



HPCS Toolkit: Extending High Performance to High Productivity

David Klepacki
Manager, Advanced Computing Technology
T.J. Watson Research Center

July 2006

Outline

- Vision for Performance Tools
- Roadmap for HPCS Toolkit
- New Technologies Needed for HPCS
- Core Productivity Technology: pSigma
- Current Technology Review: HPC Toolkit
- Productivity Inhibitors of Current Technology
- HPCS Toolkit Deliverables
- Summary

Sanity Check on System Evolution

- Device Scaling imposing **fundamental constraints on system**
Power dissipation and energy consumption
Physical size / packaging
- Pressure to **re-think system architecture**
Blue Gene: low power devices, embedded (small)
Cell: Attached (embedded) co-processing engine
- Systems become inherently **more complex**
Connectivity / hierarchical topology (torus, intra-cell)
Memory constraints (less per processor)
Additional technology “boosts” (hyper-threading, SMT)
- This poses **new challenge to application programming**
New programming paradigm? (not on horizon - legacy codes, ISV apps, etc.)
- Conclusion: **New software tools essential** to mitigate evolving system complexity

Projected Roadmap for a HPCS Toolkit

- Phase I

 - Core Productivity Infrastructure (pSigma)

 - New Tools to further leverage pSigma capability

 - Cache simulator

 - Performance prediction assistant

 - I/O Profiling tools

 - Abstracts Developer from the Source Code Instrumentation**

- Phase II

 - Automation of Performance-Tuning Cycle

 - Intelligent agents to collect performance metrics, mine the information, determine/characterize likely bottlenecks and propose/implement an optimization

 - Investigation of real-time decision system for run-time optimizations

 - Abstracts Developer from the System**

Programmer can be removed from the Performance-Tuning Cycle!

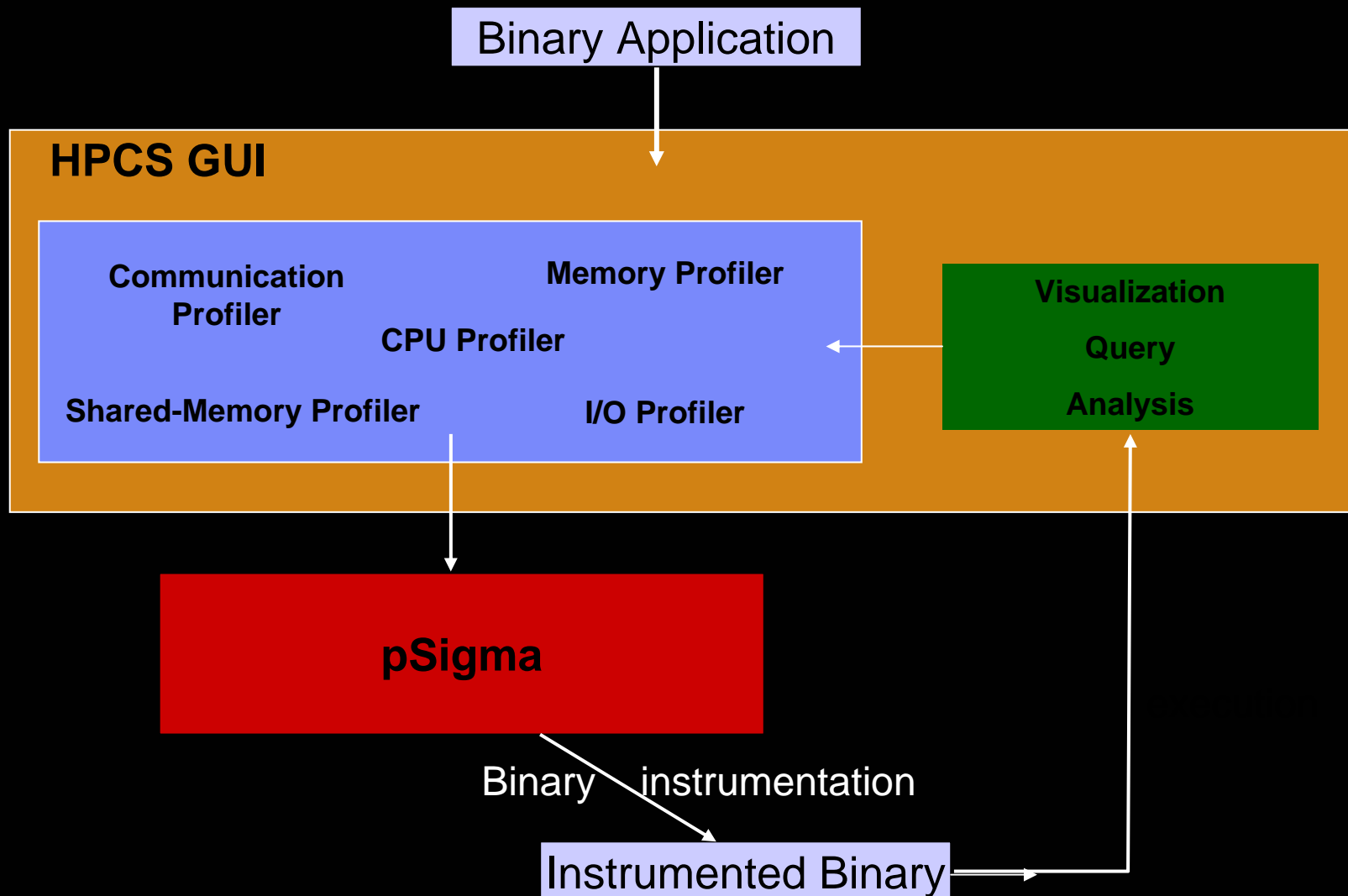
Innovative Technologies for HPCS

- **Completely Binary Approach (pSigma)**
Programmable and dynamic, yet without the need for source code modification.
- **Data-Centric Analysis (DCA)**
For HPCS systems, new tools are needed to provide detailed information on the impact of an **application's data structures** in relation to the underlying hardware.
- **Alternate Scenario Prediction (ASP)**
Data structure layout (“what if my array A was dimensioned like ...”)
Order of a parallel computation, scheduling of threads, etc.
- **User-Controlled Automation (autoPerf)**
Productivity is controlled by degree of automation chosen by programmer.
Can be fully automated if desired.

Core Productivity Technology: pSigma

- pSigma technology was created in response to HPCS challenge. It is the foundation that makes automation possible. **Automation is the key to productivity.**
- pSigma is a general method to insert 'data-collection points' into a binary application **non intrusively** (i.e., without source code modification).
- pSigma provides an **interface between the user and a measurement system** such that the user can interact with it at a higher conceptual level. pSigma translates the user's higher-level input into lower-level instructions (sort of a "compiler" for performance analysis).

The IBM HPCS Toolkit



upGUI

Binary to Patch: ...

Architecture
Events
Instrumentation
Execution

Choose File Event file: EventFile.pSigma-generated ...

Event General Information
 Name: Event Initial State: Enabled

Event Condition Information
 Instruction
 Load In Function: foo For Symbol: array1

Memory Effect of the Instruction
 Miss@L1 In Function: foo For Symbol: array1

Counters Conditions

Counter	Function	Symbol	Operator	Quantity	Function	Symbol
Accesses@L1 <input style="width: 20px;" type="button"/>	foo <input style="width: 20px;" type="button"/>	array1 <input style="width: 20px;" type="button"/>	> <input style="width: 20px;" type="button"/>	0 <input style="width: 20px;" type="button"/>	<input style="width: 20px;" type="button"/>	<input style="width: 20px;" type="button"/>

+
 ↑
 ↓
 -

Event Handler
 Function Name: print_event

Events Summary
 Cache Simulator: Enabled Symbolic Mapping: Enabled

Name	Initial State	Instruction	Memory Effect of the Instru	Counters Condition	Event Handler

+
 ↑

Generate Events File

Data-Centric Analysis Technology (DCA)

peekperf
File Tools

Label	MemTime /	Access
u@Global	1.57702e+06	216097
u@Global_inital	447628	32511
<u>u@Global_calc3</u>	<u>444718</u>	<u>32258</u>
u@Global_calc3z	433822	32258
u@Global_calc1	250850	119070
v@Global	1.57569e+06	216097
p@Global	1.52905e+06	192786
h@Global	1.30777e+06	168216
z@Global	1.30759e+06	168216
pold@Global	1.27448e+06	112144
vold@Global	1.27217e+06	112144
uold@Global	1.27047e+06	112144
cu@Global	1.26258e+06	145158
cv@Global	1.26066e+06	145158
unew@Global	1.19097e+06	81151
vnew@Global	1.15966e+06	81151
pnew@Global	1.15684e+06	81151
psi@Global	486142	51913
unknownDt@Global	11220	539
unknownDt@Local	424	30

```

swim.f
      VOLD(N1,N2), POLD(N1,N2),
2     CU(N1,N2), CV(N1,N2),
*     Z(N1,N2), H(N1,N2), PSI(N1,N2)
C
COMMON /CONS/ DT,TDT,DX,DY,A,ALPHA,ITMAX,MPRINT,M,N,MP1,
1     NP1,EL,PI,TPI,DI,DJ,PCF
C
TDT = TDT+TDT
DO 400 J=1,NP1
DO 400 I=1,MP1
UOLD(I,J) = U(I,J)
VOLD(I,J) = V(I,J)
POLD(I,J) = P(I,J)
U(I,J) = UNEW(I,J)
V(I,J) = VNEW(I,J)
P(I,J) = PNEW(I,J)
400 CONTINUE
RETURN
END
C SPEC removed CCMIC$ MICRO
SUBROUTINE CALC3
C
C TIME SMOOTHER
C
IMPLICIT REAL*8 (A-H, O-Z)
#include "swim.h"

COMMON LI(N1,N2) V(N1,N2) P(N1,N2)

```

Metric Browser: u@Global_calc3

Task	thread	Misses	LoadMisses	StoreMisses	Hits	LoadHits	StoreHits	Access	LoadAccesses	StoreAccess
0	L1	442	193	249	31816	15936	15880	32258	16129	16129
0	L2	190	74	116	16132	119	16013	16322	193	16129
0	L3	0	0	0	1191	74	1117	1191	74	1117
0	TLB	0	0	0	32258	32258	0	32258	32258	0

C SPEC removed CCMIC\$ DO GLOBAL

HPCS Toolkit

© 2006 IBM Corporation

DCA Interactive Cache Tool (Prototype)

Applet Holder
_ □ ×

500

Variable	Acc	Mis
a@Global	~450	~300
b@Global	~450	~0
c@Global	~450	~250

0

mmult

```

0[i][j] = i+j;
}
}
for (i=0; i<n; i++) {
  for (j=0; j<n; j++) {
    c[i][j] = 0;
  }
}
}

void mmult(float c[SIZE][SIZE], float a[SIZE][SIZE], float b[SIZE][SIZE]) {
  int i,j,k;

  for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
      for (k=0; k<n; k++) {
        c[i][j]+a[i][k] * b[k][j];
      }
    }
  }
}
                    
```

Sleep Time:10(ms)

Alternate Scenario Prediction Technology (ASP)

Memory Profile for a1, a2, a3

	L1	L2	TLB
Data: a1			
Load Hit Ratio:	96%	99%	7%
Store Hit Ratio:	99%	16%	-
Data: a2			
Load Hit Ratio:	97%	50%	7%
Store Hit Ratio:	99%	16%	-
Data: a3			
Load Hit Ratio:	96%	59%	0.1%
Store Hit Ratio:	99%	17%	-

Query Repository:

parameter (nx=512, ny=512)

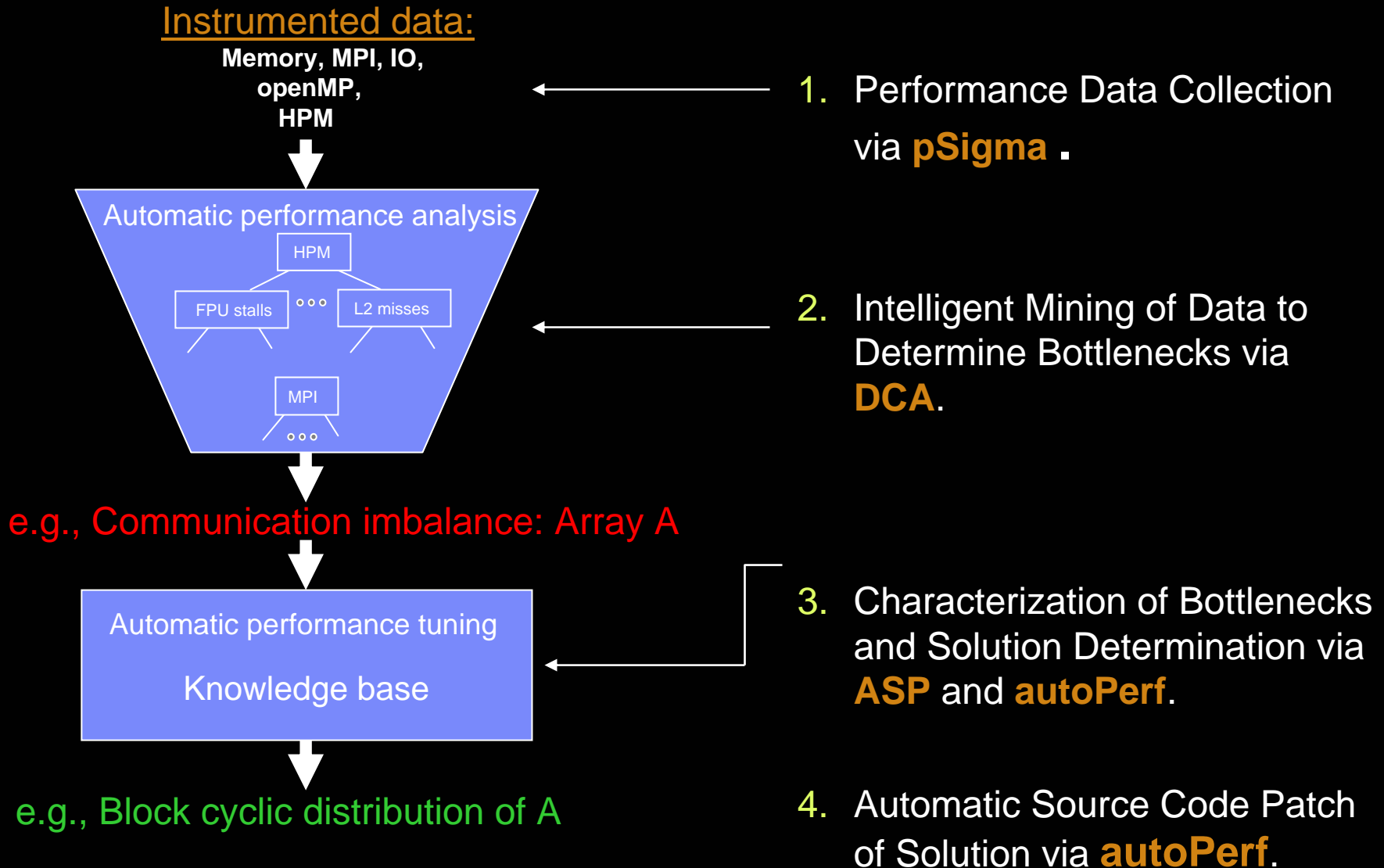
real a1(nx+1,ny), a2(nx+1,ny), a3(nx+1,ny)

...

Predicted Memory Profile for a1, a2, a3

	L1	L2	TLB
Data: a1			
Load Hit Ratio:	96%	99%	99%
Store Hit Ratio:	99%	16%	-
Data: a2			
Load Hit Ratio:	97%	50%	98%
Store Hit Ratio:	99%	16%	-
Data: a3			
Load Hit Ratio:	96%	59%	96%
Store Hit Ratio:	99%	17%	-

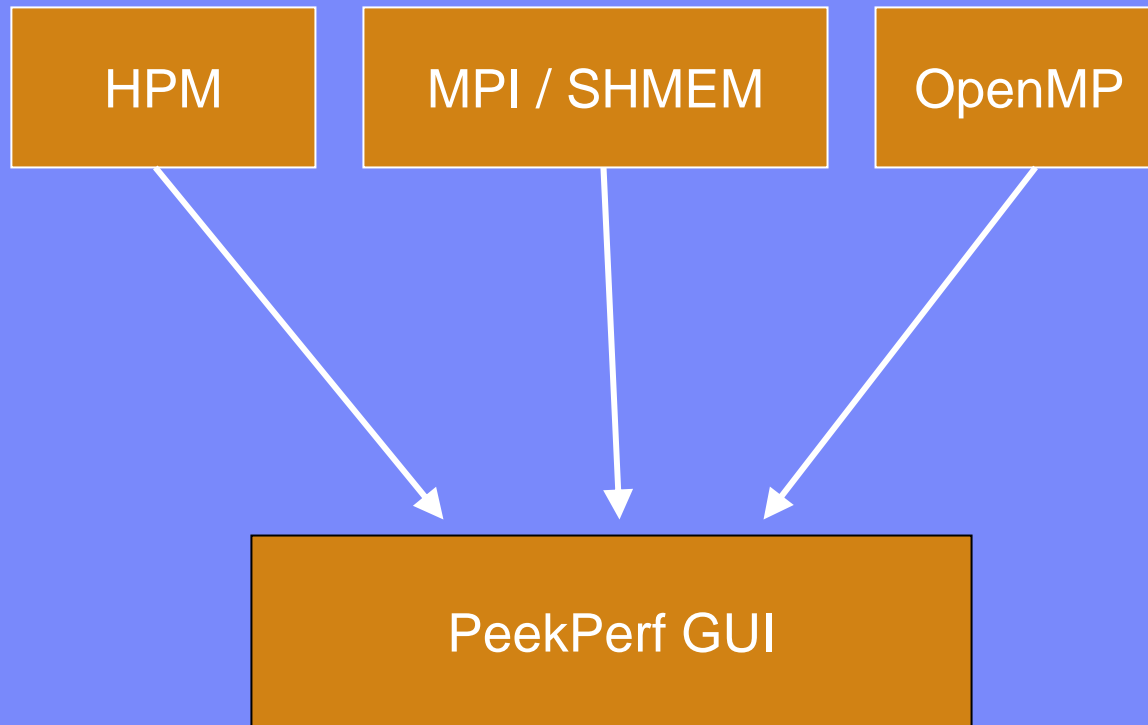
autoPerf Technology (Automation)



Current Technology: The IBM HPC Toolkit

- IBM Corporate strategy for a common application performance analysis environment across all of its HPC platforms:
 - pSeries (POWER, PowerPC) – AIX, Linux**
 - xSeries (Intel, AMD) - Linux**
 - Blue Gene Systems**
 - Standalone or use within Eclipse environment**
- Unified Performance Analysis Environment:
 - Hardware Performance Monitor (HPM)**
 - MPI Profiler and Tracer**
 - Cray SHMEM Porting Library and Profiler**
 - OpenMP Profiler**
 - GUI framework w/ source code traceback**

Unified GUI with Source Code Traceback



Optional Eclipse Environment

Simultaneous Performance Analysis within a Single Framework!

IBM ACTC PeekPerf: Main Window

File Tools Options Action

HPM MPI SIGMA DPOMP

calc1.f calc2.f calc3.f swim_omp.f swim_omp.f calc1.f calc2.f calc3.f

Name

- Probe
 - MPI_Iprobe
 - MPI_Probe
- Recv
 - MPI_Irecv
 - MPI_RECV
- Send
 - Blocking Send
 - MPI_Bsend
 - MPI_Rsend
 - MPI_Send
 - MPI_Ssend
 - Nonblocking Send
 - MPI_Ibsend
 - MPI_Irsend
 - MPI_Isend
 - MPI_Issend
- SendRecv
- Test
 - MPI_Test
 - MPI_Testall
 - MPI_Testany

swim_omp

Label	Count	ExcSec	IncSec
-ALL	1	0.006	1.062
-Calc1	102	0.006	0.263
-Calc2	102	0.072	0.373
-Calc3	100	0.098	0.379
-Inital	1	0.042	0.042
-Loop 100	102	0.119	0.119
-Loop 200	102	0.178	0.178
-Loop 300	100	0.209	0.209
-MPI Calc1 end	102	0.08	0.08
-MPI Calc1 start	102	0.05	0.05
-MPI Calc2 end	102	0.068	0.068
-MPI Calc2 start	102	0.055	0.055
-MPI in Calc3	100	0.072	0.072
-loop 110	102	0.009	0.009

```

101      1 0,2,MPI_COMM_WORLD,req(2),ierr)
102      cALL mpi_irecv(p(m+1,n+1),1,MPI_DOUBLE_PRECISIO
103      1 0,3,MPI_COMM_WORLD,req(3),ierr)
104      cALL mpi_irecv(vold(m+1,n+1),1,MPI_DOUBLE_PRECI
105      1 0,4,MPI_COMM_WORLD,req(4),ierr)
106      cALL mpi_irecv(uold(m+1,n+1),1,MPI_DOUBLE_PRECI
107      1 0,5,MPI_COMM_WORLD,req(5),ierr)
108      cALL mpi_irecv(pold(m+1,n+1),1,MPI_DOUBLE_PRECI
109      1 0,6,MPI_COMM_WORLD,req(6),ierr)
110      endif
111      if(taskid.eq.0)then
112      cALL mpi_isend(v(1,1),1,MPI_DOUBLE_PRECISION,
113      1 numtasks-1,1,MPI_COMM_WORLD,req(7),ierr)
114      cALL mpi_isend(u(1,1),1,MPI_DOUBLE_PRECISION,
115      1 numtasks-1,2,MPI_COMM_WORLD,req(8),ierr)
116      cALL mpi_isend(p(1,1),1,MPI_DOUBLE_PRECISION,
117      1 numtasks-1,3,MPI_COMM_WORLD,req(9),ierr)
118      cALL mpi_isend(vold(1,1),1,MPI_DOUBLE_PRECISION
119      1 numtasks-1,4,MPI_COMM_WORLD,req(10),ierr)
120      cALL mpi_isend(uold(1,1),1,MPI_DOUBLE_PRECISION
121      1 numtasks-1,5,MPI_COMM_WORLD,req(11),ierr)
122      cALL mpi_isend(pold(1,1),1,MPI_DOUBLE_PRECISION
123      1 numtasks-1,6,MPI_COMM_WORLD,req(12),ierr)
124      endif
125      if(taskid.eq.0.or.taskid.eq.numtasks-1)then
126      cALL MPI_WAITALL(12,req,istat,ierr)
127      endif
128      call f_hpmstop( 17 )
129
130      C
131      RETURN
132      END
133
134      SUBROUTINE CALC3Z
135      C
136      C          TIME SMOOTHER FOR FIRST ITERATION
137      C
138      C          PARAMETER (N1=513, N2=513)
139
140      include "mpif.h"
141      COMMON/decomp/js,je,taskid,numtasks,req(16),
142      1 istat(MPI_STATUS_SI
143      integer taskid,req
144      COMMON U(N1,N2), V(N1,N2), P(N1,N2),
145      * UNEW(N1,N2), VNEW(N1,N2),
146      1 PNEW(N1,N2), UOLD(N1,N2),
147      * VOLD(N1,N2), POLD(N1,N2),
    
```

High Performance Computing Toolkit (HPCT) - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.absoft.com/Products/Tools/hpc-toolkit/

absoft® **Leading Fortran/C/C++ Compilers, Debuggers & Development Tools for more than 26 Years**

Products Platforms Order Support My Account Company

Windows | Mac OS X | Linux

High Performance Computing Toolkit (HPCT)

Absoft-IBM HPC Seminar Recording

Product Overview

The IBM High Performance Computing Toolkit is a suite of performance-related tools and libraries to assist in application tuning. This toolkit is an integrated environment for performance analysis of sequential and parallel applications using the MPI and OpenMP paradigms. It provides a common framework for IBM's mid-range server offerings, including pSeries and eSeries servers on Linux.

The HPC Toolkit is available from Absoft as a standalone product or as a component of the [HPC SDK Enhanced Edition for Linux on POWER™](#).

Table of Contents	Purchase Options
<p>Product Information</p> <ul style="list-style-type: none"> ◆ Key Benefits ◆ Key Features ◆ Product Description ◆ Special Features ◆ Availability Date ◆ System Requirements 	<p>Pricing</p> <ul style="list-style-type: none"> ◆ Academic ◆ Government ◆ Commercial
<p>Related Information</p> <ul style="list-style-type: none"> ◆ Compatibility & Companion Products ◆ Technical Support ◆ Order Information ◆ Terms and Conditions ◆ Additional Information / Notices 	

Key Benefits

The IBM High Performance Toolkit includes the following components:

Hardware Performance Monitor
 Provides comprehensive reports of events that are critical to performance on IBM systems. HPM is able to gather the usual timing information, as well as critical hardware performance metrics, such as the number of misses on all cache levels; the number of floating point instructions executed; and the number of instruction loads that cause TLB misses, which all that help the algorithm designer or programmer identify and eliminate performance bottlenecks.

MPI Tracer/Profiler
 Consists of a set of libraries that collect profiling and tracing data for MPI programs. Performance metrics, such as the time used by MPI function calls and message sizes, are reported. MPI tracer works with visualization tools peekperf and peekview to better help users identify performance bottlenecks. Peekperf maps performance metrics back to the

Msg #	Message Size	Num	Count	Processors
01	1024	100	100	1
02	2048	50	50	2
03	4096	25	25	4
04	8192	12	12	8
05	16384	6	6	16
06	32768	3	3	32
07	65536	1	1	64
08	131072	0	0	128
09	262144	0	0	256
10	524288	0	0	512
11	1048576	0	0	1024
12	2097152	0	0	2048
13	4194304	0	0	4096
14	8388608	0	0	8192
15	16777216	0	0	16384
16	33554432	0	0	32768
17	67108864	0	0	65536
18	134217728	0	0	131072
19	268435456	0	0	262144
20	536870912	0	0	524288
21	1073741824	0	0	1048576
22	2147483648	0	0	2097152
23	4294967296	0	0	4194304
24	8589934592	0	0	8388608
25	17179869184	0	0	16777216
26	34359738368	0	0	33554432
27	68719476736	0	0	67108864
28	137438953472	0	0	134217728
29	274877906944	0	0	268435456
30	549755813888	0	0	536870912
31	1099511627776	0	0	1073741824
32	2199023255552	0	0	2147483648
33	4398046511104	0	0	4294967296
34	8796093022208	0	0	8589934592
35	17592186044416	0	0	17179869184
36	35184372088832	0	0	34359738368
37	70368744177664	0	0	68719476736
38	140737488355328	0	0	137438953472
39	281474976710656	0	0	274877906944
40	562949953421312	0	0	549755813888
41	1125899906842624	0	0	1099511627776
42	2251799813685248	0	0	2199023255552
43	4503599627370496	0	0	4398046511104
44	9007199254740992	0	0	8796093022208
45	18014398509481984	0	0	17592186044416
46	36028797018963968	0	0	35184372088832
47	72057594037927936	0	0	70368744177664
48	144115188075855872	0	0	140737488355328
49	288230376151711744	0	0	281474976710656
50	576460752303423488	0	0	562949953421312
51	1152921504606846976	0	0	1125899906842624
52	2305843009213693952	0	0	2251799813685248
53	4611686018427387904	0	0	4503599627370496
54	9223372036854775808	0	0	9007199254740992
55	18446744073709551616	0	0	18014398509481984
56	36893488147419103232	0	0	36028797018963968
57	73786976294838206464	0	0	72057594037927936
58	147573952589676412928	0	0	144115188075855872
59	295147905179352825856	0	0	288230376151711744
60	590295810358705651712	0	0	576460752303423488
61	1180591620717411303424	0	0	1152921504606846976
62	2361183241434822606848	0	0	2305843009213693952
63	4722366482869645213696	0	0	4611686018427387904
64	9444732965739290427392	0	0	9223372036854775808
65	18889465931478580854784	0	0	18446744073709551616
66	37778931862957161709568	0	0	36893488147419103232
67	75557863725914323419136	0	0	73786976294838206464
68	151115727451828646838272	0	0	147573952589676412928
69	302231454903657293676544	0	0	295147905179352825856
70	604462909807314587353088	0	0	590295810358705651712
71	1208925819614629174706176	0	0	1180591620717411303424
72	2417851639229258349412352	0	0	2361183241434822606848
73	4835703278458516698824704	0	0	4722366482869645213696
74	9671406556917033397649408	0	0	9444732965739290427392
75	19342813113834066792898816	0	0	18889465931478580854784
76	38685626227668133585797632	0	0	37778931862957161709568
77	77371252455336267171595264	0	0	75557863725914323419136
78	154742504910672534343190528	0	0	151115727451828646838272
79	309485009821345068686381056	0	0	302231454903657293676544
80	618970019642690137372762112	0	0	604462909807314587353088
81	1237940039285380274745524224	0	0	1208925819614629174706176
82	2475880078570760549491048448	0	0	2417851639229258349412352
83	4951760157141521098982096896	0	0	4835703278458516698824704
84	9903520314283042197964193792	0	0	9671406556917033397649408
85	1980704062856608439592838752	0	0	19342813113834066792898816
86	3961408125713216879185677504	0	0	38685626227668133585797632
87	7922816251426433758371355008	0	0	77371252455336267171595264
88	15845632502852867516742710016	0	0	154742504910672534343190528
89	31691265005705735033485420032	0	0	309485009821345068686381056
90	63382530011411470066970840064	0	0	618970019642690137372762112
91	126765060022822940133941768128	0	0	12379400392852867516742710016
92	253530120045645880267883536256	0	0	2475880078570760549491048448
93	507060240091291760535767072512	0	0	4951760157141521098982096896
94	1014120480182583521071534145024	0	0	9903520314283042197964193792
95	2028240960365167042143068290048	0	0	1980704062856608439592838752
96	4056481920730334084286137180096	0	0	3961408125713216879185677504
97	8112963841460668168572274360192	0	0	7922816251426433758371355008
98	16225927683221336371544548720384	0	0	15845632502852867516742710016
99	32451855366442672743089097440768	0	0	31691265005705735033485420032
100	64903710732885345486179194881536	0	0	63382530011411470066970840064

Productivity Limitations of Current Technology

- Automation not possible
 - Lack of a programmable dynamic binary approach.
- Dependency on instrumentation libraries limits HPCS needs for memory/data-movement optimization
 - Library-based methods are not symbolic:
 - If a measurement is invoked on a memory operation (e.g., L2 cache miss), they cannot provide information about which data structure the memory operation refers.
 - They cannot use symbolic entities of the application source program to describe the measurement event (e.g., “Count L1 cache misses every time the array A is touched”).
 - Library-based methods are not performance-oriented:
 - They cannot provide information about effects on the memory subsystem (e.g., What is the impact to L1 cache performance when loading array A in function foo?).
 - They cannot provide conditional control of the measurement process (e.g., “Count cycles only on each L1 cache miss of loading array A while in function foo...”).

Productivity Limitations of Current Technology

- Dynamic Probe Class Library (DPCL/DynInst) approaches (as used in the HPC Toolkit) are considered state-of-the-art today but have inherent productivity limitations:
 - **System daemons running in the background are required**
 - Acceptable in most cases, but does not optimize productivity needs.
 - **Cannot perform instruction-level instrumentation**
 - The DPCL approach is limited to function-boundary instrumentation only, which meets most HPC needs...but not those of HPCS.
 - **Cannot dynamically respond to user-defined (or automated machine-generated) events and criteria**
 - Requests from the user or intelligent agent cannot be filtered to decide whether to interrupt the execution of the program and invoke an event handler, which is also an automation inhibitor for HPCS.
 - **Not data-centric**
 - Execution cannot be intercepted when something happens to some selected data structure.

HPCS PERCS Proposed Deliverables

- **HPC Toolkit**

 - Integrated HPM, MPI, SHMEM, and OpenMP tools

 - Support for Fortran/95 and C/C++

 - Already established and widely available on multiple platforms

- **Modified to optimize productivity**

 - pSigma technology (binary approach)

 - DCA technology (data centric)

 - ASP technology (prediction)

 - autoPerf technology (automation)

- **Enhancements and New Tools**

 - Support for X10 and UPC and within Eclipse environment

 - Memory profiling and analysis

 - Cache visualization

 - I/O analysis

HPC Toolkit Versus HPCS Toolkit

■ HPC Toolkit

- End of life-cycle for new technology and research
- Will be relegated to commercial product with support for IBM, Intel, AMD, etc. as defined by demand and economy.

■ HPCS Toolkit

- Potential to add “quantum leap” in productivity to application development
- Complex/non-trivial to implement (labor-intensive, expensive)
- Only possible with DARPA backing

Summary of Productivity Impact

- “End-to-End” Optimization of Application Performance-Tuning Cycle

 - Intelligent Automation** (selective removal of programmer from tuning cycle)

 - Eliminates the traditional manual process of collecting performance information, trying to make sense of it, and then doing something about it.

 - Reduces complexity of the system to the application developer.

 - Intelligent Prediction**

 - Alternate scenario evaluation capability provides fast-path to testing effects of altering data structures and/or code structure, without having to change any source code.

 - Automated usage provides solution validation component of intelligent automation.

 - Data Centric Approach**

 - Performance information is directly related to the application data structures.

 - Critical to understanding data movement and memory-related performance of shared memory hierarchical systems intended for HPCS.

 - Source Code Secure**

 - Eliminates user-injected bugs and any unintentional changes to source code.

HPCS Toolkit Community Activity

- **Conference Participation**

 - EuroPar 2005 (Sbaraglia, Eknath, et al.)

 - SC 2005 (Sbaraglia, Eknath, et al.)

 - IPDPS 2006 (Chung, Walkup, et al.)

 - ICS 2006 (Chung; Sbaraglia, Eknath, et al.)

- **Watson Petascale Tools Workshop**

 - SC PIC Supported

 - Participants: IBM, Academia, Government

 - May 3,4 2005

 - May 16,17 2006



Thank you!

Supplemental

MPI Profiler Visualization

PeekPerf Main Window

File Tools Options

Label	Call Count [Ma
...MPI_Allreduce_807	16
...MPI_Allreduce_868	38
...MPI_Barrier_1496	1
...MPI_Barrier_248	1
...MPI_Barrier_765	4
...MPI_Bcast_924	285
...MPI_Comm_rank_973	5
...MPI_Comm_size_972	5
...MPI_Iprobe_1160	
...MPI_Recv_1027	
...MPI_Recv_1135	
...MPI_Ssend_1002	
...Summary_MPI_Allr	
...Summary_MPI_Barr	
...Summary_MPI_Bcas	
...Summary_MPI_Comm	
...Summary_MPI_Comm	
...Summary_MPI_Ipro	
...Summary_MPI_Recv	
...Summary_MPI_Sser	

gamsess.f ddi.f

```

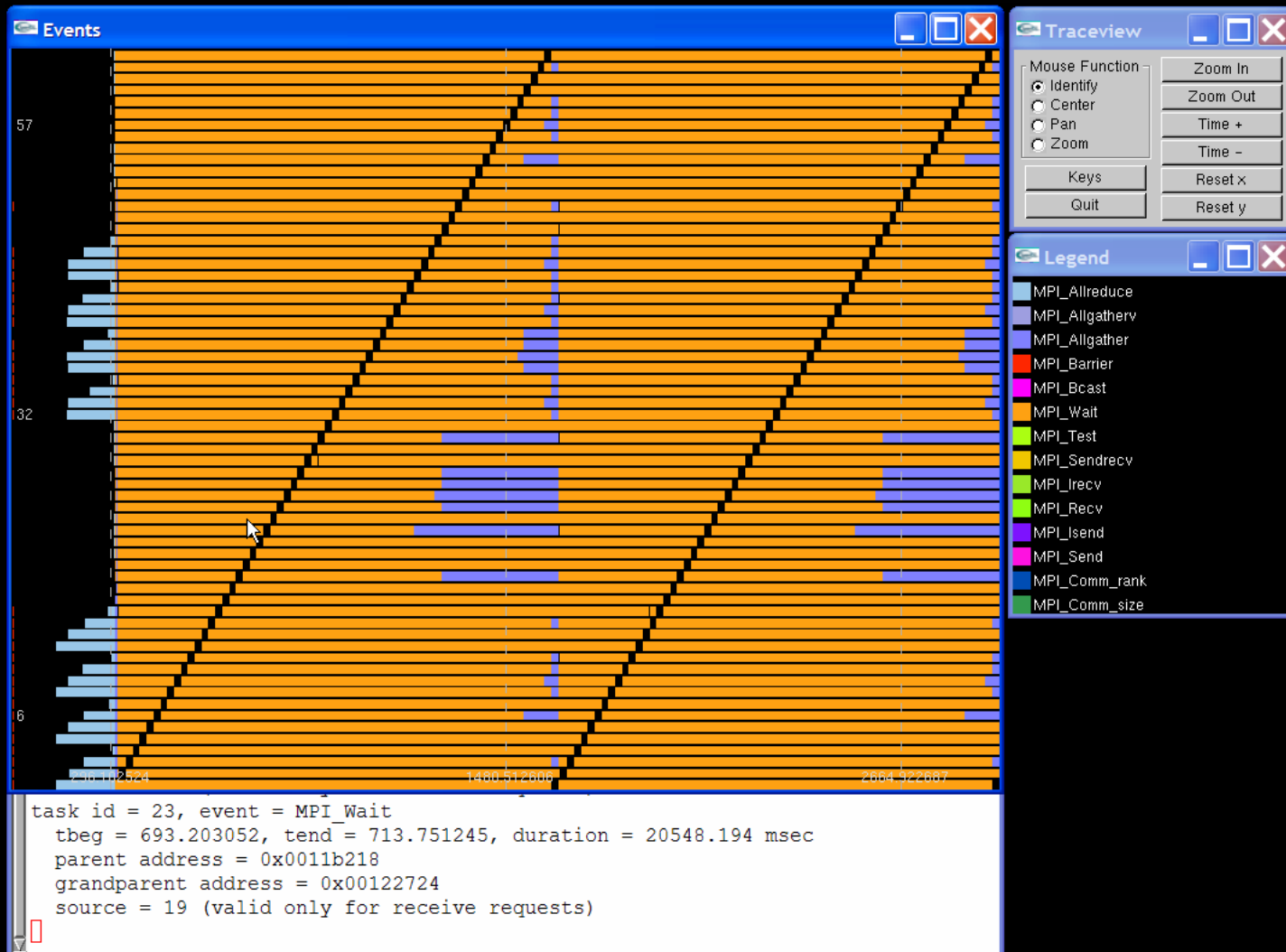
OF COMPUTE
C PROCESSES ONLY, NOT THE TOTAL NUMBER OF
PROCESSES.
C
-----
C
          IMPLICIT NONE
          INTEGER DDI_NP, DDI_ME
C
          INCLUDE 'mpif.h'
          
```

Metric Browser: MPI_Bcast_924

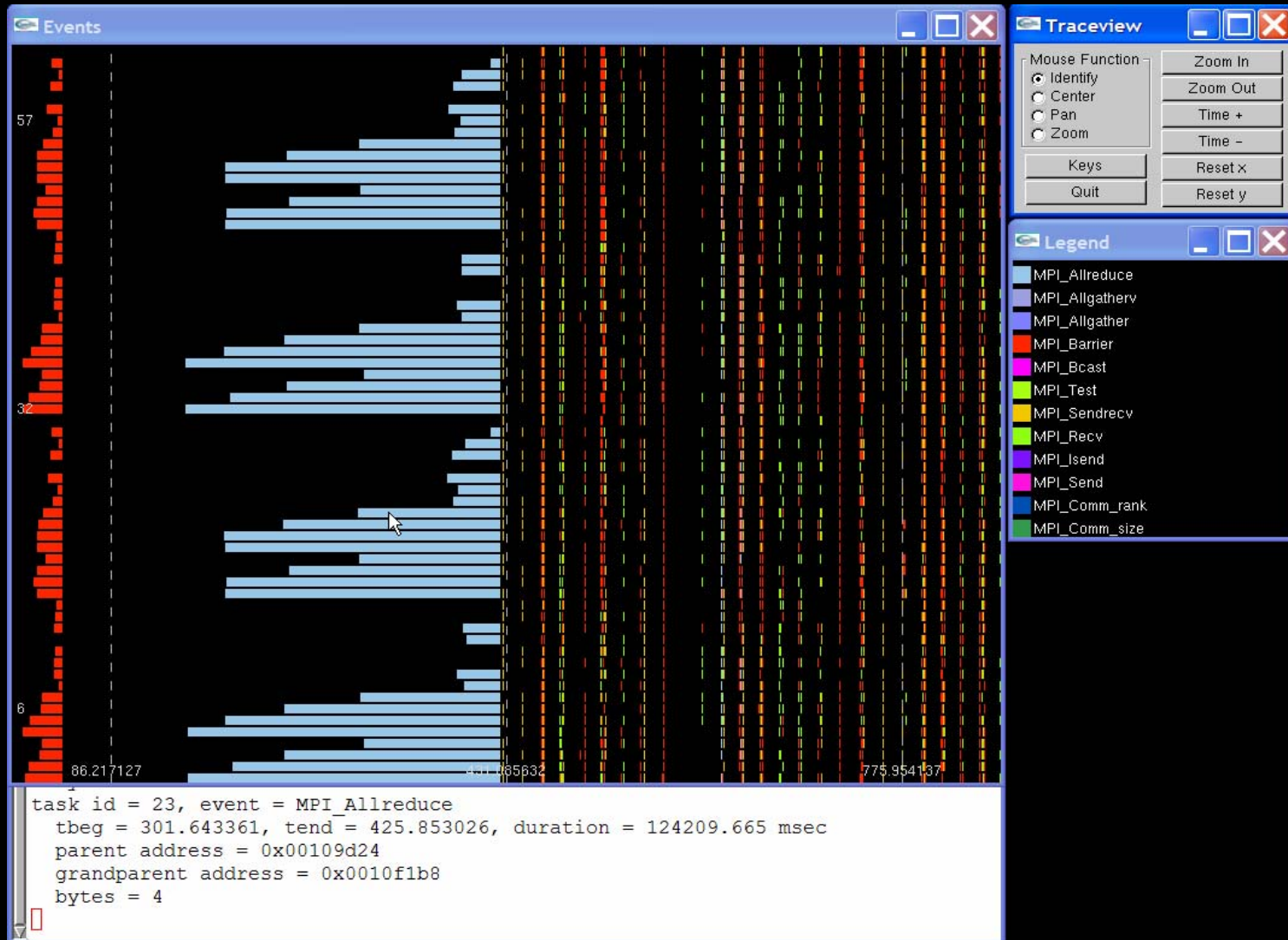
Close Metric Options Precision

Task	Message Size	WallClock	Count	Call Count [Max]	WallClock [Max]	Transferred Bytes
0	(2) 5 ... 16	0.049632	133	133	0.049632	1064
0	(3) 17 ... 64	0.000144	2	2	0.000144	64
0	(4) 65 ... 256	0.001124	16	16	0.001124	1408
0	(5) 257 ... 1K	0.030647	163	163	0.030647	47408
0	(6) 1K ... 4K	0.011084	134	134	0.011084	198472
0	(7) 4K ... 16K	0.024244	285	285	0.024244	1.69084e+06
0	(8) 16K ... 64K	0.001328	16	16	0.001328	227200
0	(9) 64K ... 256K	0.078051	121	121	0.078051	1.09938e+07
1	(2) 5 ... 16	0.185036	133	133	0.185036	1064
1	(3) 17 ... 64	0.000183	2	2	0.000183	64
1	(4) 65 ... 256	0.009773	16	16	0.009773	1408
1	(5) 257 ... 1K	0.018897	163	163	0.018897	47408
1	(6) 1K ... 4K	0.019423	134	134	0.019423	198472
1	(7) 4K ... 16K	0.251916	285	285	0.251916	1.69084e+06
1	(8) 16K ... 64K	0.406296	16	16	0.406296	227200
1	(9) 64K ... 256K	0.087307	121	121	0.087307	1.09938e+07

MPI Tracer Visualization using PeekPerf – Original



MPI Tracer Visualization using PeekPerf – Tuned



OpenMP Profiler Visualization

peekperf
_ _ X

File Tools

main.f
main.f | runhyd3.f

Label	Count	Excl. Time	Incl. Time	%Total Overhead	%Imbalance	A
-region_324	1	54.4638	128.508	4.1e-05	6.3e-05	12
loop_1125	10	13.3219	13.3219	0.005431	88.4638	21
loop_852	10	12.854	12.854	0.005178	52.5563	17
loop_549	10	12.3907	12.3907	0.005676	98.8048	21
loop_1685	10	12.3036	12.3036	0.005682	62.1319	16
loop_1408	10	11.8975	11.8975	0.005578	96.0548	19
loop_1934	10	10.1843	10.1843	0.006926	41.4999	13
loop_790	1	0.522151	0.522151	0.013839	37.576	0.4
loop_1668	1	0.485633	0.485633	0.00896	36.8549	0.4
loop_1623	1	0.039318	0.039318	0.596515	2.79627	0.0
loop_1379	1	0.031769	0.031769	0.198128	13.846	0.0
func_rddparam_174	1	0.01605	0.01605	0	0	0
loop_855	1	0.013306	0.013306	0.224878	342.237	0.0
func_main_155	1	0.000771	128.525	0	0	0
func_deltat_550	1	2.2e-05	2.2e-05	0	0	0
func_layout_296	1	6e-06	6e-06	0	0	0
func_changedir_1841	1	3e-06	3e-06	0	0	0

Metric Browser: loop_1125
_ _ X

Close
Metric Options
Precision

Task	thread	Time in Master	TT: Thread Time	CT: Computation Time	%Imbalance	TO = TT - CT	%TO (Barrier)	%TO (RTL)
0	0	13.3219	13.3219	13.3211	0	0.000723	0.000275	0.005156
0	1	0	15.8615	15.861	19.0664	0.000504	0.000108	0.003679
0	2	0	21.0295	21.029	57.8616	0.000514	0.000103	0.003753
0	3	0	24.4342	24.4338	83.4208	0.000416	4e-06	0.003119
0	4	0	24.1806	24.1802	81.5175	0.000423	0	0.003173
0	5	0	25.0957	25.0952	88.3864	0.00047	1.4e-05	0.00351
0	6	0	25.1062	25.1055	88.4638	0.00062	0.000157	0.004495
0	7	0	23.9859	23.9854	80.0548	0.000556	0.000112	0.004063

```

&      msg_mxzbdy, msg_mxxbdy, msg_mxybdy)
endif

c$omp do schedule(static)
do ip=1,nchunk

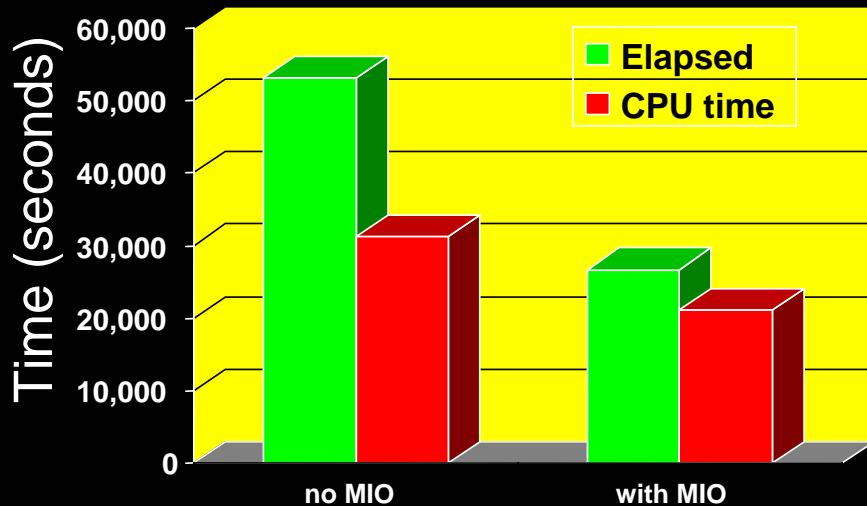
if ((ithread.eq.1).and(ip.ge.icomm_point(1,4)).and.
&      (irec1br.eq.0) .and.      (iflag.eq.0)) then
irec1br = 1

call bdrys1br(dddo, nz, nx, ny, 5,
&      msg_zm, msg_zp, msg_xm, msg_xp, msg_ym, msg_yp,
&      msg_mnzbdy, msg_mnxbdy, msg_mnybdy,
&      msg_mxzbdy, msg_mxxbdy, msg_mxybdy)
call bdry2o1s(dddo, nz, nx, ny, 5,
&      msg_zm, msg_zp, msg_xm, msg_xp, msg_ym, msg_yp,
&      msg_mnzbdy,
&      msg_mnybdy,
&      msg_mxybdy)
xp, msg_ym, msg_yp,
&      msg_mnybdy,
&      msg_mxybdy)
(2,4).and.
0)) then
xp, msg_ym, msg_yp,
&      msg_mnzbdy, msg_mnxbdy, msg_mnybdy,
&      msg_mxzbdy, msg_mxxbdy, msg_mxybdy)
call bdry3o1r(dddo, nz, nx, ny, 5,
    
```

Modular I/O Performance Tool

- I/O Analysis
 - Trace module
 - Summary of File I/O Activity + Binary Events File
 - Low CPU overhead
- I/O Performance Enhancement Library
 - Prefetch module (optimizes asynchronous prefetch and write-behind)
 - System Buffer Bypass capability
 - User controlled pages (size and number)

Example: MSC.Nastran V2001 using Modular I/O



Benchmark:

SOL 111, 1.7M DOF, 1578 modes, 146 frequencies, residual flexibility and acoustics. 120 GB of disk space.

Machine:

4-way, 1.3 GHz p655, 32 GB with 16 GB large pages, JFS striped on 16 SCSI disks.

MSC.Nastran:

V2001.0.9 with large pages, dmp=2 parallel=2 mem=700mb
The run with MIO used mio=1000mb

6.8 TB of I/O in 26666 seconds is an average of about 250 MB/sec

NO source code modifications needed!