

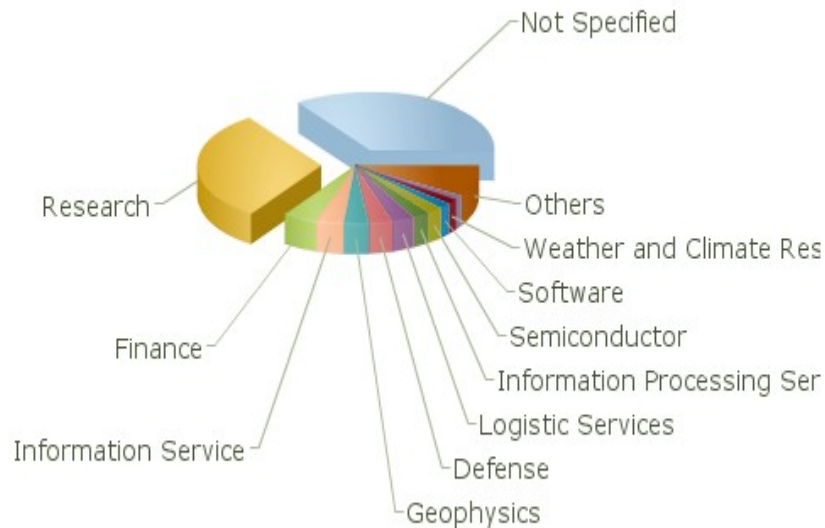


IBM High Performance Computing Toolkit

Pidad D'Souza (pidsouza@in.ibm.com)
IBM, India Software Labs

Top 500 : Application areas (November 2011)

Systems

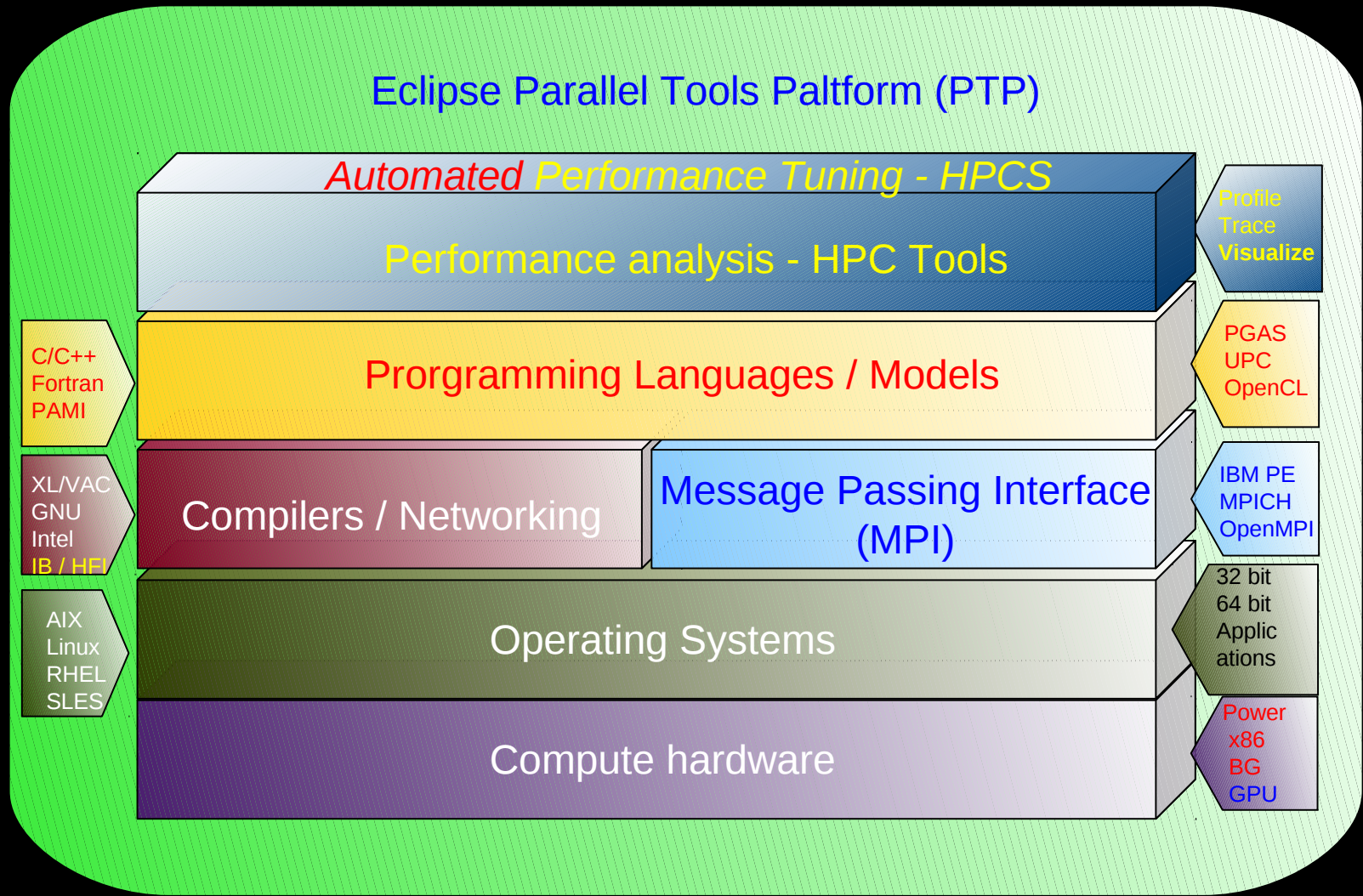


Performance



Source : <http://www.top500.org/charts/list/34/appeara>

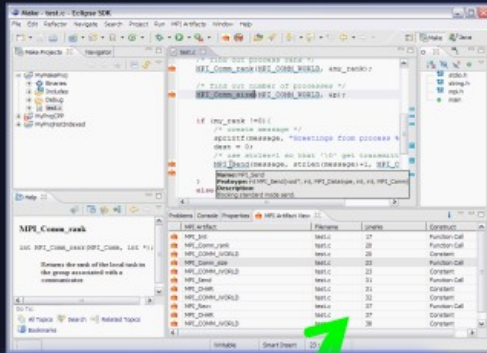
IBM HPC Toolkit – Integrated Picture and dimensions



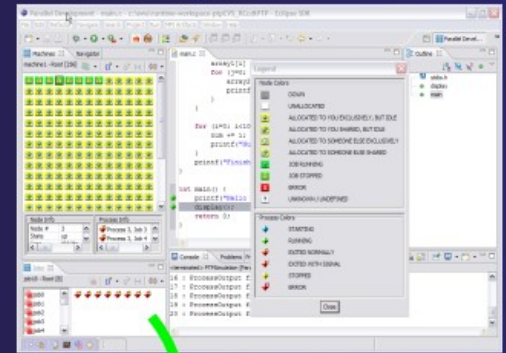
IBM HPC Toolkit

An integrated framework to assist application performance tuning

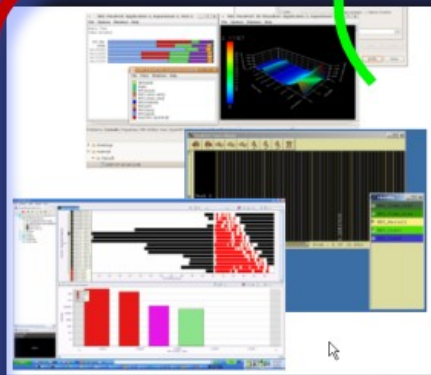
Coding & Analysis



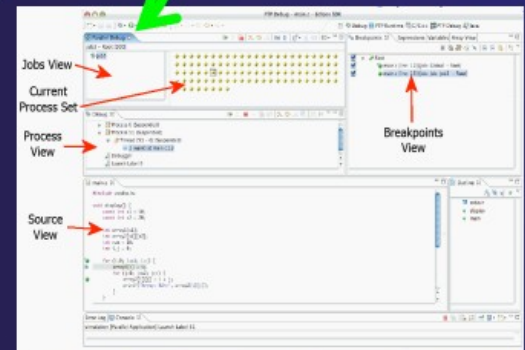
Launching & Monitoring



Parallel Application Development Process

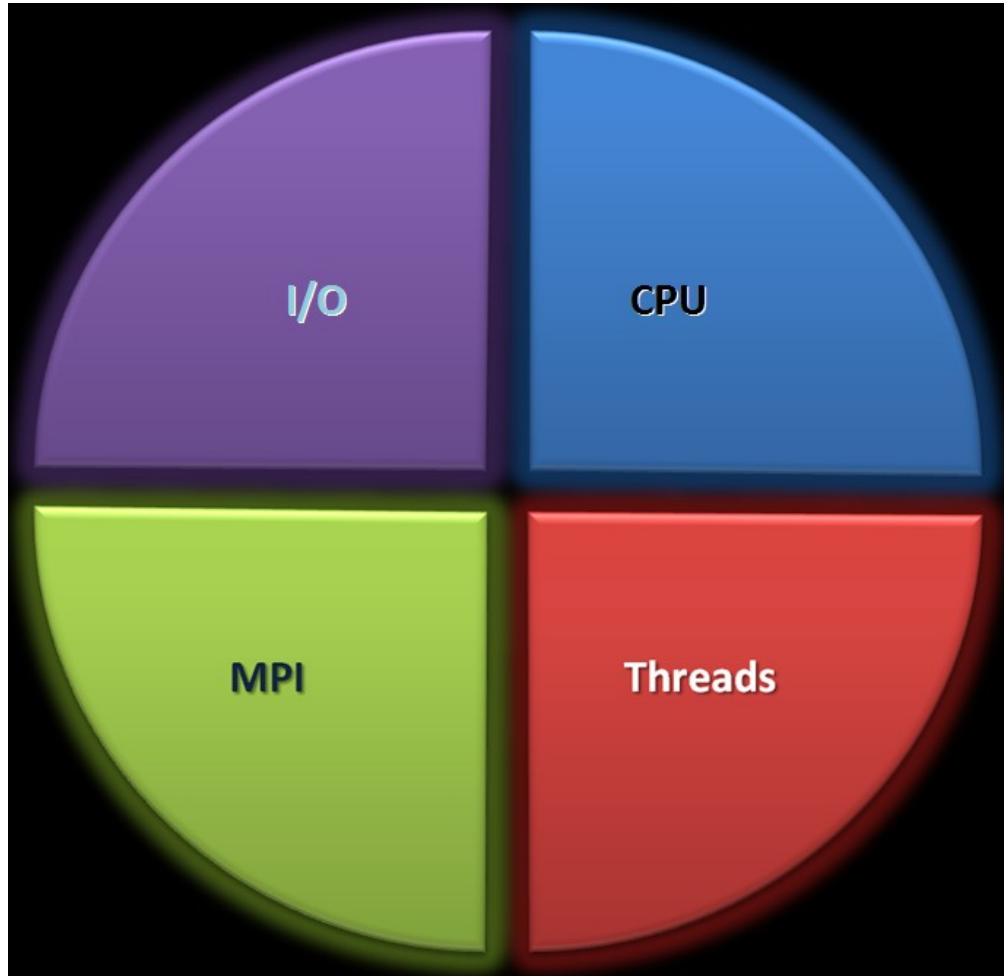


Performance Tuning



Debugging

The Performance Pie



Performance Dimensions

CPU Performance

MPI Performance

Threading Performance

I/O Performance

IBM HPC Toolkit – usability and features

The screenshot shows the Eclipse IDE with the IBM HPC Toolkit. The top editor displays the following code:

```

integer*4 MPI_VERSION, MPI_SUBVERSION
parameter (MPI_VERSION=1, MPI_SUBVERSION=2)

integer*4 MPI_SUCCESS, MPI_ERR_BUFFER, MPI_ERR_COUNT, MPI_ERR_TYPE
integer*4 MPI_ERR_TAG, MPI_ERR_COMM, MPI_ERR_RANK, MPI_ERR_REQUEST
integer*4 MPI_ERR_ROOT, MPI_ERR_GROUP, MPI_ERR_OP, MPI_ERR_TOPOLOGY
integer*4 MPI_ERR_DIMS, MPI_ERR_ARG, MPI_ERR_UNKNOWN
integer*4 MPI_ERR_TRUNCATE
integer*4 MPI_ERR_OTHER, MPI_ERR_INTERN, MPI_ERR_IN_STATUS
integer*4 MPI_ERR_PENDING, MPI_ERR_PENDING, MPI_ERR_INFO_KEY
integer*4 MPI_ERR_INFO_VALUE, MPI_ERR_INFO_NOKEY, MPI_ERR_INFO
integer*4 MPI_ERR_FILE, MPI_ERR_NOT_SAME, MPI_ERR_AMODE
integer*4 MPI_ERR_UNSUPPORTED_DATAREP
integer*4 MPI_ERR_UNSUPPORTED_OPERATION
integer*4 MPI_ERR_NO_SUCH_FILE, MPI_ERR_FILE_EXISTS
integer*4 MPI_ERR_BAD_FILE
integer*4 MPI_ERR_ACCESS, MPI_ERR_NO_SPACE, MPI_ERR_QUOTA
integer*4 MPI_ERR_READ_ONLY, MPI_ERR_FILE_IN_USE
integer*4 MPI_ERR_DUP_DATAREP
integer*4 MPI_ERR_CONVERSION, MPI_ERR_IO
integer*4 MPI_ERR_WIN, MPI_ERR_BASE, MPI_ERR_SIZE, MPI_ERR_DISP
integer*4 MPI_ERR_LOCKTYPE, MPI_ERR_ASSERT, MPI_ERR_RMA_CONFLICT
integer*4 MPI_ERR_RMA_SYNC, MPI_ERR_NO_NEM
integer*4 MPI_ERR_KEYVAL
integer*4 MPI_ERR_LASTCODE
parameter (MPI_SUCCESS=0, MPI_ERR_BUFFER=50, MPI_ERR_COUNT=51)
parameter (MPI_ERR_TYPE=52, MPI_ERR_TAG=53, MPI_ERR_COMM=54)
parameter (MPI_ERR_RANK=55, MPI_ERR_REQUEST=56, MPI_ERR_ROOT=57)

```

The bottom window shows a performance data visualization with a legend on the right:

- MPI_Comm_spawn
- MPI_Comm_spawn_multiple
- MPI_Send
- MPI_Recv
- MPI_Barrier

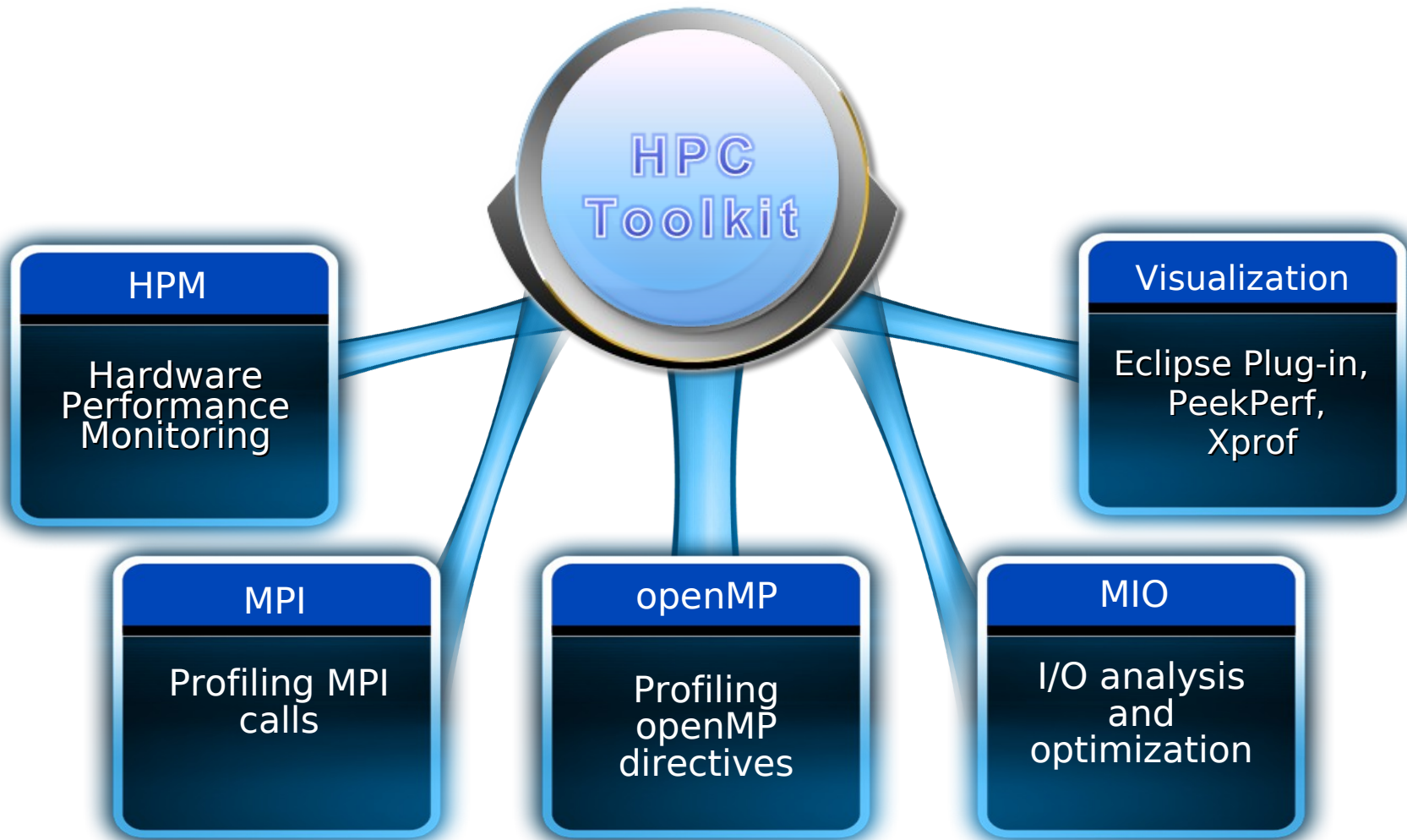
- IBM System p, Blue Gene L/P, *System x, Hybrid*
- AIX and Linux

- Operates on the application binary and generate **results** in terms of source level symbols, with full **source code traceback** capability

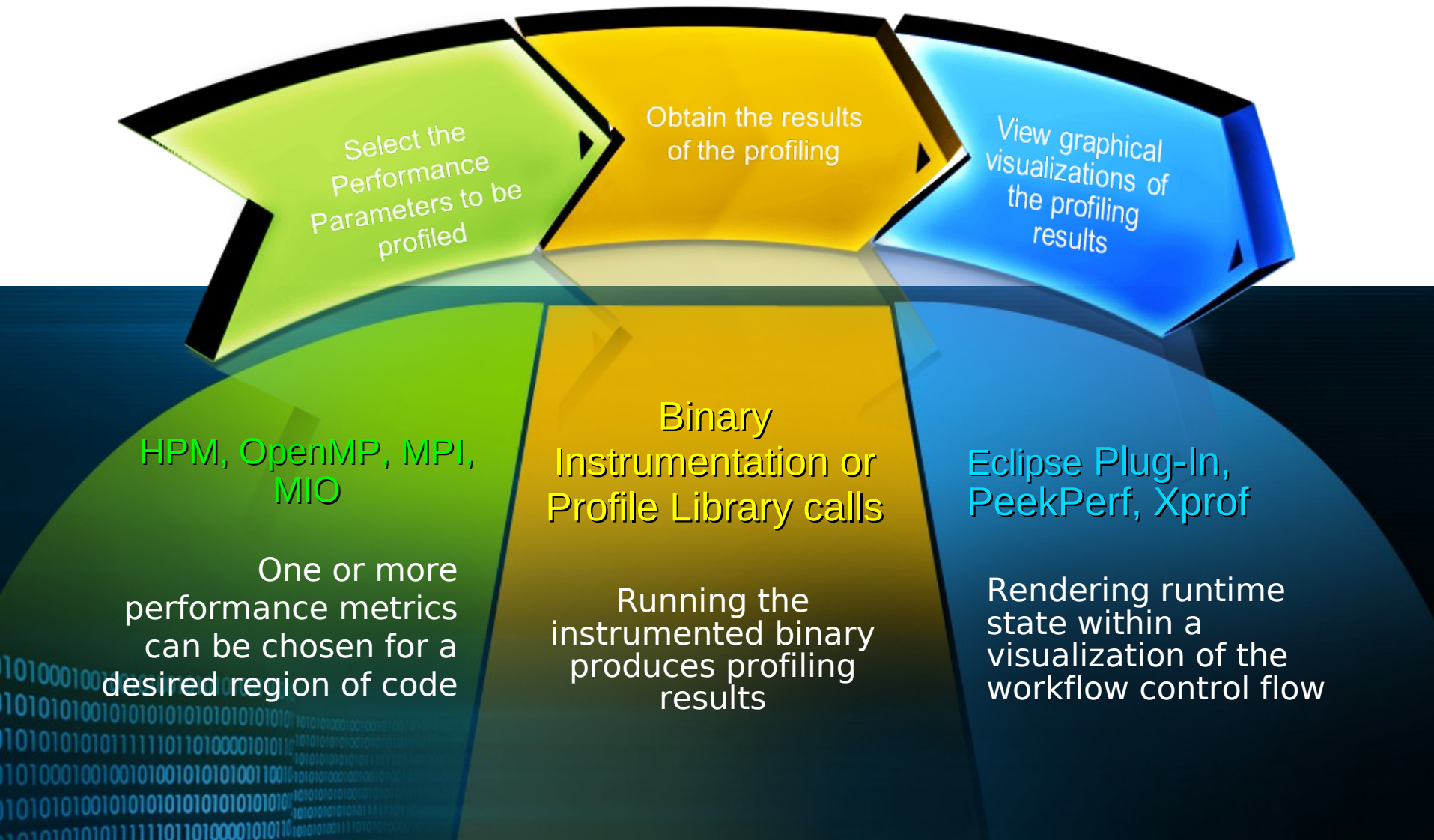
- Presents the results in an easy to use and intuitive **GUI**

- Can be used as:
 - A collection of **command line**, visualization utilities
 - A **plug-in into the Eclipse IDE**
 - As a collection of libraries which are used with the application

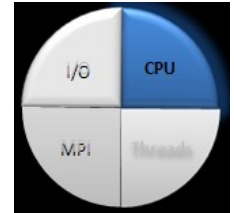
IBM HPC Toolkit Components



Using the IBM HPC Toolkit



CPU Performance – HPM module



Utilizes **Performance Counters** –
(special purpose registers built into the processor to store the counts of hardware-related activities)

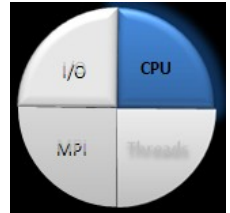
Facilitates **low-level** performance analysis and tuning

Updated at each cycle and **low overhead**

HPM: Hardware Counters Examples

- Cycles
 - Instructions
 - Floating point instructions
 - Integer instructions
 - Load/stores
 - Cache misses
 - TLB misses
 - Branch taken / not taken
 - Branch mispredictions
- Useful derived metrics
 - ✓ **IPC - instructions per cycle**
 - ✓ **Float point rate (Mflop/s)**
 - ✓ **Computation intensity**
 - ✓ **Instructions per load/store**
 - ✓ **Load/stores per cache miss**
 - ✓ **Cache hit rate**
 - ✓ **Loads per load miss**
 - ✓ **Stores per store miss**
 - ✓ **Loads per TLB miss**
 - ✓ **Branches mispredicted %**

CPU Performance



libhpm v3.2.1 (IHPCT v2.2.0) summary

Resource Usage Statistics

Total amount of time in user mode :
6.732208 seconds
Total amount of time in system mode :
5.174914 seconds
Maximum resident set size : 12184
Kbytes
Average shared memory use in text segment :
17712 Kbytes*sec
Average unshared memory use in data segment :
61598 Kbytes*sec
Number of page faults without I/O activity : 13829
Number of page faults with I/O activity : 0
Number of times process was swapped out : 0
Number of times file system performed INPUT : 0
Number of times file system performed OUTPUT : 0
Number of IPC messages sent : 0
Number of IPC messages received : 0
Number of signals delivered : 0
Number of voluntary context switches : 233
Number of involuntary context switches : 684
End of Resource Statistics

Instrumented section: 7 - Label: find_my_seed
process: 274706, thread: 1
file: is.c, lines: 412 <--> 441
Context is process context.
No parent for instrumented section.

Inclusive timings and counter values:

Execution time (wall clock time) : 0.000290516763925552 seconds

Init
Ovi
PM
PM
PM
PM
PM
PM
Util
MI
Ins
Alg
Alg
Alg
FM
%

hpmviz

File	Label	ExcSec	IncSec	Count
swim_omp	Loop 300	4.572	4.572	2398
	Loop 200	4.203	4.203	2400
	Loop 100	3.071	3.071	2400
	Calc3	1.838	6.813	2398
	Calc2	1.013	5.632	2400
	MPI Calc1 start 1.003	1.003	2400	
	MPI Calc2 start 0.528	0.528	2400	

Node	Thread	Count	ExcSec	IncSec	U time	Use rate	(M) LS	MIPS	HW FP/Cyc	Instr/LS	M Flips	IpC	Mtlp/s	WFlps	Wtlp/s	FMA %	Comp Int.
0	3	2398	4.538	4.538	3.923	86.425	590.056	261.855	0.116	2.245	589.86	0.26	129.947	589.86	129.947	80.012	1
0	0	2398	4.572	4.572	4.378	95.783	608.414	263.277	0.107	1.978	608.234	0.211	133.037	608.234	133.037	80.011	1
0	2	2398	4.548	4.548	4.366	95.879	590.019	265.241	0.104	1.868	589.838	0.205	129.663	589.838	129.663	80.011	1
0	1	2398	4.547	4.547	4.308	94.759	590.024	259.19	0.105	1.997	589.837	0.21	129.728	589.837	129.728	80.012	1
1	2	2398	4.534	4.534	4.398	96.999	590.044	253.123	0.103	1.945	589.856	0.201	130.088	589.856	130.088	80.012	1
1	1	2398	4.528	4.528	3.942	87.069	589.983	266.058	0.115	2.195	589.807	0.233	130.263	589.807	130.263	80.011	1
1	0	2398	4.517	4.517	3.766	82.838	608.431	308.065	0.124	2.302	608.214	0.286	133.762	608.214	133.762	80.011	1
1	3	2398	4.523	4.523	3.537	78.198	589.962	317.346	0.128	2.433	589.781	0.312	130.4	589.781	130.4	80.011	1
2	0	2398	4.538	4.538	3.777	83.218	608.448	312.01	0.124	2.327	608.282	0.288	134.029	608.282	134.029	80.012	1
2	2	2398	4.522	4.522	4.313	95.384	590.033	257.962	0.105	1.977	589.86	0.208	130.431	589.86	130.431	80.011	1
2	3	2398	4.52	4.52	4.307	95.285	589.985	258.863	0.105	1.983	589.806	0.209	130.492	589.806	130.492	80.011	1
2	1	2398	4.52	4.52	4.35	96.222	589.943	255.814	0.104	1.98	589.787	0.205	130.468	589.787	130.468	80.01	1
3	3	2398	4.487	4.487	4.193	93.453	571.551	259.827	0.105	2.04	571.374	0.214	127.352	571.374	127.352	80.011	1
3	1	2398	4.502	4.502	4.365	96.853	589.937	254.196	0.104	1.94	589.763	0.202	131.003	589.763	131.003	80.01	1
3	2	2398	4.483	4.483	4.138	92.33	571.556	263.864	0.106	2.07	571.38	0.22	127.445	571.38	127.445	80.011	1
3	0	2398	4.506	4.506	3.927	87.154	590.044	260.852	0.116	2.221	589.856	0.257	130.901	589.856	130.901	80.012	1

Metric Browser: Loop 300

Node	Thread	Count	ExcSec	IncSec	U time	Use rate	(M) LS	MIPS	HW FP/Cyc	Instr/LS	M Flips	IpC	Mtlp/s	WFlps	Wtlp/s	FMA %	Comp Int.
0	3	2398	4.538	4.538	3.923	86.425	590.056	261.855	0.116	2.245	589.86	0.26	129.947	589.86	129.947	80.012	1
0	0	2398	4.572	4.572	4.378	95.783	608.414	263.277	0.107	1.978	608.234	0.211	133.037	608.234	133.037	80.011	1
0	2	2398	4.548	4.548	4.366	95.879	590.019	265.241	0.104	1.868	589.838	0.205	129.663	589.838	129.663	80.011	1
0	1	2398	4.547	4.547	4.308	94.759	590.024	259.19	0.105	1.997	589.837	0.21	129.728	589.837	129.728	80.012	1
1	2	2398	4.534	4.534	4.398	96.999	590.044	253.123	0.103	1.945	589.856	0.201	130.088	589.856	130.088	80.012	1
1	1	2398	4.528	4.528	3.942	87.069	589.983	266.058	0.115	2.195	589.807	0.233	130.263	589.807	130.263	80.011	1
1	0	2398	4.517	4.517	3.766	82.838	608.431	308.065	0.124	2.302	608.214	0.286	133.762	608.214	133.762	80.011	1
1	3	2398	4.523	4.523	3.537	78.198	589.962	317.346	0.128	2.433	589.781	0.312	130.4	589.781	130.4	80.011	1
2	0	2398	4.538	4.538	3.777	83.218	608.448	312.01	0.124	2.327	608.282	0.288	134.029	608.282	134.029	80.012	1
2	2	2398	4.522	4.522	4.313	95.384	590.033	257.962	0.105	1.977	589.86	0.208	130.431	589.86	130.431	80.011	1
2	3	2398	4.52	4.52	4.307	95.285	589.985	258.863	0.105	1.983	589.806	0.209	130.492	589.806	130.492	80.011	1
2	1	2398	4.52	4.52	4.35	96.222	589.943	255.814	0.104	1.98	589.787	0.205	130.468	589.787	130.468	80.01	1
3	3	2398	4.487	4.487	4.193	93.453	571.551	259.827	0.105	2.04	571.374	0.214	127.352	571.374	127.352	80.011	1
3	1	2398	4.502	4.502	4.365	96.853	589.937	254.196	0.104	1.94	589.763	0.202	131.003	589.763	131.003	80.01	1
3	2	2398	4.483	4.483	4.138	92.33	571.556	263.864	0.106	2.07	571.38	0.22	127.445	571.38	127.445	80.011	1
3	0	2398	4.506	4.506	3.927	87.154	590.044	260.852	0.116	2.221	589.856	0.257	130.901	589.856	130.901	80.012	1

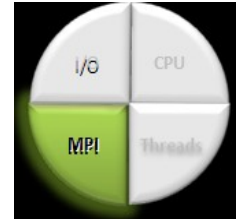
300 CONTINUE
call f_hpmstop(30+omp_get_thread_num())

MPI Profiling

Profiles MPI calls using the PMPI interface

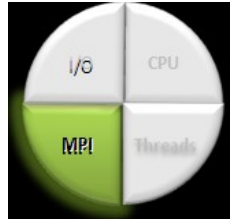
Start / stop timer

PMPI equivalent function



MPI Function	#Calls	Message Size	#Bytes	Walltime	MPI Function	#Calls	Message Size	#Bytes	Walltime
MPI_Comm_size	1 (1)	0 ... 4	0	1E-07	MPI_Irecv	2 (1)	0 ... 4	3	4.7E-06
MPI_Comm_rank	1 (1)	0 ... 4	0	1E-07	MPI_Irecv	2 (2)	5 ... 16	12	1.4E-06
MPI_Isend	2 (1)	0 ... 4	3	0.000006	MPI_Irecv	2 (3)	17 ... 64	48	1.5E-06
MPI_Isend	2 (2)	5 ... 16	12	1.4E-06	MPI_Irecv	2 (4)	65 ... 256	192	2.4E-06
MPI_Isend	2 (3)	17 ... 64	48	1.3E-06	MPI_Irecv	2 (5)	257 ... 1K	768	2.6E-06
MPI_Isend	2 (4)	65 ... 256	192	1.3E-06	MPI_Irecv	2 (6)	1K ... 4K	3072	3.4E-06
MPI_Isend	2 (5)	257 ... 1K	768	1.3E-06	MPI_Irecv	2 (7)	4K ... 16K	12288	7.1E-06
MPI_Isend	2 (6)	1K ... 4K	3072	1.3E-06	MPI_Irecv	2 (8)	16K ... 64K	49152	2.23E-05
MPI_Isend	2 (7)	4K ... 16K	12288	1.3E-06	MPI_Irecv	2 (9)	64K ... 256K	196608	9.98E-05
MPI_Isend	2 (8)	16K ... 64K	49152	1.3E-06	MPI_Irecv	2 (A)	256K ... 1M	786432	0.00039
MPI_Isend	2 (9)	64K ... 256K	196608	1.7E-06	MPI_Irecv	1 (B)	1M ... 4M	1048576	0.000517
MPI_Isend	2 (A)	256K ... 1M	786432	1.7E-06	MPI_Waitall	21 (1)	0 ... 4	0	1.98E-05
MPI_Isend	1 (B)	1M ... 4M	1048576	9E-07	MPI_Barrier	5 (1)	0 ... 4	0	7.8E-06

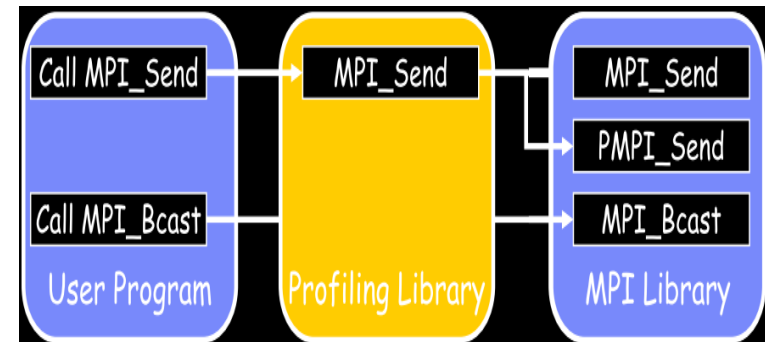
MPI Profiling



MPI Routine	#calls	avg. bytes	time(sec)
MPI_Comm_size	3	0.0	0.000
MPI_Comm_rank	12994	0.0	0.016
MPI_Send	19575	11166.9	13.490
MPI_Isend	910791	5804.2	9.216
MPI_Recv	138173	2767.9	73.835
MPI_Irecv	784936	15891.6	2.407
MPI_Sendrecv	894809	352.0	88.705
MPI_Wait	1537375	0.0	288.049
MPI_Waitall	44042	0.0	25.312
MPI_Bcast	464	41936.8	3.272
MPI_Barrier	1312	0.0	34.206
MPI_Gather	68	16399.1	2.680
MPI_Scatter	6	17237.3	0.532

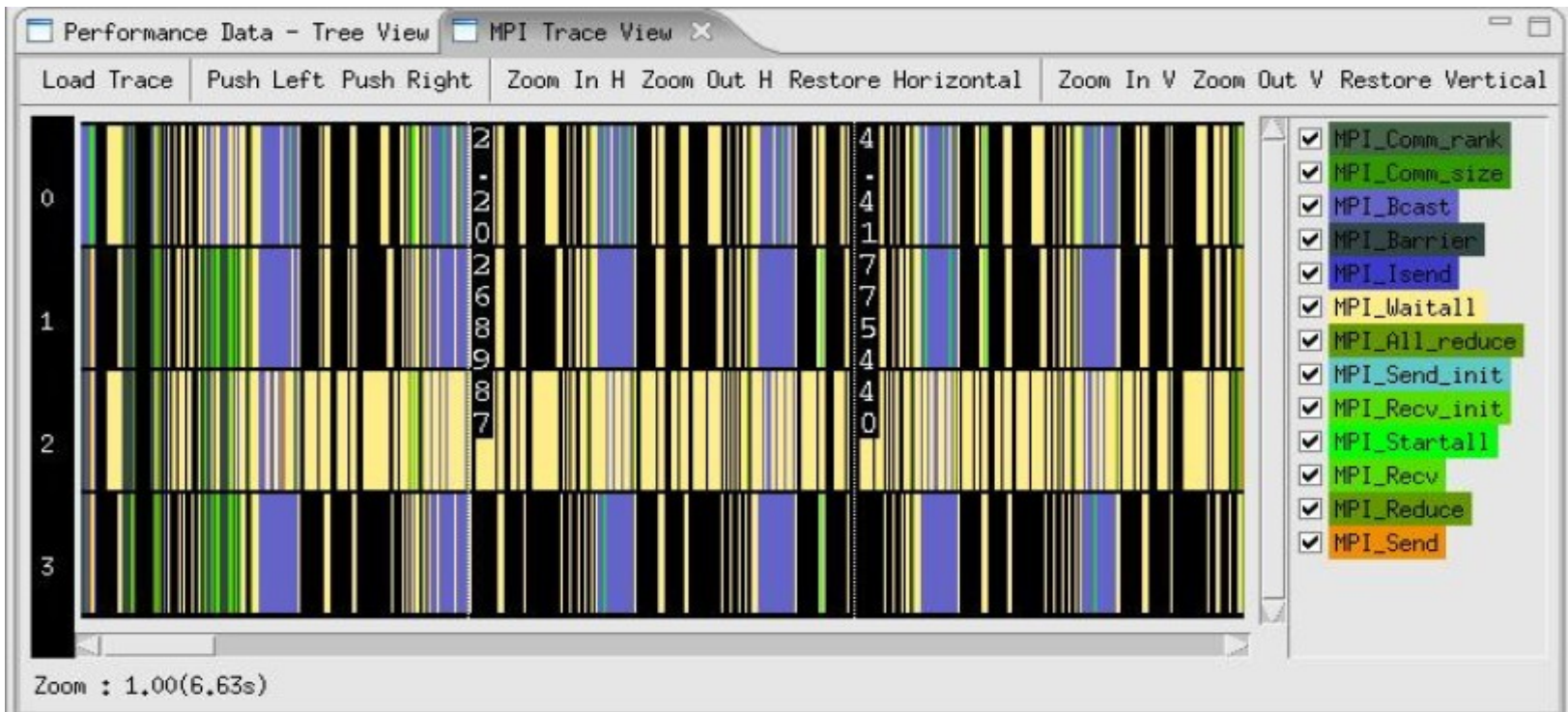
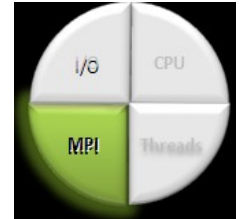
```
total communication time = 770.424 seconds.
total elapsed time       = 1168.662 seconds.
user cpu time            = 1160.960 seconds.
system time              = 0.620 seconds.
maximum memory size     = 68364 KBytes.
```

```
To check load balance : grep "total comm" mpi_profile.*
```

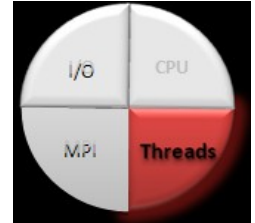


MPI Trace

- Captures **timestamped data for MPI calls**
- Provides a colour-coded trace of execution
- Provides **source traceback**
- Useful in identifying load-balancing issues



OpenMP profiling



Uses the POMP OpenMP Monitoring Interface

Groups of events identified for profiling:

- OpenMP constructs and directives/pragmas
- OpenMP API calls
- User functions and regions

Generates profile describing overheads and time spent by each thread in:

- Parallel regions
- OpenMP loops inside a parallel region
- User defined functions

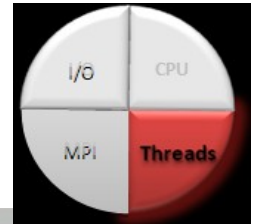
Profile data is presented in the form of an XML file => can be visualized with visualization utilities

```

1:  int main() {
2:      int id;
***  POMP_Init();
3:
***  { POMP_handle_t pomp_hdl = 0;
***      int32 pomp_tid = omp_get_thread_num();
***      POMP_Parallel_enter(&pomp_hdl, pomp_tid, -1, 1,
***          "49*type=preion*file=demo.c*slines=4,4*elines=8,8**");
4:      #pragma omp parallel private(id)
5:      {
***      int32 pomp_tid = omp_get_thread_num();
***      POMP_Parallel_begin(pomp_hdl, pomp_tid);
6:      id = omp_get_thread_num();
7:      printf("hello from %d\n", id);
***      POMP_Parallel_end(pomp_hdl, pomp_tid);
8:      }
***      POMP_Parallel_exit(pomp_hdl, pomp_tid);
***      }
***  POMP_Finalize();
9:  }

```

OpenMP profiling



peekperf

File Tools

main.f

Label	Count	Excl. Time	Incl. Time	%Total Overhead	%Imbalance	A
-pregion_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	0.2
-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_rdparam_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

main.f | runhyd3.f

```

& 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

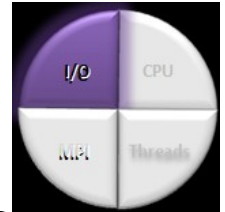
call bdrys1br(dddo, nz, nx, ny, 5,
& msg_zm, msg_zp, msg_xm, msg_xp, msg_ym, msg_yp,
& msg_mnzbdy, msg_mnxbdy, msg_mnybdy,
& msg_mxzbdy, msg_mxxbdy, msg_mxybdy)
call bdry2o1s(dddo, nz, nx, ny, 5,
& msg_zm, msg_zp, msg_xm, msg_xp, msg_ym, msg_yp,
& msg_mnybdy,
& msg_mxybdy)
xp, msg_ym, msg_yp,
& msg_mnybdy,
& 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,

```

Metric Browser: loop_1125

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

Modular I/O



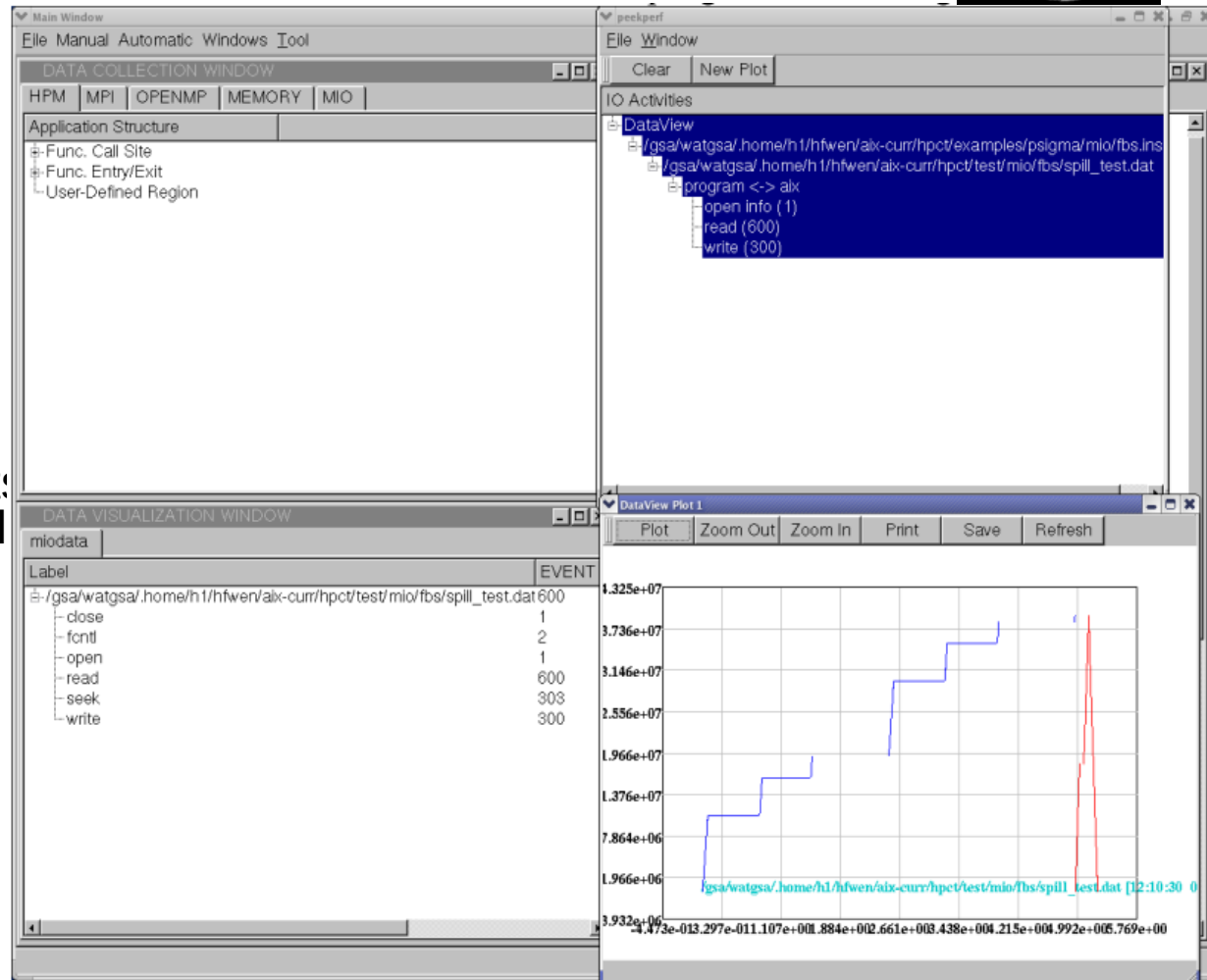
Application level optimization for I/O

Facilitates *analysis* and *tuning* of the I/O at the **application level**

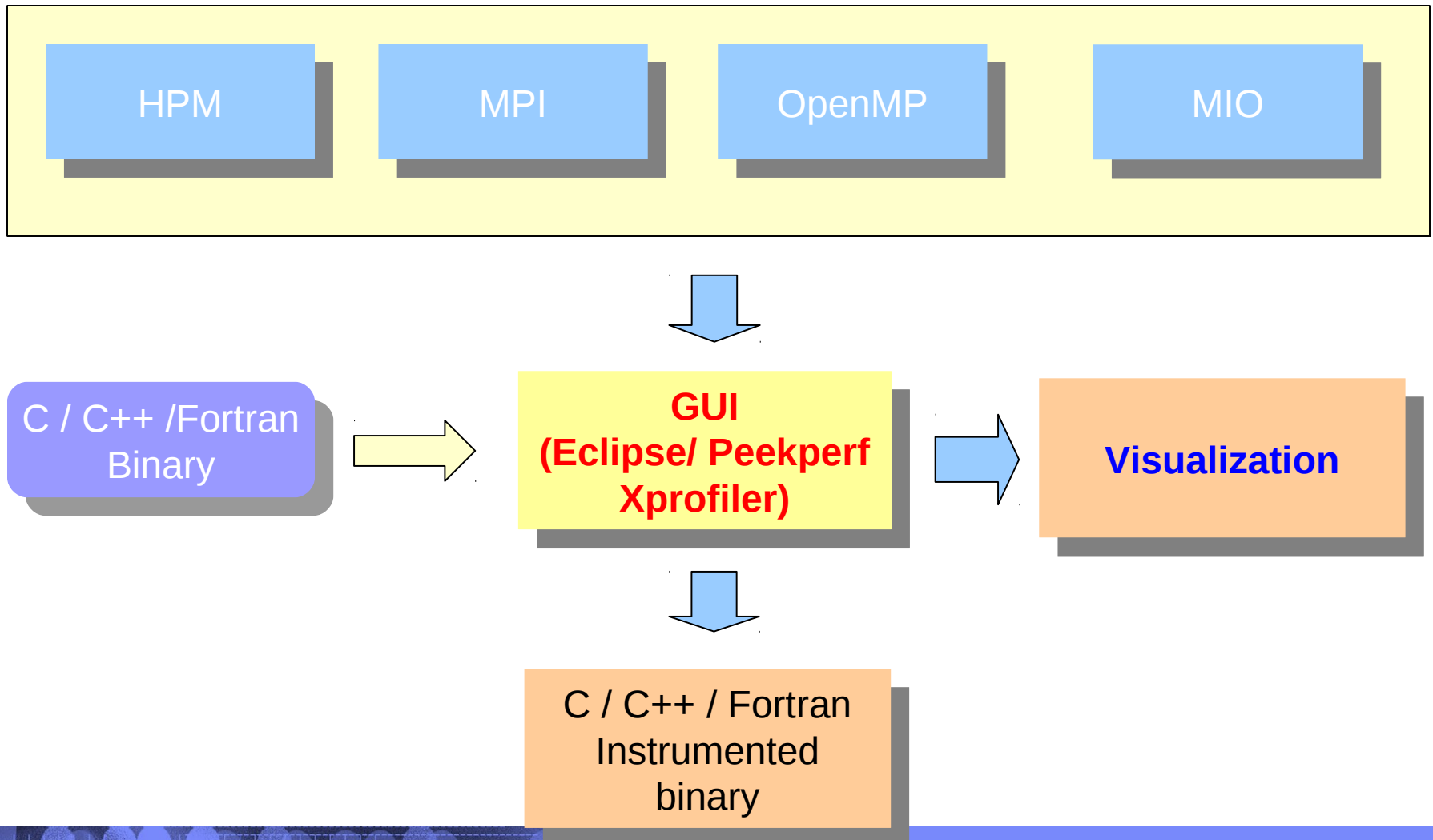
When an application exhibits the I/O pattern of sequential reading of large files, MIO

Detects the behavior

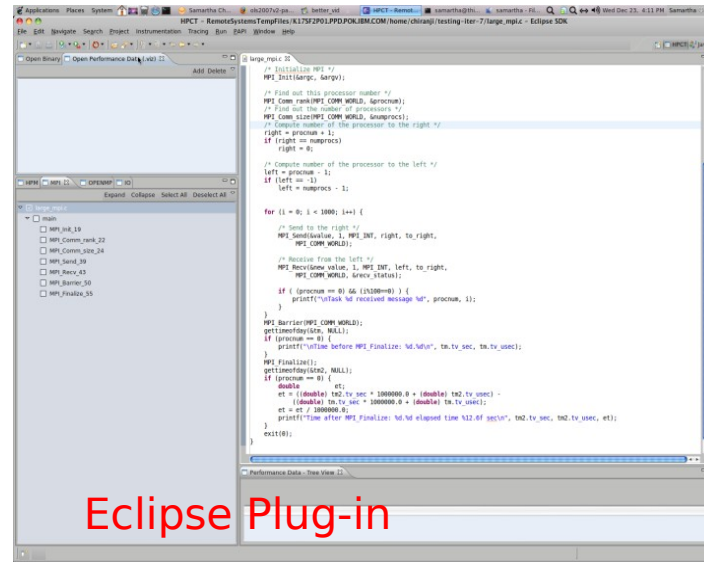
Provides *source code* *traceback*



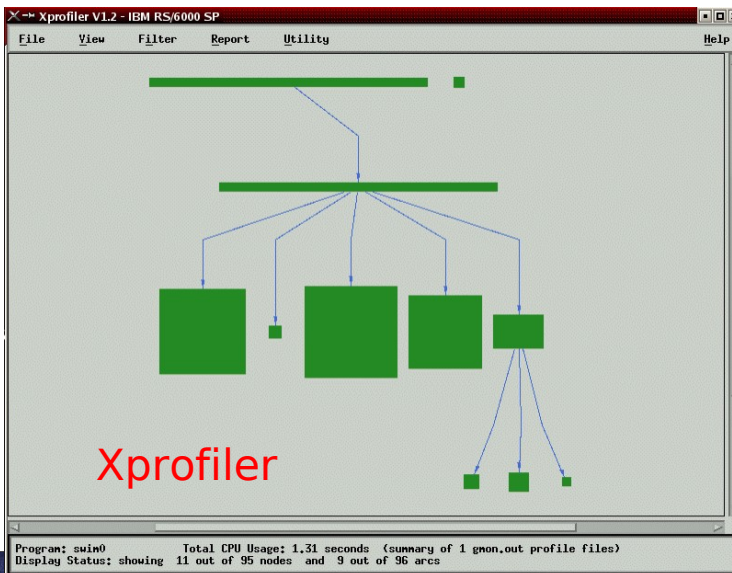
IBM HPC Toolkit – Work flow



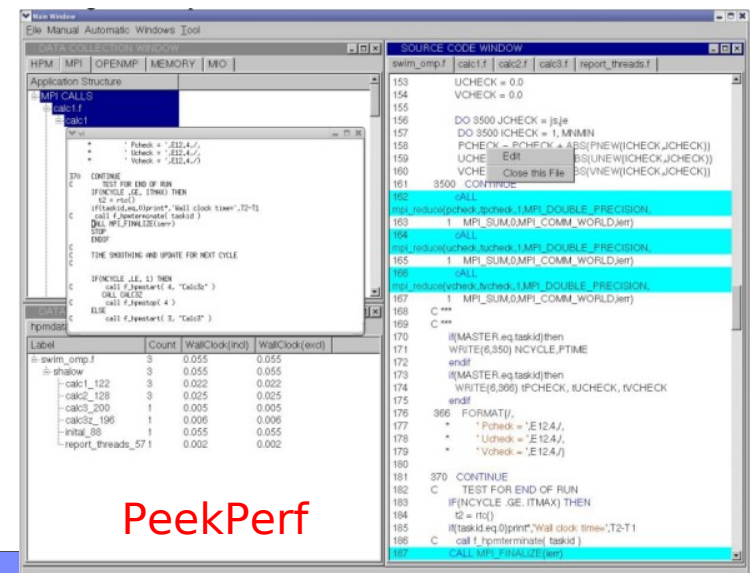
Visualization



Eclipse Plug-in



Xprofiler



PeekPerf

PeekPerf

Control centre of
the HPC Toolkit

Standalone
executable

The screenshot displays the PeekPerf application interface, which is used for performance analysis of MPI applications. It consists of several main components:

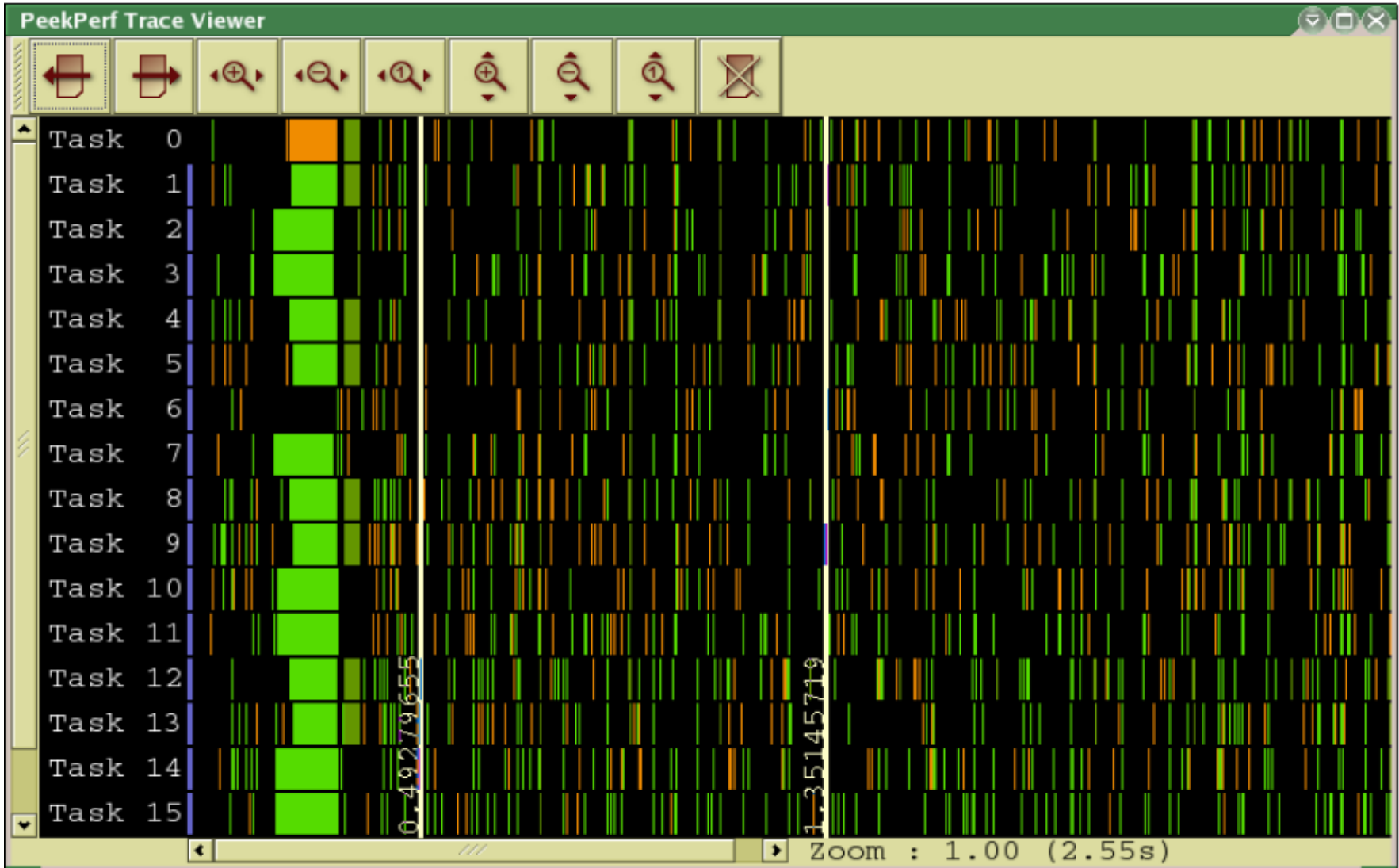
- Main Window:** Contains a menu bar (File, Manual, Automatic, Windows, Tool) and a toolbar.
- DATA COLLECTION WINDOW:**
 - Application Structure:** A tree view showing the hierarchy of MPI calls. The selected path is `calc1.f` > `calc1`. Below it, a list of MPI operations is shown, including `mpi_irecv` (ranging from 100 to 116) and `mpi_lsend` (ranging from 110 to 112).
 - HPM | MPI | OPENMP | MEMORY | MIO:** A set of tabs for different data collection methods.
- SOURCE CODE WINDOW:** Displays the source code of the application. The code includes MPI-related operations and performance checks. Key lines include:


```

153 UCHECK = 0.0
154 VCHECK = 0.0
155
156 DO 3500 JCHECK = jsje
157 DO 3500 ICHECK = 1, MNMIN
158 PCHECK = PCHECK + ABS(PNEW(ICHECK,JCHECK))
159 UCHECK = UCHECK + ABS(UNEW(ICHECK,JCHECK))
160 VCHECK = VCHECK + ABS(VNEW(ICHECK,JCHECK))
161 3500 CONTINUE
162 cALL
163 1 MPI_SUM,0,MPI_COMM_WORLD,ierr)
164 cALL
165 1 MPI_SUM,0,MPI_COMM_WORLD,ierr)
166 cALL
167 1 MPI_SUM,0,MPI_COMM_WORLD,ierr)
168 C ***
169 C ***
170 if(MASTER.eq.taskid)then
171 WRITE(6,350) NCYCLE,PTIME
172 endif
173 if(MASTER.eq.taskid)then
174 WRITE(6,366) tPCHECK, tUCHECK, tVCHECK
175 endif
176 366 FORMAT(/,
177 * 'Pcheck = ',E12.4/,
178 * 'Ucheck = ',E12.4/,
179 * 'Vcheck = ',E12.4/)
180
181 370 CONTINUE
182 C TEST FOR END OF RUN
183 IF(NCYCLE .GE. ITMAX) THEN
184 t2 = rtc()
185 if(taskid.eq.0)print*,Wall clock time=,T2-T1
186 C call f_hpmterminate( taskid )
187 CALL MPI_FINALIZE(ierr)
      
```
- PeekPerf Trace Viewer:**
 - Task View:** A Gantt chart showing the execution of tasks 0 through 15. The tasks are represented by horizontal bars, with some bars showing sub-tasks.
 - Identifier List:** A list of MPI operations with their corresponding wall clock times (excl):

Identifier	WallClock(excl)
<input type="checkbox"/> MPI_Comm_size	0.055
<input checked="" type="checkbox"/> MPI_Comm_rank	0.022
<input checked="" type="checkbox"/> MPI_Bcast	0.025
<input checked="" type="checkbox"/> MPI_Barrier	0.005
<input checked="" type="checkbox"/> MPI_Recv	0.006
<input checked="" type="checkbox"/> MPI_Send	0.055
<input checked="" type="checkbox"/> MPI_All_reduce	0.002

PeekPerf



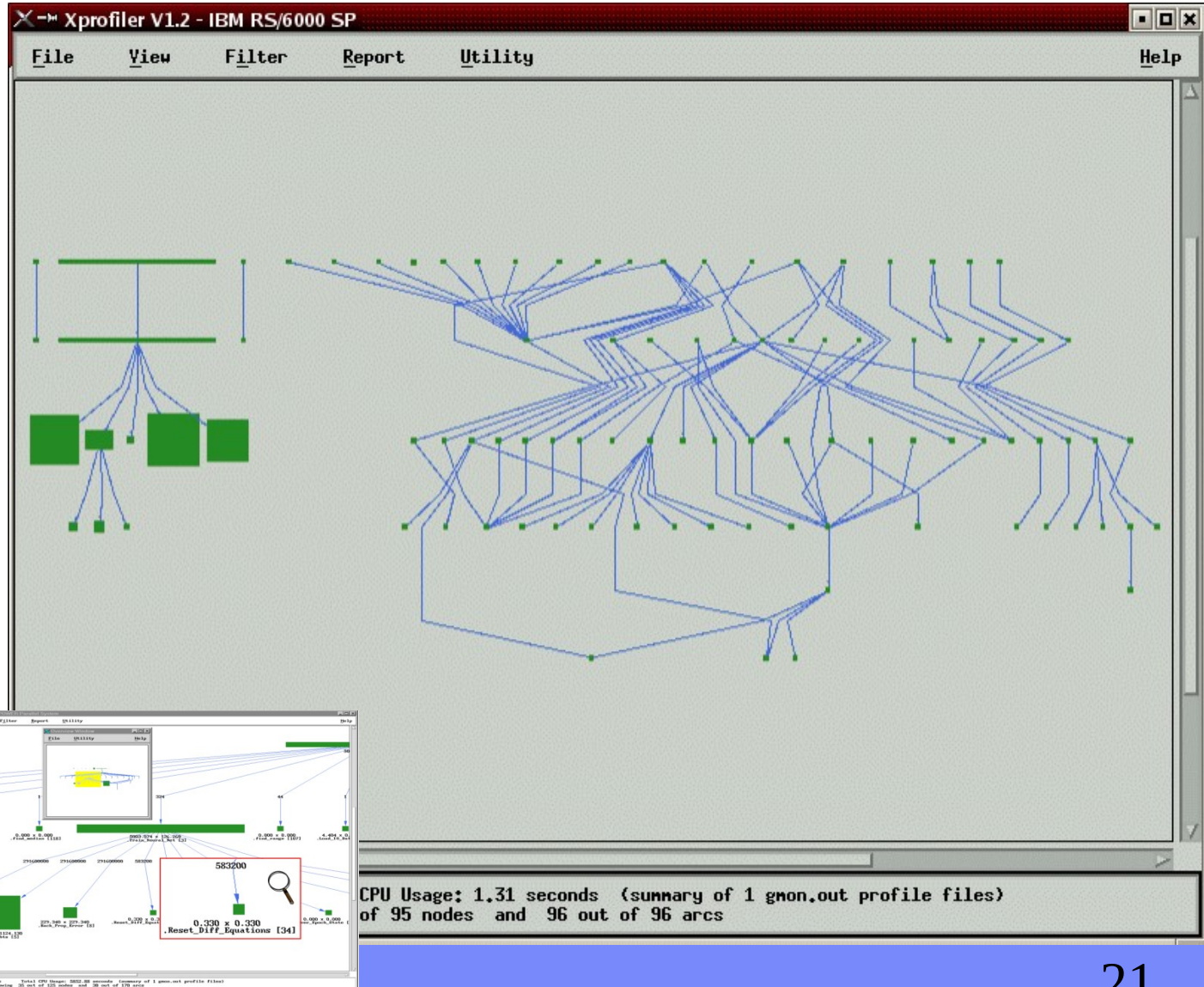
Xprofiler

CPU time profiling data

Visualization of profiled data

Profile description :

- Flat Profile
- Call Graph
- Profile Function
- Index Function
- Call Summary
- Library
- Statistics



Xprofiler

Disassembler Code for .calc3 [3]

File	no. ticks per instr.	instruction	assembler code	source code
	81	FCC4287C	fnms 6, 4, 1, 5	
	64	CCF70008	lfd 7, 0x8(23)	POLD(I, J) = P(I, J)+ALPHA*(PNEW(I, J)-
	187	C90C0008	lfd 8, 0x8(12)	
	53	C9750008	lfd 11, 0x8(21)	UOLD(I, J) = U(I, J)+ALPHA*(UNEW(I, J)-
	89	FD63582A	fa 11, 3, 11	
	63	FD28387C	fnms 9, 8, 1, 7	POLD(I, J) = P(I, J)+ALPHA*(PNEW(I, J)-
	4	DD5B0008	stfdu 10, 0x8(27)	U(I, J) = UNEW(I, J)
		C9540008	lfd 10, 0x8(20)	VOLD(I, J) = V(I, J)+ALPHA*(VNEW(I, J)-
	113	FCCA302A	fa 6, 10, 6	
	27	C8760008	lfd 3, 0x8(22)	POLD(I, J) = P(I, J)+ALPHA*(PNEW(I, J)-
	87	FD8012FA	fma 12, 0, 11, 2	UOLD(I, J) = U(I, J)+ALPHA*(UNEW(I, J)-
		0008	stfdu 5, 0x8(25)	V(I, J) = VNEW(I, J)
		482A	fa 3, 3, 9	POLD(I, J) = P(I, J)+ALPHA*(PNEW(I, J)-
		0008	lfd 10, 0x8(26)	UOLD(I, J) = U(I, J)+ALPHA*(UNEW(I, J)-
		21BA	fma 6, 0, 6, 4	VOLD(I, J) = V(I, J)+ALPHA*(VNEW(I, J)-
		0008	lfd 2, 0x8(27)	UOLD(I, J) = U(I, J)+ALPHA*(UNEW(I, J)-
		0008	stfdu 7, 0x8(12)	P(I, J) = PNEW(I, J)
		40FA	fma 8, 0, 3, 8	POLD(I, J) = P(I, J)+ALPHA*(PNEW(I, J)-
		0008	lfd 4, 0x8(25)	VOLD(I, J) = V(I, J)+ALPHA*(VNEW(I, J)-
		0008	stfdu 6, 0x8(20)	UOLD(I, J) = U(I, J)+ALPHA*(UNEW(I, J)-
		507C	fnms 3, 2, 1, 10	

Source Code for midsqmm.c

line	no. ticks per line	source code
202		/*-----*/
203		/* use 2x-unrolling of the outer two loops */
204		/*-----*/
205	4	for (i=i0; i<i0+is-1; i+=2)
206		{
207	8	for (j=j0; j<j0+js-1; j+=2)
208		{
209	1	t11 = c[i*ns+j];
210	5	t12 = c[i*ns+j+1];
211	5	t21 = c[(i+1)*ns+j];
212	19	t22 = c[(i+1)*ns+(j+1)];
213		for (k=k0; k<k0+ks; k++)
229		t21 = t21 + a[(i+1)*ns+k]*bt[j*ns+k];
218	144	t22 = t22 + a[(i+1)*ns+k]*bt[(j+1)*ns+k];
219		}
220	7	c[i*ns+j] = t11;
221	3	c[i*ns+j+1] = t12;
222	3	c[(i+1)*ns+j] = t21;
223	5	c[(i+1)*ns+(j+1)] = t22;
224		}
225		for (j=j; j<j0+js; j++)
226		{
227		t11 = c[i*ns+j];
228		t21 = c[(i+1)*ns+j];
229		for (k=k0; k<k0+ks; k++)
230		{
231		t11 = t11 + a[i*ns+k]*bt[j*ns+k];
232		t21 = t21 + a[(i+1)*ns+k]*bt[j*ns+k];
233		}
234		c[i*ns+j] = t11;
235		c[(i+1)*ns+j] = t21;
236		}
237		}

Search Engine: (regular expressions supported)

thsub

Source code window displays source code with time profile (in ticks=0.01 sec)

IBM HPC Toolkit Eclipse plug-in

HPCT - RemoteSystemsTempFiles/K17SF2P01.PPD.IBM.COM/home/chiranji/benchmark/sppm.14.7.09/main.f - Eclipse SDK

File Edit Navigate Search Project Instrumentation Tracing Run PAPI Window Help

Open Binary Open Performance Data (.viz) Add Delete

/home/chiranji/benchmark/sppm.14.7.09/run/sppm.aix

MPI IO OPENMP HPM Expand Collapse Select All Deselect All

Function Body

- runhyd3.f
 - calchydz
 - calchydy
 - calchydx
 - runhyd
- c_io.c
 - binwrite
 - changedir
 - binread
 - mvnam_khw
 - imread
 - rsclose
 - rsopen
 - brickwrite
- bdrys.f
 - zbdrys

```

! *****
integer*4 MPI_VERSION,MPI_SUBVERSION
parameter (MPI_VERSION=1,MPI_SUBVERSION=2)

integer*4 MPI_SUCCESS,MPI_ERR_BUFFER,MPI_ERR_COU
integer*4 MPI_ERR_TAG,MPI_ERR_COMM,MPI_ERR_RANK,
integer*4 MPI_ERR_ROOT,MPI_ERR_GROUP,MPI_ERR_OP,
integer*4 MPI_ERR_DIMS,MPI_ERR_ARG,MPI_ERR_UNKNO
integer*4 MPI_ERR_TRUNCATE
integer*4 MPI_ERR_OTHER,MPI_ERR_INTERN,MPI_ERR_I
integer*4 MPI_PENDING,MPI_ERR_PENDING,MPI_ERR_IN
integer*4 MPI_ERR_INFO_VALUE,MPI_ERR_INFO_NOKEY,
integer*4 MPI_ERR_FILE,MPI_ERR_NOT_SAME,MPI_ERR_
integer*4 MPI_ERR_UNSUPPORTED_DATAREP
integer*4 MPI_ERR_UNSUPPORTED_OPERATION
integer*4 MPI_ERR_NO_SUCH_FILE,MPI_ERR_FILE_EXIS

```

Performanc Performanc Progress MPI Trace Vi

Load Trace Push Left Push Right Zoom In H Zoom Out H

Zoom : 1.00(4.01s)

- MPI_Comm_ra
- MPI_Comm_si
- MPI_Send
- MPI_Recv
- MPI_Barrier

IBM HPC Toolkit Eclipse plug-in

HPCT - RemoteSystemsTempFiles/K175F2P01.PPD.IBM.COM/home/chiranji/testing-iter-7/large_mpi.c - Eclipse SDK

File Edit Navigate Search Project Instrumentation Tracing Run PAPI Window Help

Open Binary Open Performance Data (.viz) Add Delete

/home/chiranji/testing-iter-7/mpi_profile_0_1.viz

HPM MPI OPENMP IO

Expand Collapse Select All Deselect All

large_mpi.c

- main
 - MPI_Init_19
 - MPI_Comm_rank_22
 - MPI_Comm_size_24
 - MPI_Send_39
 - MPI_Recv_43
 - MPI_Barrier_50
 - MPI_Finalize_55

```

/* Initialize MPI */
MPI_Init(&argc, &argv);

/* Find out this processor number */
MPI_Comm_rank(MPI_COMM_WORLD, &procnum);
/* Find out the number of processors */
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
/* Compute number of the processor to the right */
right = procnum + 1;
if (right == numprocs)
    right = 0;

/* Compute number of the processor to the left */
left = procnum - 1;
if (left == -1)
    left = numprocs - 1;

```

Performance Data - Tree View

data below is for rank 1
aggregation for tasks: 1

Label	Count	WallClock	Transferred Bytes
▼ SUMMARY			
MPI_Recv	1000	3.997892	4000.000000
MPI_Comm_rank	1	0.000003	0.000000
MPI_Comm_size	1	0.000002	0.000000
MPI_Barrier	1	0.001098	0.000000
MPI_Send	1000	0.012439	4000.000000
▼ large_mpi.c			
▼ main(large_mpi.c)			
MPI_Barrier_50	1	0.001098	0.000000
MPI_Comm_size_24	1	0.000002	0.000000
MPI_Comm_rank_22	1	0.000003	0.000000
MPI_Recv_43	1000	3.997892	4000.000000
MPI_Send_39	1000	0.012439	4000.000000

IBM HPC Toolkit Eclipse plug-in

Performance Data - Tree View | Performance Data - Table View | Progress | MPI Trace View

data below is for rank 0
aggregation for tasks: 0

Label	Count	WallClock	Transferred Bytes
main.f			
layout(main.f)			
MPI_Comm_size_3129	1	0.000002	0.000000
MPI_Comm_rank_3128	1	0.000005	0.000000
main(main.f)			
MPI_Comm_size_450	1	0.000002	0.000000
MPI_Comm_rank_449	1	0.000034	0.000000
MPI_Barrier_749	1	0.050146	0.000000
gblbldsum(main.f)			
MPI_Allreduce_2259	11	0.194295	264.000000
gblbmax(main.f)			
MPI_Allreduce_1364	10	0.119984	80.000000
SUMMARY			
MPI_Isend	240	0.012629	9753600.000000
MPI_Recv	100	0.000000	0.000000

IBM HPC Toolkit Eclipse plug-in

```

call deltat(" Finished Z sweep",2)

call zbdrys( ddd, 1 )
call deltat(" Finished Z bdrys",2)
call calchydz( 1 )
call deltat(" Finished Z sweep",2)

call ybdrys( ddd, 1 )
call deltat(" Finished Y bdrys",2)

```

Performance Data - Tree View | MIO Tree View | MIO Trace View | DataView Table

Color: open activity | 1 | None

start	finish	next open pos	name	skip
..3095E-02	1.3096E-02	3567	mio5.txt [06:45:08 08-07-2009]	0

MPI Trace View

Load Trace Push Left Push Right Zoom In H Zoom Out H Restore Horizontal Zoom In V Zoom Out V

- MPI_Comm_siz
- MPI_Comm_ra
- MPI_Ialltoallv
- MPI_Win_free
- MPI_Isscatterv

Zoom : 1.00(1.26ms)

Select a color for this event type

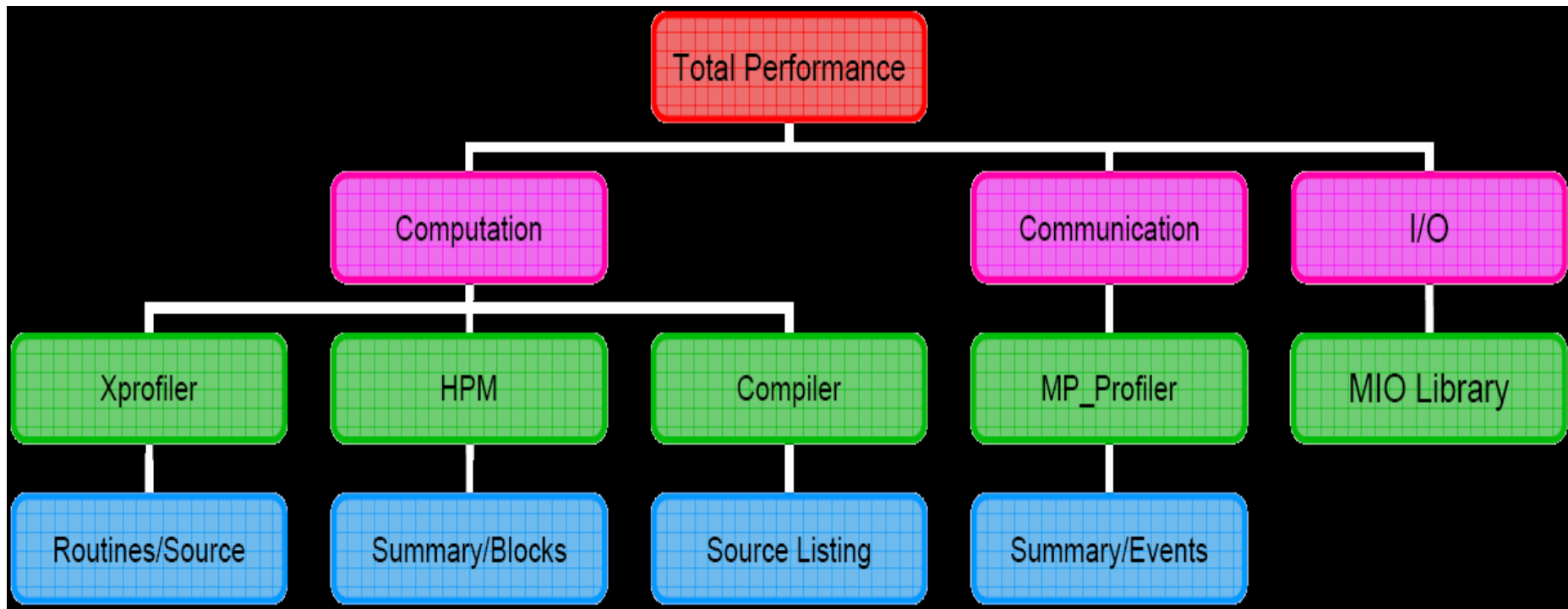
Hue: 0 Red: 240
 Saturation: 0 Green: 240
 Value: 94 Blue: 240
 Color name: #F0F0F0

Palette: [Color swatches]

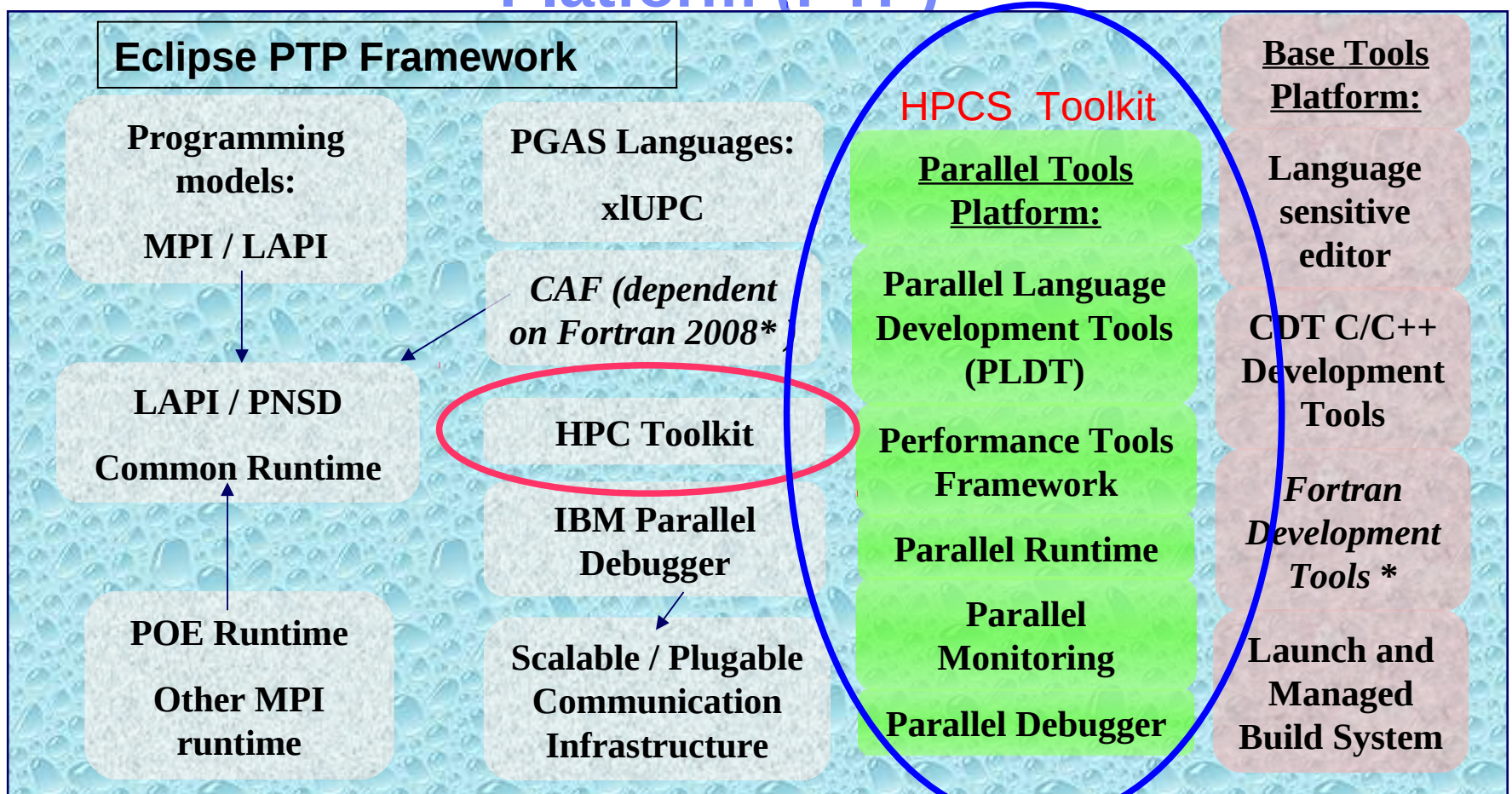
Cancel OK

Visualization of IBM HPCT in Eclipse Plug-in

IBM HPC Toolkit in a nutshell



Direction of Migration – Eclipse Parallel Tools Platform (PTP)



Power Clusters:
P5, P6, P7

x86 Clusters:
Intel/AMD

Power Clusters:
BlueGene

Hybrid Clusters:
Accelerator

Summary

- The IBM HPC Toolkit provides an **integrated framework** for performance analysis
- Support **iterative analysis** and automation of the performance tuning process
- The standardized software layers make it easy to **plug in** new performance analysis tools
- Operates on the binary and yet provide reports in terms of **source-level symbols**
- Provides **multiple layers** that the user can exploit (from low-level instrumentations to high-level performance analysis) – for **beginners** to **black-belts**
- Full **source code traceback** capability
- **Dynamically** activate/deactivate data collection and change what information to collect

Thank You