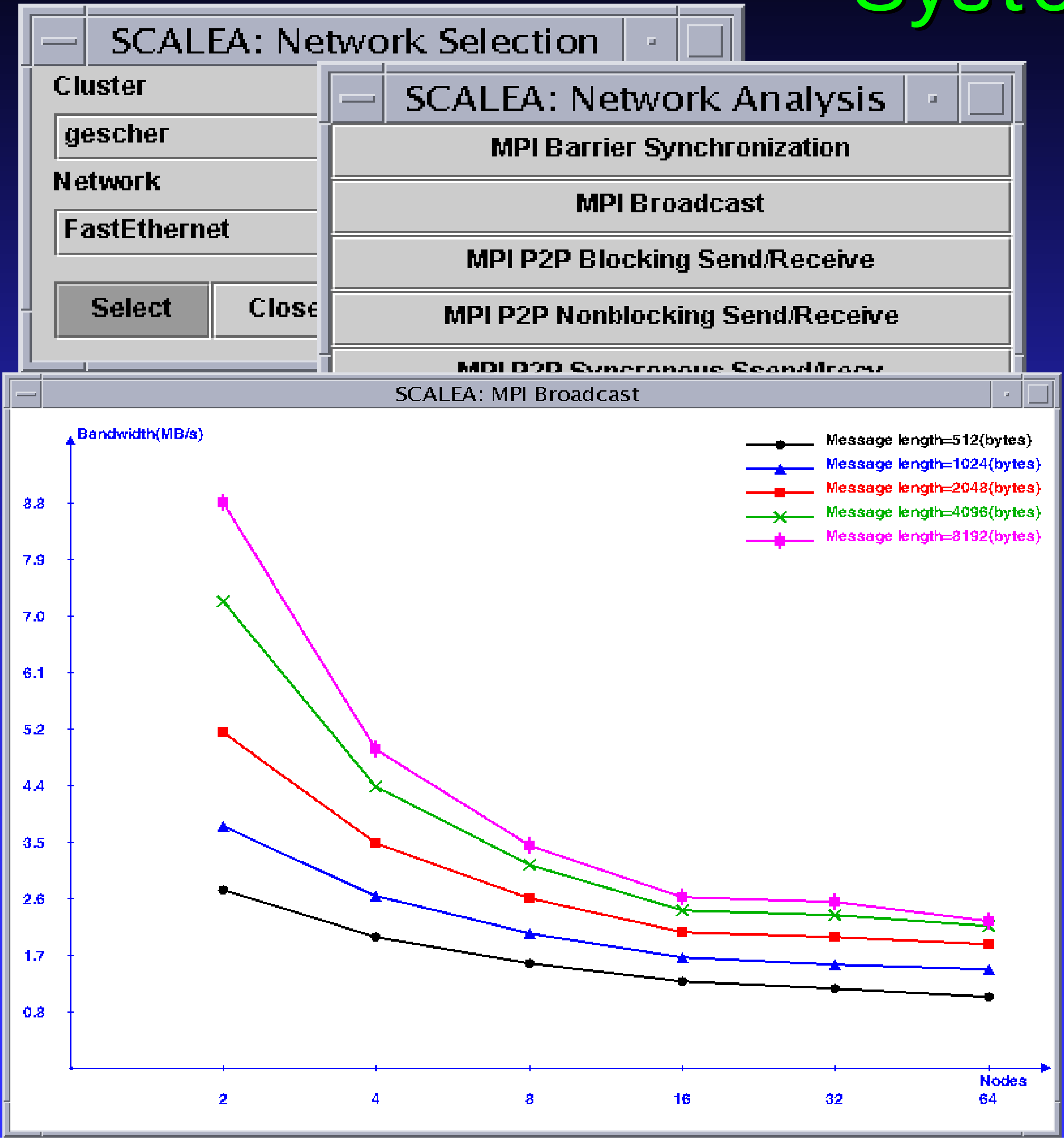
Select Experiment:
- 4Nx4P,P4,36
- 5Nx4P,P4,36
- 6Nx4P,P4,36
- 1Nx4P,GM,36
- 2Nx4P,GM,36
- 3Nx4P,GM,36

SCALEA: Multiple Sets of Experiments Analysis

Improvement/Speedup, Efficiency



System Information



SCALEA: Network Selection

Cluster: gescher

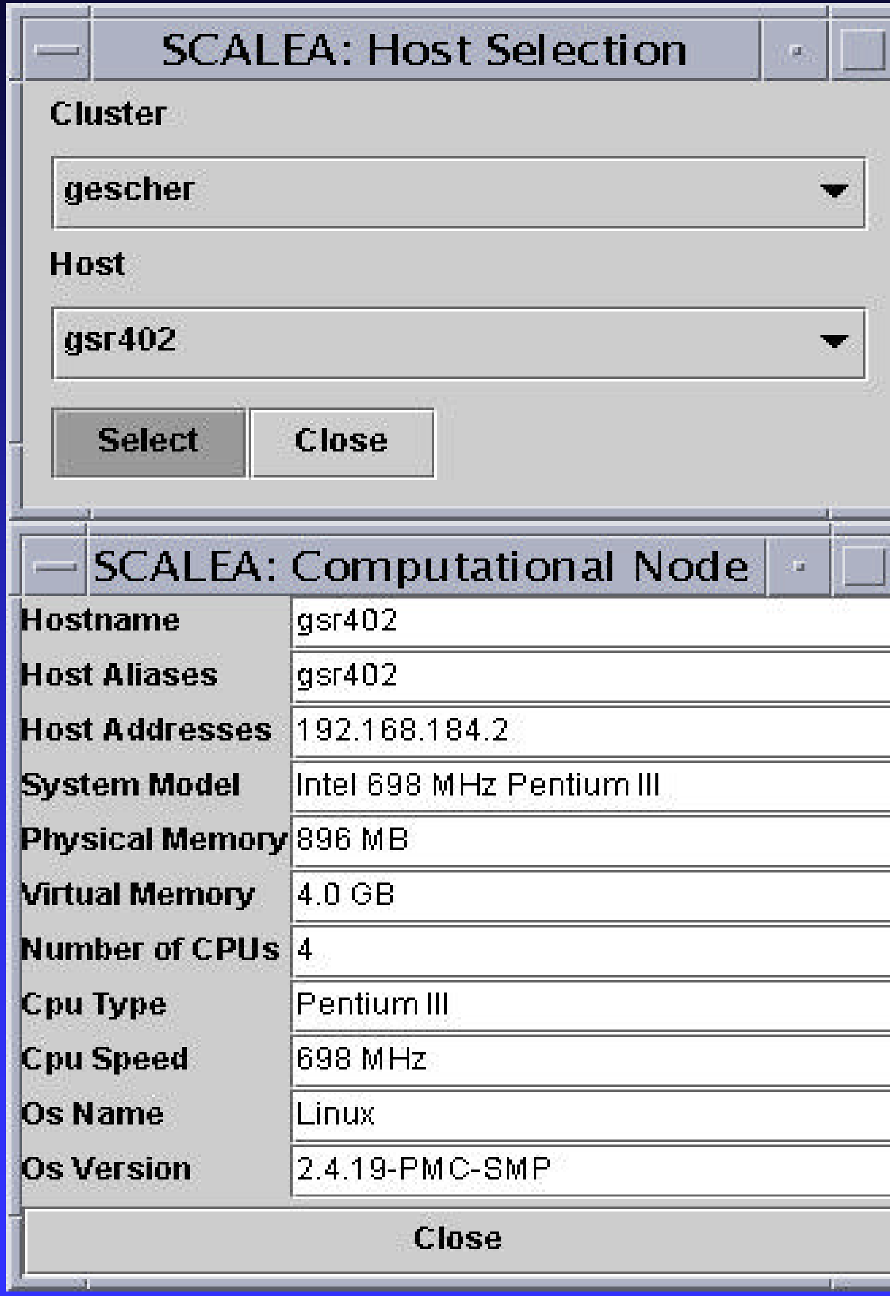
Network: FastEthernet

SCALEA: Network Analysis

MPI Barrier Synchronization
MPI Broadcast
MPI P2P Blocking Send/Receive
MPI P2P Nonblocking Send/Receive
MPI P2P Synchronous Send/recv

SCALEA: MPI Broadcast

Bandwidth(MB/s)
8.8
7.9
7.0
6.1
5.2
4.4
3.5
2.6
1.7
0.8
2
4
8
16
32
64
Nodes
Message length=512(bytes)
Message length=1024(bytes)
Message length=2048(bytes)
Message length=4096(bytes)
Message length=8192(bytes)



SCALEA: Host Selection

Cluster: gescher

Host: gsr402

SCALEA: Computational Node

Hostname	gsr402
Host Aliases	gsr402
Host Addresses	192.168.184.2
System Model	Intel 698 MHz Pentium III
Physical Memory	896 MB
Virtual Memory	4.0 GB
Number of CPUs	4
Cpu Type	Pentium III
Cpu Speed	698 MHz
Os Name	Linux
Os Version	2.4.19-PMC-SMP

Tools Integration

http://www.par.univie.ac.at/project/askalon/

Aksum

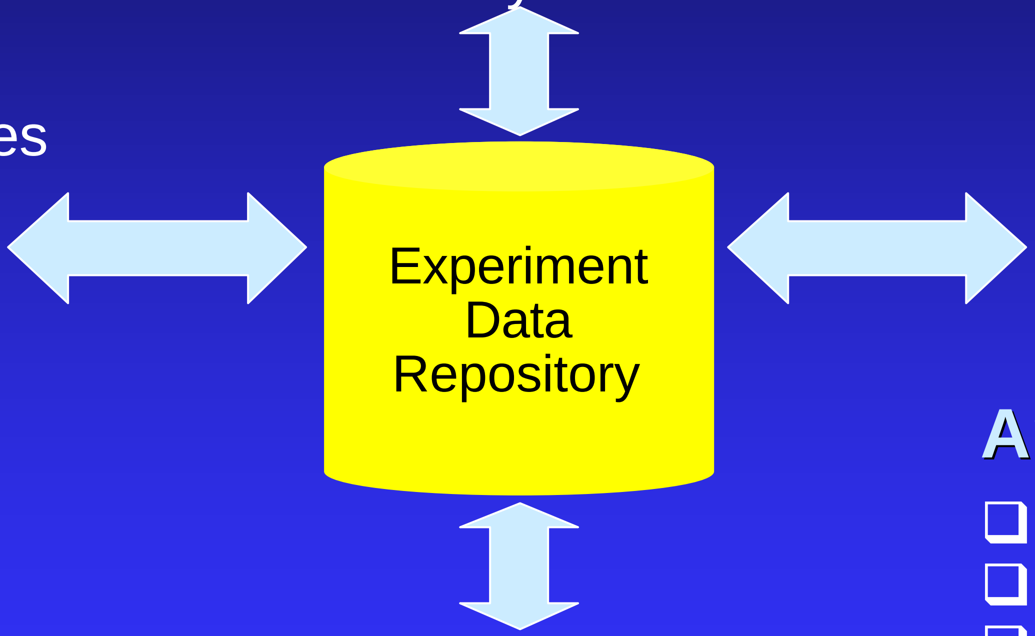
- □ Automatic Bottleneck Analysis

Zenturio

- □ Parameter Studies
- □ Experiment Management

Performance Prophet

- □ Modeling
- □ Simulation
- □ Performance Prediction



Experiment Data Repository

Programming Paradigms

- □ MPI,GlobusMPI
- □ OpenMP/MPI
- □ HPF/OpenMP
- □ JavaSymphony

Architectures

- □ NOWs
- □ PC-Clusters
- □ SMP Clusters
- □ GRID Systems
- □ DM/SM Systems

SCALEA

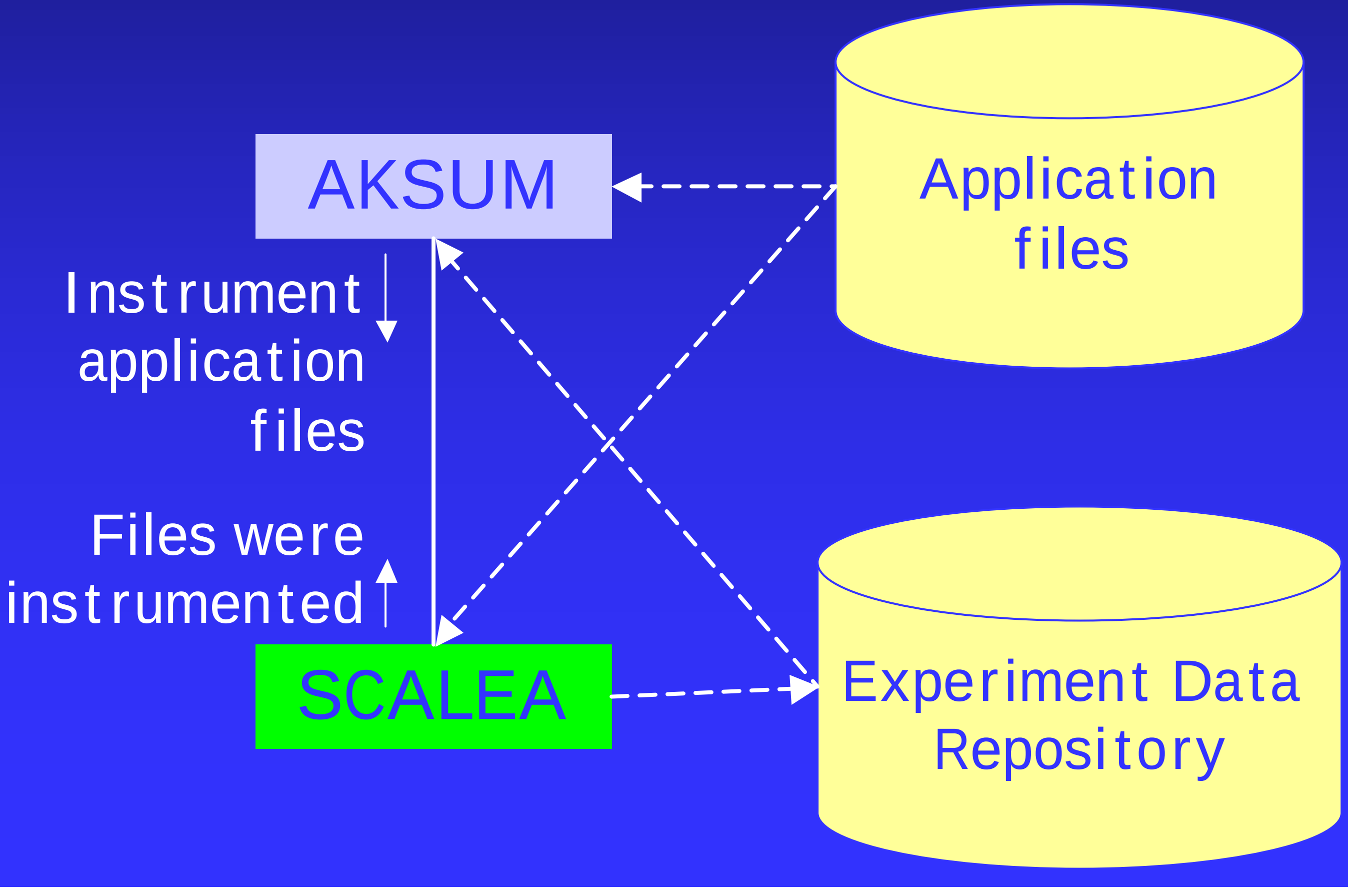
- □ Instrumentation
- □ Measuring
- □ Performance Analysis

Case 1: AKSUM

- URL: http://www.par.univie.ac.at/project/aksum/
- Fahringer and Seragiotto, Jr.

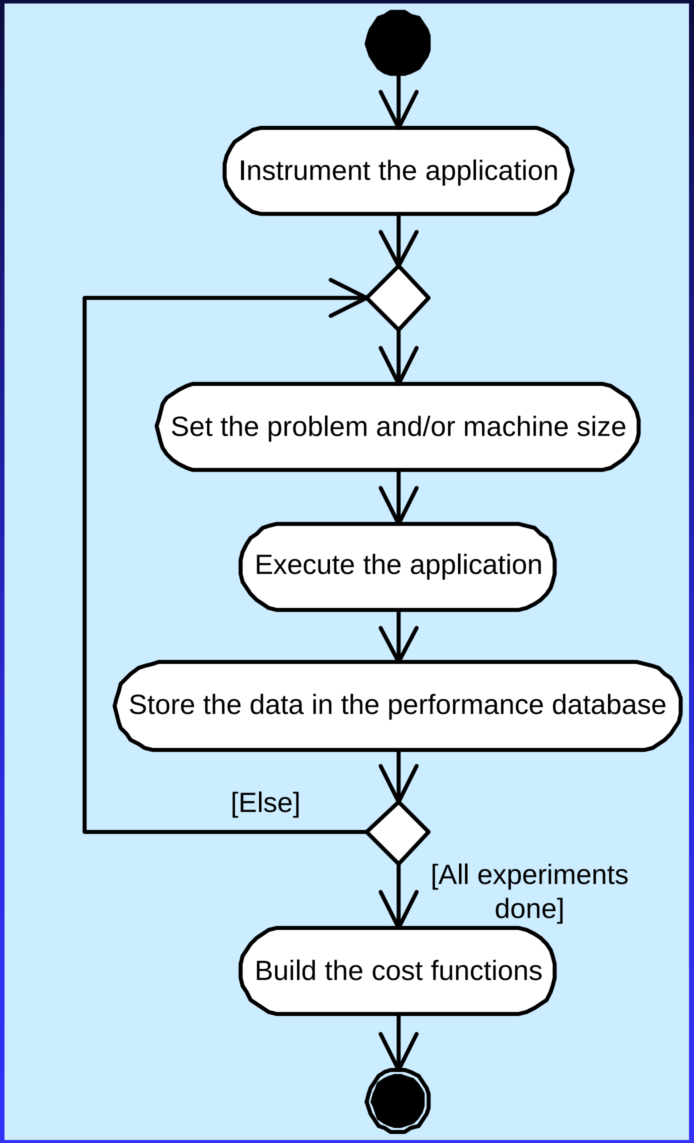
AKSUM employs SCALEA to:

- instrument files given an arbitrary code region
- transfer application files to the repository
- transfer the data generated by the monitoring to the repository
- use data provided by SCALEA for property analysis



Case 2: Performance Prophet

- URL: http://www.par.univie.ac.at/project/prophet/
- Fahringer, Pllana, and Testori
- PP is a performance modeling and prediction system
- PP utilizes SCALEA to obtain timing parameters for application.
  - PP uses performance data in XML format exported by SCALEA for automatic building of cost functions
  - Cost functions are used to develop a hybrid analytical and simulation model of the application


Instrument the application
Set the problem and/or machine size
Execute the application
Store the data in the performance database
[Else]
[All experiments done]
Build the cost functions

Conclusions and Future Work

- Conclusions
  - Design of experiment data repository for performance analysis tool
  - Demonstration of achievements gaining when employing experiment data repository
  - → Data repository has increasingly supported the automation of performance analysis and optimization process
- Ongoing work
  - Working on simple and efficient way to search on performance data
  - Applying automatic scalable analysis techniques
  - Semantic representation of performance data

www.par.univie.ac.at/project/scalea