

Performance HPC Linux  
Bull Echirolles

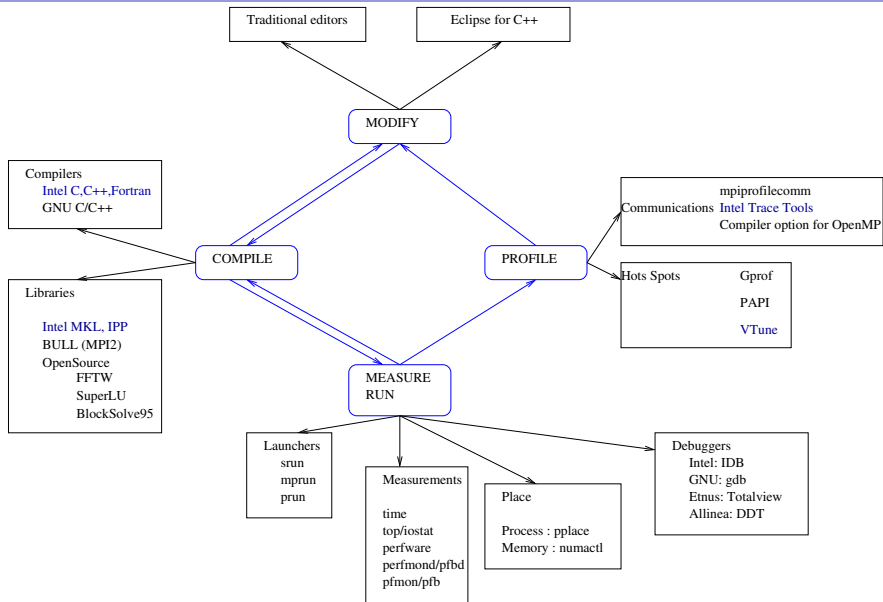
# Software Development Environment

## Part Three

C.Berthelot  
Christophe.Berthelot@bull.net

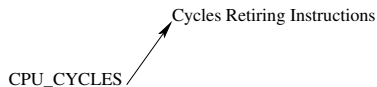
Copyright (©) Bull S.A.S. 2008





- Bubble Analysis
- Latency Analysis
- First dimension: The node
- Second dimension The interconnect
- Questions

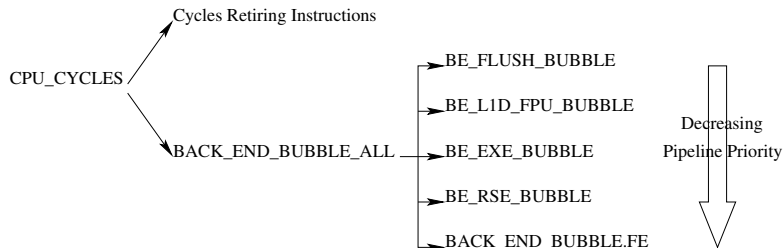
## Starting the Tree Structure Analysis



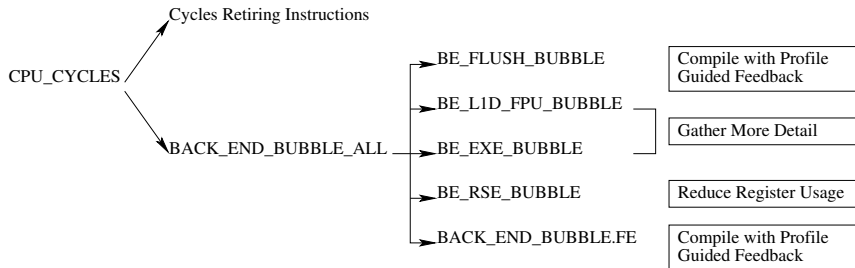
BACK\_END\_BUBBLE\_ALL are the stall cycles to be reduced.

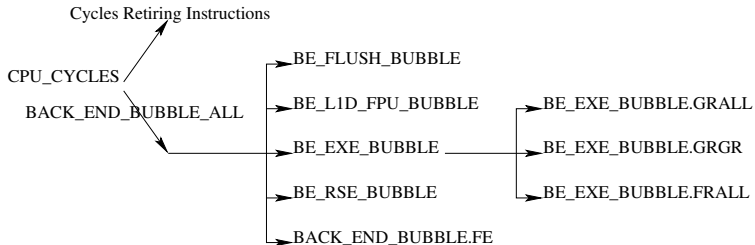
Eliminating the major contributions to this number is the most promising way for optimization

## Stall Cycles Have 5 Components



$$\begin{aligned} \textit{BACK\_END\_BUBBLE} &= \textit{BE\_FLUSH\_BUBBLE} \\ &+ \textit{BE\_L1D\_FPU\_BUBBLE} \\ &+ \textit{BE\_EXE\_BUBBLE} \\ &+ \textit{BE\_RSE\_BUBBLE} \\ &+ \textit{BACK\_END\_BUBBLE.FE} \end{aligned}$$





- ▶ BE\_EXE\_BUBBLE.GRALL Counts all stall cycles due to waiting for valid data delivery to general register
- ▶ BE\_EXE\_BUBBLE.GRGR Counts all stall cycles due to waiting for valid data delivery to general register from an integer instruction
- ▶ BE\_EXE\_BUBBLE.GRALL-BE\_EXE\_BUBBLE.GRGR Approximates memory access stall for integer data
- ▶ BE\_EXE\_BUBBLE.FRALL Counts all stall cycles due to waiting for valid data delivery to an FP register

# Long Latency Loads

## L3 cache misses

L3 cache misses due to loads (and stores) cause hundreds of stalled cycles.  
L3\_reads.data\_reads.miss is caused by loads, stores and prefetch instructions.

## Data EAR events

- ▶ Allow user to select minimum latency (DATA\_EAR\_CACHE\_LAT 8,16,32,64,...cycles)
- ▶ Minimum latency  $> 32/64$  identifies unprefetched data loads



# The Big four

## Events

- ▶ CPU\_CYCLES (SAV<sup>a</sup>=1000000)
- ▶ BACK\_END\_BUBBBLE.ALL (SAV=1000000)
- ▶ DEAR\_LATENCY\_GT\_64 (SAV=1000)
- ▶ BUS\_MEMORY.ALL.SELF (SAV=10000)

---

<sup>a</sup>Sample After Value

## Identifies

- ▶ Low hanging fruit with BACK\_END\_BUBBBLE.ALL
- ▶ Hotspots (cpu\_cycles/back\_end\_bubble.all)
- ▶ Long latency data access/non prefetched data/thread sharing problemes
- ▶ Bandwidth limited loops

- Bubble Analysis
- Latency Analysis
- First dimension: The node
  - Level 1: subroutines by subroutines
  - Level 2: lines by lines
  - Level 3: assembly
- Second dimension The interconnect
- Questions

## Level 1: subroutines by subroutines

### Tool

Classical open source gprof

### Advantages

- ▶ Standard
- ▶ Well known
- ▶ Supported by Intel compiler
- ▶ Easy to use
- ▶ MPI support

## Example

### Traditional tool: gprof

- ▶ Simple to use (compile with -p)
- ▶ Use GPROF\_PREFIX or GMON\_OUT\_PREFIX
- ▶ `gprof -s a.out $GMON_OUT_PREFIX.*`
- ▶ `gprof a.out gmon.sum > summary-profile`

## Level 2: lines by lines

### For tools

- ▶ Gprof
- ▶ Intel Vtune
- ▶ API PAPI
- ▶ HPCToolkit

### Advantages

- ▶ Gprof: simple, standard
- ▶ Vtune: time information and hardware (and ratio) lines by lines inside a GUI or with command line interface
- ▶ PAPI: You need use api inside your source code

## Medium Level: PAPI

PAPI aims to provide the tool designer and application engineer with a consistent interface and methodology for use of the performance counter hardware. PAPI is used for the following reasons:

- ▶ to provide a solid foundation for cross-platform performance analysis tools
- ▶ to present a set of standard definitions for performance metrics on all platforms
- ▶ to provide a standard API among users, vendors and academics.

PAPI supplies two interfaces:

- ▶ a high-level interface, for simple measures
- ▶ a low-level interface, programmable, adapting the specificity of the machine and linking the measures.

## Example: Simple use

```
#include "fpapi.h"
    real real_time,cpu_time, mflops
    integer*8 fp_ins

    call PAPIf_flops(real_time,cpu_time,fp_ins,mflops,ierr)
c
c   Code to profile
c
    call PAPIf_flops(real_time,cpu_time,fp_ins,mflops,ierr)

    write (6,120) real_time, cpu_time, fp_ins,mflops
120 format (/'PAPI result'/ 'real time (secs): ', f15.3,
$ /'CPU time (secs): ',f15.3,
$ /'floating point instructions: ', i15,
$ /'MFLOPS: ', f15.3)
```

## Usage haut-niveau

- ▶ Include header

```
#include "fpapi.h"
```

- ▶ Some declaration

```
integer events(2), numevents, ierr  
character*PAPI_MAX_STR_LEN errorstring
```

- ▶ Array for counters

```
integer*8 values(2)
```

- ▶ Counters (cf fpapi.h)

```
numevents= 2  
events(1)=PAPI_FP_INS  
events(2)=PAPI_TOT_CYC
```



► Start measurements

```
call PAPIf_start_counters(events, numevents, ierr)
if ( ierr .NE. PAPI_OK ) then
  call PAPIF_perror(ierr,errorstring,PAPI_MAX_STR_LEN)
  print *, errorstring
end if
```

► Save counters values and restart counters

```
call PAPIf_read_counters(values,numevents,ierr)
if ( ierr .NE. PAPI_OK ) then
  call PAPIF_perror(ierr,errorstring,PAPI_MAX_STR_LEN)
  print *, errorstring
end if
```

- ▶ Accumulate counters

```
PAPIf_accum_counters(values,numevents,ierr)
```

- ▶ Stop counters:

```
call PAPIf_stop_counters(values, numevents, ierr)
if ( ierr .NE. PAPI_OK ) then
  call PAPIF_perror(ierr,errorstring,PAPI_MAX_STR_LEN)
  print *,errorstring
end if
```

# PAPI with OpenMP

- ▶ Include: use `omp_lib #include <f90papi.h>`

- ▶ Initialization

```
chkflg = PAPI_VER_CURRENT
call PAPIF_library_init(chkflg)
if (chkflg .ne. PAPI_VER_CURRENT) then
    print *, 'Error initializing the PAPI Library'
    call abort
end if
```

- ▶ Threads Initialization

```
call PAPIF_thread_init(omp_get_thread_num,chkflg)
if (chkflg /= PAPI_OK) then
    print *, 'Error in subroutine PAPIF_thread_init'
    call abort
end if
```

## Example

```
[berthelc@ns17 (CONF 09) MandelOpenMP_PAPI] ./mandel
```

```
-----  
Thread      2 : MFLOPS =    1103.203  
Thread      1 : MFLOPS =    1028.133  
Thread      0 : MFLOPS =    1009.210  
Thread      6 : MFLOPS =    1120.061  
Thread      3 : MFLOPS =    1032.386  
Thread      5 : MFLOPS =    1009.772  
Thread      7 : MFLOPS =    1009.336  
Thread      4 : MFLOPS =    1009.654  
Loop seconds: 0.8052310  
Total MFLOPS:    8321.755
```

## HPCToolKit

HPCToolkit is an open-source suite of multi-platform tools for profile-based performance analysis of applications.

- ▶ hpcrun: a tool for profiling executions of unmodified application binaries using statistical sampling of hardware performance counters.
- ▶ csprof: a tool for collecting call path profiles of optimized application binaries.
- ▶ bloop: a tool for analyzing application binaries to recover program structure; namely, to identify where loops are present and what program source lines they contain.
- ▶ hpcprof: a tool for interpreting sample-based execution profiles and relating them back to program source lines.
- ▶ hpcview: a tool for correlating program structure information, multiple sample-based performance profiles, and program source code to produce a performance database.
- ▶ hpcviewer: a java-based GUI for exploring databases consisting of performance information correlated with program source.

# First Use

## Compile and Run

- ▶ Compile with -g
- ▶ run with hpcrun ./a.out with default events ( PAPI\_TOT\_CYC)
- ▶ Result inside file a.out.PAPI\_TOT\_CYC.linux0.25098.0x620a

## Analysis 1/2

- ▶ Run hpcprof ./a.out a.out.PAPI\_TOT\_CYC.linux0.25098.0x620a
- ▶ Result: ASCII or HTML

```
Columns correspond to the following events [event:period (events/sample)]  
PAPI_TOT_CYC:32767 - Total cycles (5162 samples)
```

```
Load Module Summary:
```

```
99.8% /<path_to_bin>/cg.S
```

```
0.2% /lib/ld-2.3.4.so
```

```
[...]
```

```
File Summary:
```

```
98.7% <<path_to_bin>/cg.S>><path_to_src>/cg.f
```

```
0.9% <<path_to_bin>/cg.S>><path_to_src>/randi8.f
```

```
[...]
```

## Analysis 2/2

### Function Summary:

```
85.5% <</path_to_src>/cg.S>>conj_grad
12.5% <<<path_to_bin/cg.S>>>sparse
[...]
```

File <<path\_to\_bin/cg.S>><path\_to\_src>/cg.f with profile annotations.

```
516      c-----
517      c  q = A.p
518      c  The partition submatrix-vector multiply: use workspace w
519      c-----
520      C
521      C  NOTE: this version of the multiply is actually (slightly: maybe %5)
522      C        faster on the sp2 on 16 nodes than is the unrolled-by-2 version
523      C        below.  On the Cray t3d, the reverse is true, i.e., the
524      C        unrolled-by-two version is some 10% faster.
525      C        The unrolled-by-8 version below is significantly faster
526      C        on the Cray t3d - overall speed of code is 1.5 times faster.
527      C
528      0.6%      do j=1,lastrow-firstrow+1
529                sum = 0.d0
530      29.3%      do k=rowstr(j),rowstr(j+1)-1
531      49.8%          sum = sum + a(k)*p(colidx(k))
532                enddo
533      1.1%      q(j) = sum
534                enddo
[...]
```

## Use GUI

### Run

- ▶ Analyze application binaries: `bloop ./a.out > a.out.psxml`
- ▶ Run: `hpcrun -e IA64_INST_RETIRED -e L3_MISSES -e PAPI_TOT_CYC ./a.out>profile`
- ▶ Run: `hpcquick -I<SRC_PATH> -S a.out.psxml -P profile -n`
- ▶ Run: `hpcview -o experiment-db hpcquick.xml`
- ▶ Run: `hpcviewer`



# Hpcviewer

The screenshot displays the Hpcviewer application interface. The top pane shows the source code of a file named 'cg.f'. The code includes a multi-line comment explaining performance differences between unrolled versions on different hardware (sp2 vs Cray t3d). The code contains nested loops for calculating a sum and updating an array 'q'. The bottom pane, titled 'Flat View', shows a tree of scopes on the left and a table of performance metrics on the right.

**Source Code (cg.f):**

```
519 C -----
520 C
521 C NOTE: this version of the multiply is actually (slightly. maybe %5)
522 C faster on the sp2 on 16 nodes than is the unrolled-by-2 version
523 C below. On the Cray t3d, the reverse is true, i.e., the
524 C unrolled-by-two version is some 10% faster.
525 C The unrolled-by-8 version below is significantly faster
526 C on the Cray t3d - overall speed of code is 1.5 times faster.
527 C
528 C do j=1,lastrow-firstrow+1
529 C   sum = 0.d0
530 C   do k=rowstr(j),rowstr(j+1)-1
531 C     sum = sum + a(k)*p(colidx(k))
532 C   enddo
533 C   q(j) = sum
534 C enddo
535 C
536 CC do j=1,lastrow-firstrow+1
```

**Scopes:**

- Experiment Aggregate Metrics
  - Load module cg.S
    - cg.f
      - conj\_grad
        - loop at cg.f: 528-622
        - loop at cg.f: 631-636
        - loop at cg.f: 489-493

**Performance Metrics Table:**

IA64_INST_RET...	L3_MISSES	PAPL_TOT...
6.25e08 100.0		1.77e08 100.0
6.25e08 100.0		1.77e08 99.9%
6.24e08 99.7%		1.75e08 98.9%
5.54e08 88.5%		1.52e08 85.8%
5.32e08 85.1%		1.46e08 82.5%
2.08e07 3.3%		5.57e06 3.1%
1.64e05 0.0%		6.55e04 0.0%

## Level 3: assembly

### Vtune

Find performance bottlenecks with advanced profiling technologies:

- ▶ Event-Based, System-Wide Sampling with little impact on program execution (typically  $< 1\%$ ).
- ▶ Call Graph Profiling offers a pictorial view of program flow to help you quickly identify critical functions.



# Sampling

*File->New->Project...*

Select a wizard



Wizards:

- ▶ C
- ▶ C++
- ▶ CVS
- ▶ Simple
- ▶ Tuning Activity



< Back

Next >

Finish

Cancel

Select a wizard



Create an Activity with the sampling data collector. VTune analyzer will collect and display system-wide software performance data.

Wizards:

- ▶ C
- ▶ C++
- ▶ CVS
- ▶ Simple
- ▼ Tuning Activity
  - ▶ Call Graph Wizard
  - ▶ First Use Wizard
  - ▶ Sampling Wizard



< Back

Next >

Finish

Cancel

# Sampling

## Select application

### Environment and Activity Settings

Choose the type of application that will be profiled.

Select collection environment

☒ Linux® executable  
☐ Java®

Select application type:

Application

< Back   Next >   Finish   Cancel

### Application/Module Profile Settings

The application below will be launched by VTune(TM) Performance Analyzer during the Activity run. Fill in the application details.

Application parameters

Application to launch:  Browse...

☐ No application to launch

Optional

Application arguments:

Working directory:  Browse...

Remote collection settings

The application can be run locally, or on a remote system. [Learn more...](#)

☐ Enable remote collection for [user@]host:

Application to profile (Module of interest)

Ignore this unless you plan to use 'Pack and Go' or remote collection where you may lose the connection to the remote host. [Learn more...](#)

Change...

< Back   Next >   Finish   Cancel

# Sampling

## Customize application

### Application/Module Profile Settings

The application below will be launched by VTune(TM) Performance Analyzer during the Activity run. Fill in the application details.

#### Application parameters

Application to launch:    
☐ No application to launch

#### Optional

Application arguments:   
Working directory:

#### Remote collection settings

The application can be run locally, or on a remote system. [Learn more...](#)

☐ Enable remote collection for [user@]host:

#### Application to profile (Module of interest)

Ignore this unless you plan to use 'Pack and Go' or remote collection where you may lose the connection to the remote host. [Learn more...](#)

< Back

Next >

Finish

Cancel

### Sampling Collection Settings

You can configure some of the sampling collection parameters. Use the selected default values if you are not sure.

#### Stop collection

☐ After duration of  seconds  
☒ When the application terminates (before duration completes)

< Back

Next >

Finish

Cancel

# Sampling First windows

The screenshot displays the Vtune(TM) Performance Analyzer interface. The main window shows a table of sampling results for three events: CPU\_Cycles, IA64\_INST\_RETIRED, and FP\_OPS\_RETIRED. The table is organized into columns for Process, CPU\_Cycles samples, IA64\_INST\_RETIRED-THIS samples, and FP\_OPS\_RETIRED samples. The data is grouped by process, with 'cg.8.8' and 'java' being the primary processes shown. The 'cg.8.8' process shows a total of 704,263 CPU\_Cycles samples, 2,107,485 IA64\_INST\_RETIRED-THIS samples, and 704,263 FP\_OPS\_RETIRED samples. The 'java' process shows 819 CPU\_Cycles samples, 310 IA64\_INST\_RETIRED-THIS samples, and 356 FP\_OPS\_RETIRED samples. The 'top' process shows 325 CPU\_Cycles samples, 356 IA64\_INST\_RETIRED-THIS samples, and 356 FP\_OPS\_RETIRED samples.

Process	CPU_CYCLES samples	IA64_INST_RETIRED-THIS samples	FP_OPS_RETIRED samples
pk6_0n0	704,263	2,107,485	
cg.8.8	29,300	27,123	7
cg.8.8	29,177	26,077	7
cg.8.8	29,002	26,586	7
cg.8.8	28,984	26,395	7
cg.8.8	28,848	25,750	7
cg.8.8	28,701	26,265	7
cg.8.8	28,530	25,842	7
cg.8.8	28,485	26,055	7
java	819	310	
top	325	356	

The interface also includes a 'Related Topics' section on the left, a 'Navigator' pane showing the project structure, and a 'Console' pane at the bottom displaying the sampling process logs. The logs show the calibration of the sampling events, the setting of the CPU mask to 0-15, and the collection of sampling data.

# Sampling First windows

## Commands line

- ▶ The default project file

```
>/opt/intel/vtune/bin/vtl query -project  
Project file: /tmp/vtune_berthelot/Projects/vtldefault.vpj
```

- ▶ Import data base inside

```
>/opt/intel/vtune/bin/vtl import tbs7b16.tb5  
File tbs7b16.tb5 successfully imported...Use "vtl view <activity result name>"  
to view results
```

- ▶ Look activity name

```
>/opt/intel/vtune/bin/vtl show  
f1__ImportedResults  
r1__Imported Sampling Results
```

## Command line view processes

vtl view f1::r1 -processes

Process ID	CPU_CYCLES	IA64_INST_RETIRED-THIS	FP_OPS_RETIRED	IA64_IPC
pid_0x0	704263	2107485	43	2.992 [...]
cg.B.8	29300	27123	76274	0.926 [...]
cg.B.8	29177	26077	75207	0.894 [...]
cg.B.8	29002	26586	74955	0.917 [...]
cg.B.8	28984	26395	75421	0.911 [...]
cg.B.8	28848	25750	75686	0.893 [...]
cg.B.8	28701	26265	76132	0.915 [...]
cg.B.8	28530	25842	75532	0.906 [...]
cg.B.8	28485	26055	75955	0.915 [...]
java	819	310	88	0.379 [...]
top	325	356	529	1.095 [...]
pid_0x43	234	127	7	0.543 [...]



## File Edit Refactor Navigate Search Pr

The screenshot displays the Visual Studio Performance Profiler interface. On the left, the 'Tuning Browser' shows the project hierarchy: 'Project1' containing 'CG\_8', which is running a sample on 'Mon Apr 16 15:54:22 2007'. The 'Run 1' session is selected, showing a list of activities: 'CPU\_CYCLES', 'IA64\_INST\_RETIRED-THIS', and 'FP\_OPS\_RETIRED'. The 'Related Topics' pane on the far left includes 'About Sampling Thread View'. The main window shows a table of thread data for 'thread224' in process 'cg\_8.8'. The table has columns for Thread, Process, CPU\_CYCLES samples, IA64\_INST\_RETIRED-THIS samples, and FP\_OPS\_RETIRED. The data row shows 29,300 CPU cycles, 27,123 retired instructions, and 76... FP ops. Below this, a secondary table shows 'Activity ID' (10), 'Activity Result' (Mon Apr 16 15:54:22 2007 - Sampling Results [ns17]), 'Total Samples' (3863100), 'Duration' (60.45 ns17), 'Machine N' (0), 'CPU ID' (0), 'HW Package' (0), and 'Idle time' (49.83%). The bottom of the interface shows tabs for 'Processes', 'Threads', and 'Modules', with 'Threads' selected. The 'Console' window is visible at the very bottom.

## File Edit Refactor Navigate Search Pr

The screenshot displays the Intel VTune Performance Analyzer interface. The left sidebar shows the 'About Sampling Module View' and a tree structure for 'Project1' with a sub-entry 'CG\_8'. The main window is titled 'Welcome' and 'Run | X'. It contains a table of sampling results with columns: Module, Process, CPU\_CYCLES samples, IA64\_INST\_RETIRED-THIS samples, and Events / Total. The table lists various modules and processes, including 'cg\_8.8', 'libelan.so.1', 'vmlinux-2.6.18-864k1.7', 'libdev.elanbul2.so', 'libpthread-2.3.4.so', 'Other64', 'elan4', 'libc-2.3.4.so', 'mdm', 'libmpibull.so.core', 'librdm.so', 'ld-2.3.4.so', 'libelan4.so.1', 'sumpc', and 'libmf.so.6'. The right sidebar shows a table of events and their totals, including 'CPU\_CYCLES %', 'CPU\_CYCLES events', 'CPU\_CYCLES samples', 'FP\_OPS\_RETIRED %', 'FP\_OPS\_RETIRED events', 'FP\_OPS\_RETIRED samples', 'IA64\_INST\_RETIRED-THIS %', 'IA64\_INST\_RETIRED-THIS events', 'IA64\_INST\_RETIRED-THIS samples', and 'IA64\_IPC'.

Module	Process	CPU_CYCLES samples	IA64_INST_RETIRED-THIS samples	Events / Total
cg_8.8	cg_8.8	26,671	24,198	CPU_CYCLES % 91.03
libelan.so.1	cg_8.8	837	1,046	CPU_CYCLES events 42...
vmlinux-2.6.18-864k1.7	cg_8.8	815	739	CPU_CYCLES samples 26...
libdev.elanbul2.so	cg_8.8	508	681	FP_OPS_RETIRED % 99.95
libpthread-2.3.4.so	cg_8.8	133	175	FP_OPS_RETIRED events 4.6...
Other64	cg_8.8	107	134	FP_OPS_RETIRED samples 76...
elan4	cg_8.8	81	11	IA64_INST_RETIRED-THIS % 89.22
libc-2.3.4.so	cg_8.8	73	120	IA64_INST_RETIRED-THIS events 38...
mdm	cg_8.8	34	8	IA64_INST_RETIRED-THIS samples 24...
libmpibull.so.core	cg_8.8	28	4	IA64_IPC 0.91
librdm.so	cg_8.8	6	0	
ld-2.3.4.so	cg_8.8	5	2	
libelan4.so.1	cg_8.8	2	3	
sumpc	cg_8.8	0	1	
libmf.so.6	cg_8.8	0	1	

## Command line view modules

vtl view fl::r1 -pid 0x6be4

Module	Process	CPU_CYCLES	IA64_INST_RETIRED-THIS	FP_OPS_RETIRED	IA64_IPC
cg.B.8	cg.B.8	26671	24198	76234	0.907
libelan.so.1	cg.B.8	837	1046	0	1.250
vmlinux-2.6.18-B64k.1.7	cg.B.8	815	739	37	0.907
libdev.elanbull2.so	cg.B.8	508	681	0	1.341
libpthread-2.3.4.so	cg.B.8	133	175	0	1.316
Other64	cg.B.8	107	134	0	1.252
elan4	cg.B.8	81	11	0	0.136
libc-2.3.4.so	cg.B.8	73	120	0	1.644
mdm	cg.B.8	34	8	0	0.235
libmpibull.so.core	cg.B.8	28	4	0	0.143
libmdm.so	cg.B.8	6	0	0	0.000
ld-2.3.4.so	cg.B.8	5	2	3	0.400
libelan4.so.1	cg.B.8	2	3	0	1.500
sunrpc	cg.B.8	0	1	0	0.000
libimf.so.6	cg.B.8	0	1	0	0.000

# Sampling: Fonction view

The screenshot displays the Vtune(TM) Performance Analyzer interface. The left sidebar shows the 'Tuning Browser' with a tree view of the sampling session. The main window is divided into two panes. The top pane shows a table of sampling results for various functions. The bottom pane shows a table of activity results for the selected function.

**Related Topics**  
 About Sampling Hotspot View

**Tuning Browser**  
 Project1  
 CG\_8  
 Mon Apr 16 15:54:22 2007 - Sampling  
 Run 1  
 CPU\_CYCLES  
 IA64\_INST\_RETIRED-THIS  
 FP\_OPS\_RETIRED

Display Name	CPU_CYCLES samples	IA64_INST_RETIRED-THIS samples	FP_OPS_RETIRED samples	Events	Total
cpu_gnd_	26,215	23,859		CPU_CYCLES %	98.29
sparse_	231	107		CPU_CYCLES events	41...
makea_	124	142		CPU_CYCLES samples	26...
randk_	89	74		FP_OPS_RETIRED %	99.36
MAIN_	8	16		FP_OPS_RETIRED events	4.6...
cvf_kee_t_to_text_ex	3	0		FP_OPS_RETIRED samples	75...
\$\$\$155_3\$TAG\$13\$054	1	0		IA64_INST_RETIRED-THIS %	98.60
				IA64_INST_RETIRED-THIS events	38...
				IA64_INST_RETIRED-THIS sa...	23...
				IA64_IPC	0.91

Activity ID	Activity Result	Total Samples	Duration	Machine N	CPU ID	H/W Package	Idle time
10	Mon Apr 16 15:54:22 2007 - Sampling Results [ns17]	3863100	60.45	ns17	0	0	49.83%
1					1	1	49.59%
2					2	2	50.20%

Processes Threads Modules Hotspots

Console

## Command line fonction view

vtl view f1::r1 -hf -mn cg.B.8

Name	ModuleName	CPU_CYCLES	IA64_INST_RETIRED-THIS	FP_OPS_RETIRED	IA64_IPC
conj_grad_	cg.B.8	209425	189597	600975	0.905
sparse_	cg.B.8	1820	810	696	0.445
makea_	cg.B.8	928	1091	1079	1.176
randlc_	cg.B.8	739	609	1457	0.824
MAIN__	cg.B.8	69	121	589	1.754
cvt_ieee_t_to_text_ex	cg.B.8	3	0	0	0.000
\$\$1\$5_3\$TAG\$13\$0\$4	cg.B.8	1	0	0	0.000
setup_submatrix_info_	cg.B.8	0	0	1	0.000
cvtas_t_to_a	cg.B.8	0	1	0	0.000

# Sampling: line view

The screenshot shows the Intel VTune Performance Analyzer interface. The top menu bar includes File, Edit, Refactor, Navigate, Search, Project, Timing, Run, Window, and Help. The main window is titled 'Welcome' and shows a sampling session for 'ns17' on 'Mon Apr 16 15:54:22 2007'. The 'Line View' is active, displaying source code with line numbers 1099 to 1113. The code includes a loop for matrix-vector multiplication and a summation operation. To the right of the code, a table shows performance metrics for CPU, IA64, and FP operations. Below the code, a 'Size' table lists memory usage for various components, with 'conjug\_grad\_' highlighted as the selected range.

Line Number	Source	CPU_	IA64_	FP_OPS_RETI
1099	c			
1100	c q = A p			
1101	c The partition submatrix-vector multiply: use workspace w			
1102	c			
1103	do j=1,lastrow-firstrow+1	0.04%	0.04%	0.03%
1104	sum = 0.d0			
1105	do k=colstart(j),colend(j)-1	0.23%	0.46%	1.31%
1106	sum = sum + a(k/pcolids(k))	11.05%	10.75%	9.81%
1107	enddo			
1108	w(j) = sum	0.02%	0.01%	0.00%
1109	enddo			
1110				
1111	c			
1112	c Sum the partition submatrix-vec A.p's across rows			
1113	c Exchange and sum piece of w with procs identified in reduce_each_proc			

Size	Name	CPU_	IA64_	FP_OP	IA64_IPC
-- Selected Range --					
0x3f40	MAIN_	0.00%	0.01%	0.01%	1.66667
0x57c0	conjug_grad_	12.50%	12.27%	12.35%	0.886093
0x1680	setup_submatrix_info_			0.00%	
0x1400	makea_	0.06%	0.07%	0.02%	1.15254
0x1600	sparse_	0.11%	0.05%	0.01%	0.396694
0x740	setup_proc_info_				
0x180	setup_				


## Command line view

vtl view f1::r1 -code -mn cg.B.8 -fn conj\_grad\_

```
Legend
Ev1    =      CPU_CYCLES samples
Ev2    =      IA64_INST_RETIRED-THIS samples
Ev3    =      FP_OPS_RETIRED samples
Address Line Number Ev1    Ev2    Ev3    Source
0x7d90 1103        682    549    1529    do j=1,lastrow-firstrow+1
0x8121 1104         0      0      0      sum = 0.d0
0x8120 1105       3903    6796    63078    do k=rowstr(j),rowstr(j+1)-1
0x8260 1106     185751 166594 478451    sum = sum + a(k)*p(colidx(k))
1107         0      0      0      enddo
0x8310 1108       276     85     93      w(j) = sum
1109         0      0      0      enddo
```

# Sampling

## Modify

 You are about to modify an Activity that already has results associated with it

Would you like to:

☒ Make a copy of the Activity and modify the copy

☐ Modify the selected Activity

☐ Make selection as default and don't show me this dialog again

OK Cancel

General

Activity Name: Copy of Sampling Activity

Collector(s)	Application/Module Profile(s)
Sampling	TF

Configure... Configure...

Activity Duration:

☒ No time limit

☐  second(s)

☐ Start with data collection paused

OK Cancel Apply Help



# Sampling

## Properties

General	General
Events	
Ratios	

Sampling mechanism

☐ Time-based sampling (TBS)

☒ Event-based sampling (EBS)

☒ Don't calibrate sample after value

☐ Calibrate sample after value for all the selected events

☐ Calibrate sample after value based on the selected event properties

Sampling collection options

Sampling interval (milliseconds):

Sampling buffer size (KB):

☐ Delay sampling (seconds):

☒ Track thread creation

☒ Terminate application(s) when data collection finishes

Stop collection

☒ When application(s) terminate (before duration expires)

☐ Maximum samples collected:

List of CPUs where samples will be collected:

☐ All ☒ All Available CPU

☐ System-wide CPU numbers:

☐ Available CPU indexes:

Enter CPU numbers and/or ranges separated by commas. For example, 0,5,86-127.

OK Cancel Apply Help

# Sampling

## Events

General  
**Events**  
Ratios

**Events**

Event groups: All Events

Available events:

- ALAT\_CAPACITY\_MISS-ALL
- ALAT\_CAPACITY\_MISS-FP
- ALAT\_CAPACITY\_MISS-INT
- ALAT\_CAPACITY\_MISS-NONE
- BACK\_END\_BUBBLE-ALL
- BACK\_END\_BUBBLE-FE
- BACK\_END\_BUBBLE-L1D\_FPU\_RSE

Event description:  
Please select event for display description.

Selected events:

Event Name	Sample After
CPU_CYCLES	1600000
IA64_INST_RETIRED-THIS	1600000

Run Information:  
Calibration runs: 0  
Sampling runs: 1

OK Cancel Apply Help

# Call Graph

*File->New->Project...*

Select a wizard



Wizards:

- ▶ C
- ▶ C++
- ▶ CVS
- ▶ Simple
- ▶ Tuning Activity



< Back

Next >

Finish

Cancel

Select a wizard



Create an activity with the call graph data collector. VTune analyzer will instrument and profile your application and display a call graph.

Wizards:

- ▶ C
- ▶ C++
- ▶ CVS
- ▶ Simple
- ▼ Tuning Activity
  - ▶ Call Graph Wizard
  - ▶ First Use Wizard
  - ▶ Sampling Wizard



< Back

Next >

Finish

Cancel

# Call Graph

## Options

### Options

Choose any desired options and Finish.

Project settings


Change the project in which this Activity will be created.

Current setting: Project1

Directory: /home/berthek/example\_vtune\_callgraph/Project1

Modify configuration

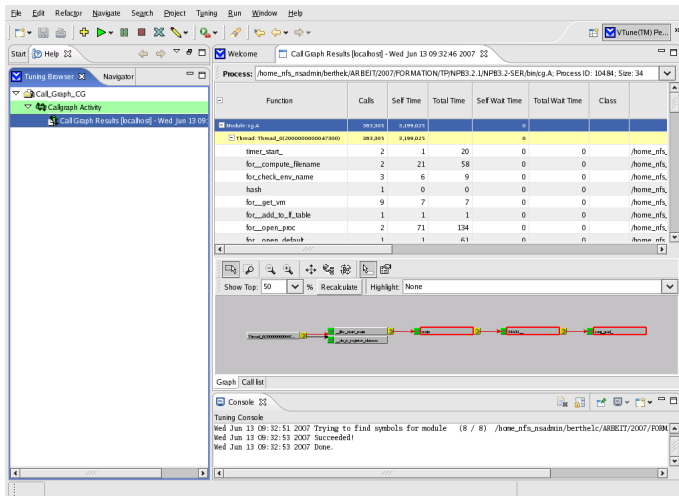
☐ Change the Call Graph instrumentation level and set advanced options.  
(Opens a dialog after this wizard creates the Activity, but before it is run.  
On large applications it can take a long time before the dialog opens.)

This new Activity will appear in the Tuning Browser when you click 'Finish'.  
To run the Activity, check the box below or click the run button  after you finish.

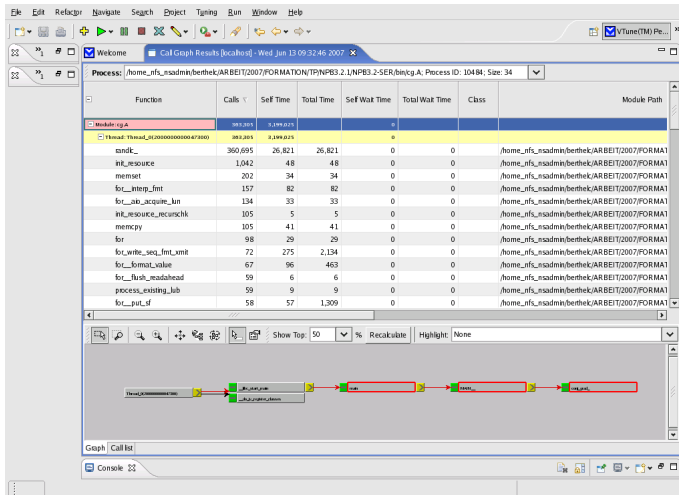
☒ Run this Activity

# Call Graph

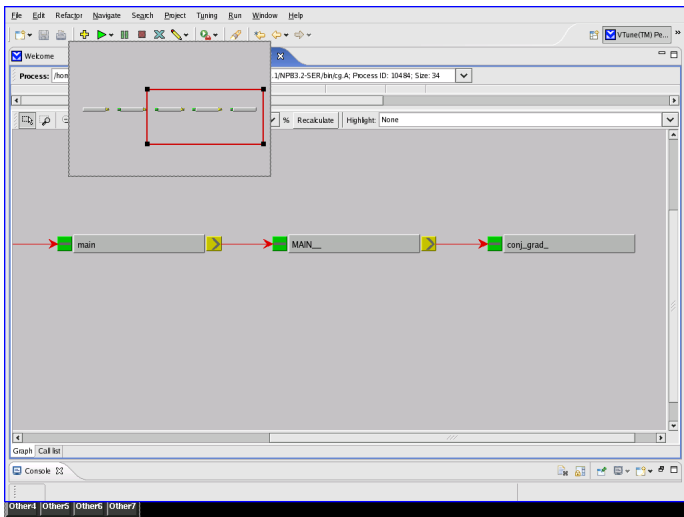
## Results



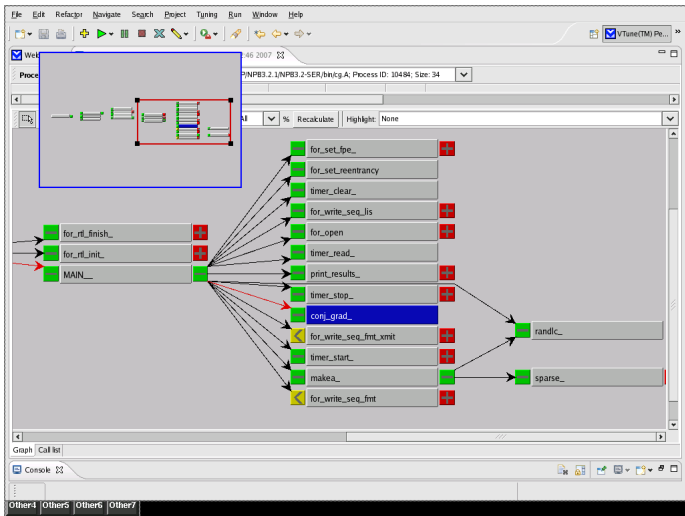
# Call Graph



# Call Graph



## Call Graph





- Bubble Analysis
- Latency Analysis
- First dimension: The node
- Second dimension The interconnect
  - MPI Analyser
  - Intel Trace Analyzer
- Questions

### MPIAnalyser

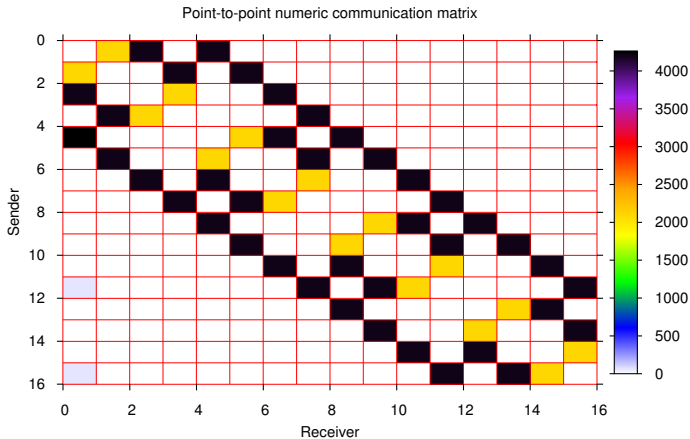
MPIAnalyser is a profiling tool dedicated to MPI applications.

- ▶ Light: it uses few resources and so does not slow down the application.
- ▶ Easy to run: it is used to characterize the MPI communications in a program. Communication matrices are constructed with it. Profilecomm is a "post-mortem" tool.

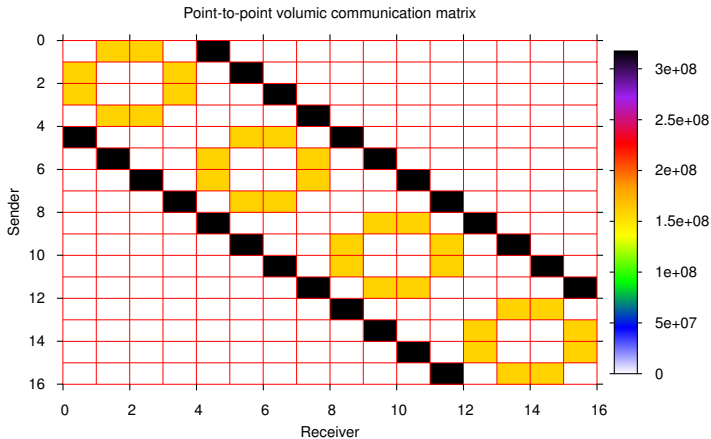
## How to use it ?

- ▶ Link with -Impianalyser
- ▶ Use environment
  - MPIANALYSER\_PROFILECOMM=1
  - MPIANALYSER\_GLOBAL=1
- ▶ Run your application
- ▶ Use tool: readpfc

## Point to Point numeric command matrix



## Point to Point Volumic communication matrix



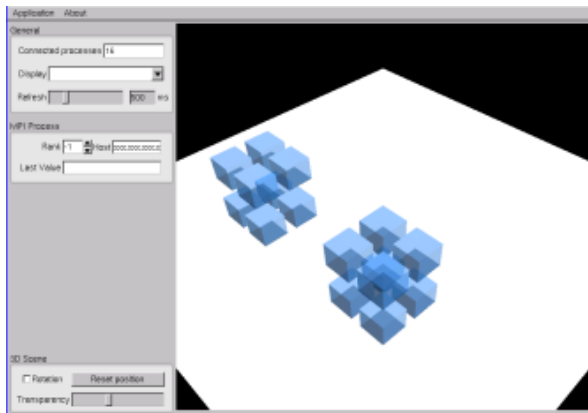
## How to use it ?

### On Compute node

```
export MPIANALYSER_GLOBAL=1
export MPIANALYSER_DATA_COLL=1
export MPIANALYSER_OUTPUT_SOCKET=1
export MPIANALYSER_CALL=1
export MPIANALYSER_VOLUME=1
export MPIANALYSER_TIME=1
export MPIANALYSER_VISU_SERVER_IP=xxx.xxx.xxx.xxx
export MPIANALYSER_VISU_SERVER_PORT=12345
```

### On Vizual node

```
export MPIANALYSER_VISU_SERVER_PATH=/opt/mpi/mpibull2-X/lib/mpianalyser/
export MPIANALYSER_VISU_SERVER_PORT=12345
mpibull2-visualize
```



## Communications: advanced Level

### *Intel Trace Tools : Collector and Analyser*

- ▶ Timeline Views and Parallelism Display
- ▶ Communication Statistics
- ▶ Execution Statistics
- ▶ Profiling Library
- ▶ MPI Bull support Intel Trace Tools





## How to use it ?

### Level 1

- ▶ Environment: source itacvars.sh or use a module file
- ▶ Link with `-L$VT_LIB_DIR -IVT -ldwarf -lelf -ltunwind -lnsl -lm -ldl -lpthread` or `$VT_ADD_LIBS`
- ▶ Run your application
- ▶ Run traceanalyzer

## Level 2 with Class

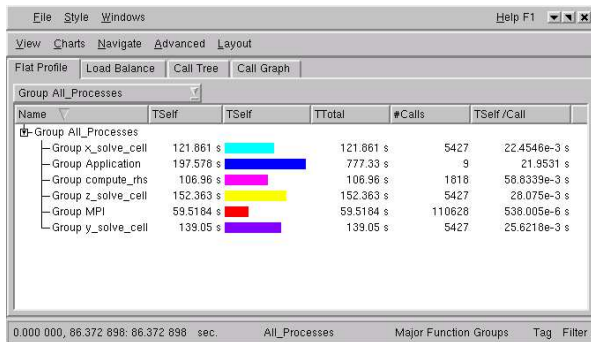
- ▶ At the beginning

```
include "VT.inc"  
integer function1,VT_ierr,class1  
call VTCLASSDEF("GODFINE1",class1,VT_ierr)  
call VTFUNCDEF("GODFINE1",class1,function1,VT_ierr)  
call VTENTER(function1,VT_NOSCL,VT_ierr)
```

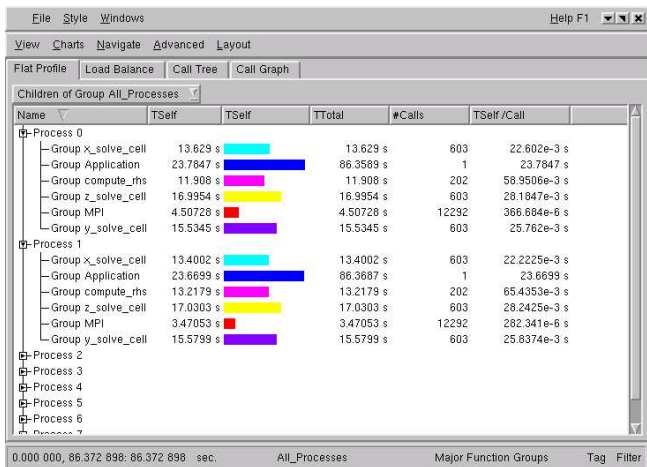
- ▶ At the end

```
call VTLEAVE(function1,VT_ierr)
```

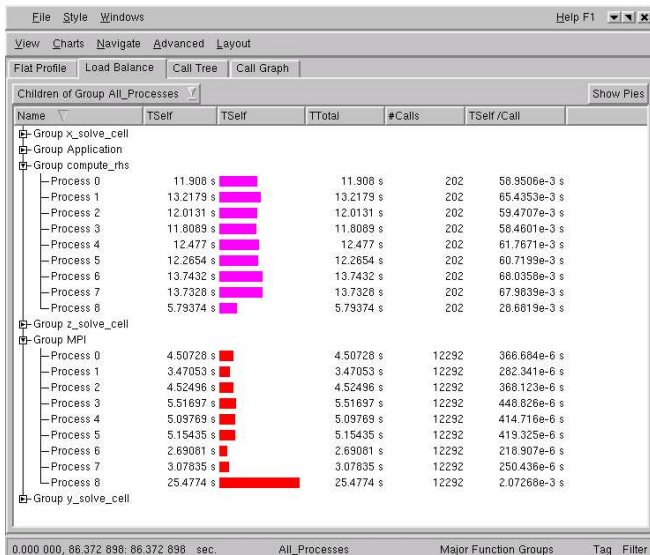
## Example BT from NAS



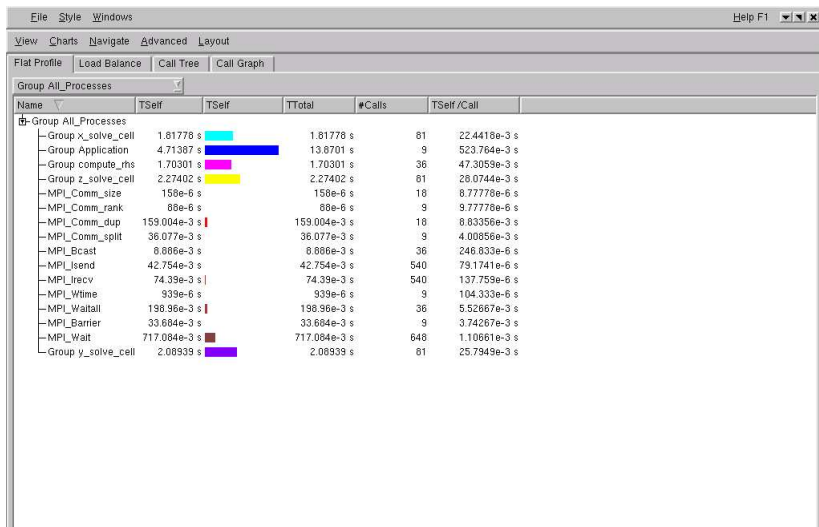
# Example



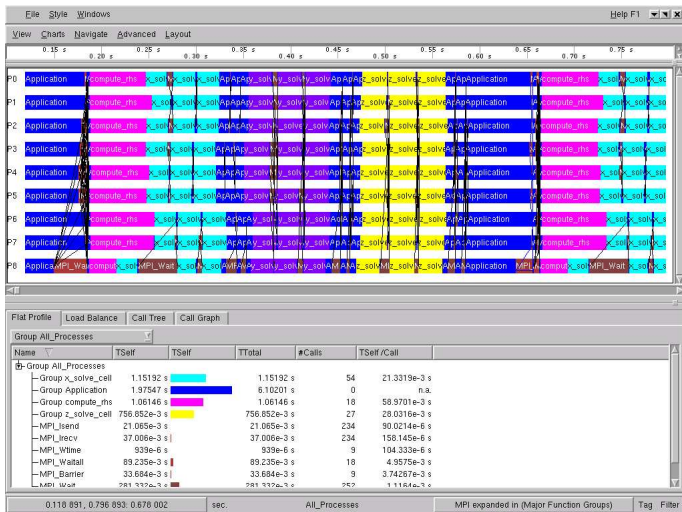
## Example: Load balance



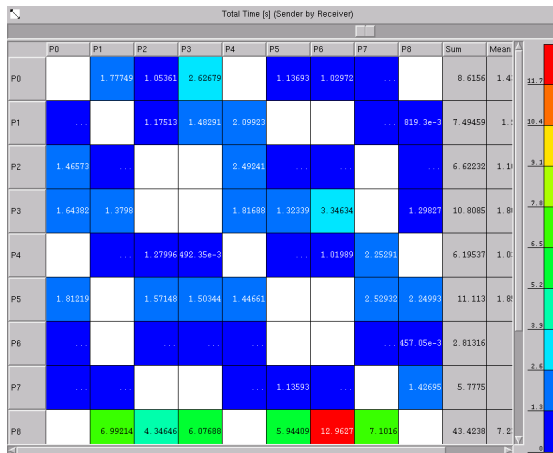
## Example: Ungroup MPI



# Example



# Example

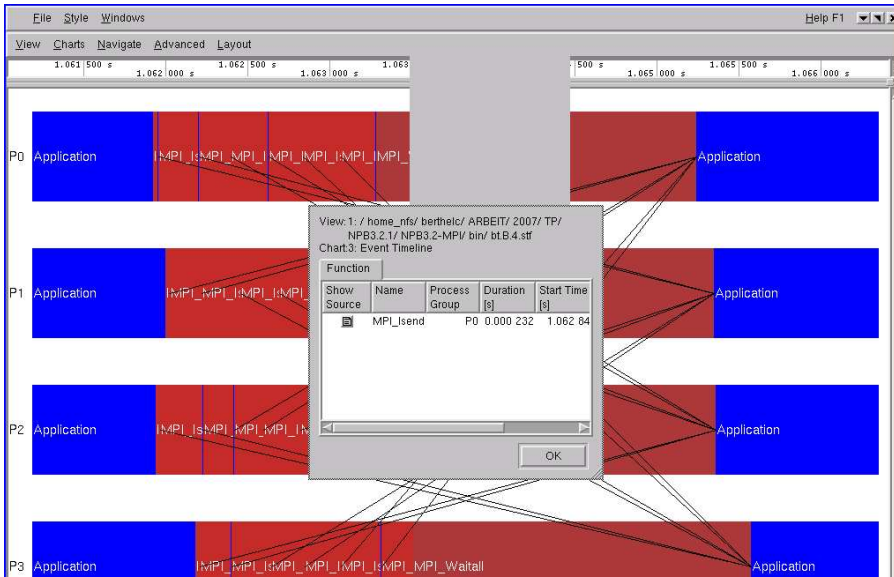




# Source and Profiling

## Compile and Run

- ▶ You need compile and link with -g
- ▶ You need export VT\_PCTRACE=5



View: 1: / home\_nfs/ berthelc/ ARBEIT/ 2007/ TP/ NPB3.2.1/ NPB3.2-MPI/ bin/ bt.b4.stf  
Chart3: Event Timeline

Process 0

```
193 > dp_type, successor(1), EAST,  
194 > comm_rhs, requests(6), error)  
195 call mpi_isend(out_buffer(ss(1)), b_size(1)  
196 > dp_type, predecessor(1), WEST,  
197 > comm_rhs, requests(7), error)  
198 call mpi_isend(out_buffer(ss(2)), b_size(2)  
199 > dp_type, successor(2), NORTH,  
200 > comm_rhs, requests(8), error)  
201 call mpi_isend(out_buffer(ss(3)), b_size(3)  
202 > dp_type, predecessor(2), SOUTH,  
203 > comm_rhs, requests(9), error)  
204 call mpi_isend(out_buffer(ss(4)), b_size(4)  
205 > dp_type, successor(3), TOP,  
206 > comm_rhs, requests(10), error)  
207 call mpi_isend(out_buffer(ss(5)), b_size(5)  
208 > dp_type, predecessor(3), BOTTOM,  
209 > comm_rhs, requests(11), error)  
210  
211  
212 call mpi_waitall(12, requests, statuses, er:  
213  
214 -----  
215 c unpack the data that has just been received  
216 -----  
217 p0 = 0
```

Call stack:

/home\_nfs/berthelc/ARBEIT/2007/TP/NPB3.2.1/NPB3.2-MPI/bt\_itac/copy\_faces.f, line 204

OK

Questions ?

## For Further Reading



HPC BAS4 User's Guide

REFERENCE 86 A2 29ER 03.



HPC BAS4 Application Tuning Guide

REFERENCE 86 A2 19ER 00.



Black Belt Itanium 2 Processor Performance: Use of Constrained Event Collection



Intel Itanium 2 Processor Reference Manual for Software Development and Optimization

Order Number: 251110-003



Architect of an Open World™



- ▶ (c) Copyright Bull. All rights reserved
  - ✓ Users Restricted Rights - Use, duplication or disclosure restricted.
  - ✓ Any copy of these documents should keep all copyright, logos and other proprietary notices contained herein.
  - ✓ This publication may include technical inaccuracies or typographical errors.
  - ✓ This publication is provided "AS IS" without any warranty either expressed or implied including but not limited to the implied warranties of merchantabilities or fitness of the described product.
  - ✓ Course Material Licensing Terms : No sublicensing rights.
  - ✓ For other licensing needs, please contact Bull