# C-DAC Four Days Technology Workshop

***ON***

## Hybrid Computing – Co-Processors/Accelerators Power-aware Computing – Performance of Applications Kernels

**hyPACK-2013**
**(Mode-1:Multi-Core)**

# Lecture Topic :

# Multi-Core Processors : Intel Tools

# (Thread Checker, Profiler, Performance Analyzer).

***Venue* : CMSD, UoHYD ; *Date* : October 15-18, 2013**

# Introduction

- Moving from Multiple processor on single box (SMP) Multiple Core on Single Chip.
- Two, four or even eight processor cores on the same die are fast becoming commonplace.
- Moving to a multi-core world means applications will have to be written in a different manner.
- Multicore architectures involve multi-processing, and to take advantage of that, parallel programming is almost compulsury.
- The lack of parallel-programming tools and expertise is threatening the progress of multi-core architectures.

# Cause of Poor Scalability

- Insufficient parallel work
- Synchronization overhead
- Contention
- Load imbalance
- Task granularity
- Memory bandwidth / false sharing

# Road Map To Better Performance

- Fully utilize available cores

- Identify which synchronization objects are contended and whose waiting actually affect performance

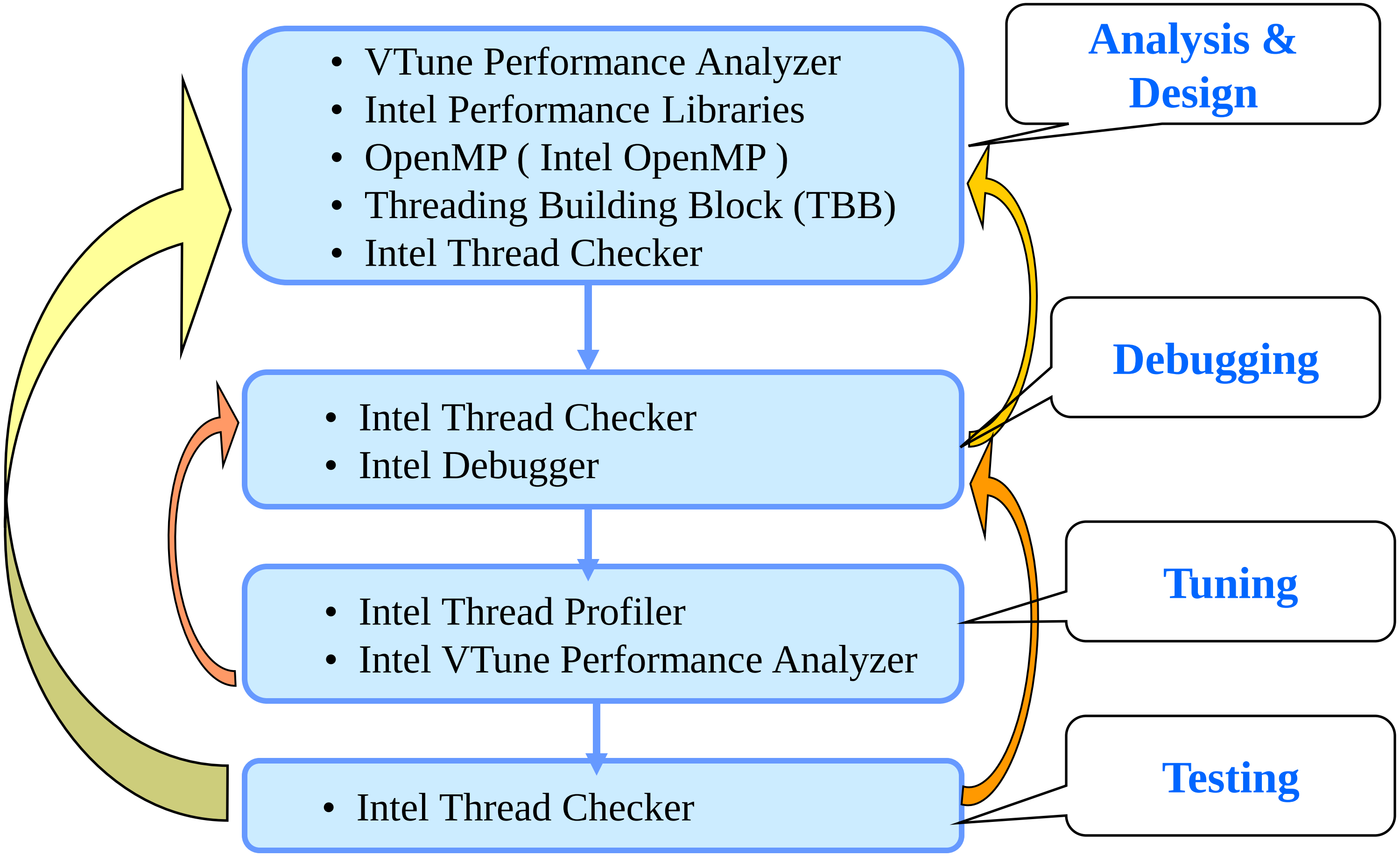
- Highlight workload imbalance

- Pinpoints issues regarding performance bottleneck in the source code

# Intel Multicore Tools

- Intel Thread Checker.
- Intel Thread Profiler.
- Intel VTune Performance Analyzer.

# Performance Improvement Cycle



- VTune Performance Analyzer
- Intel Performance Libraries
- OpenMP ( Intel OpenMP )
- Threading Building Block (TBB)
- Intel Thread Checker

- Intel Thread Checker
- Intel Debugger

- Intel Thread Profiler
- Intel VTune Performance Analyzer

- Intel Thread Checker

# Intel Thread Checker : Features

Intel® Thread Checker detects data races, deadlocks, stalls, and other threading issues. It can detect the potential for these errors even if the error does not occur during an analysis session.

- ❖ Detect the potential errors.
- ❖ Filter out specific types of Diagnostics
- ❖ Identify critical source locations
- ❖ Get tips to improve the robustness

# Intel Thread Checker : Benefits

Intel® Thread Checker detects data races, deadlocks, stalls, and other threading issues. It can detect the potential for these errors even if the error does not occur during an analysis session.

- ❖ Pinpoint the function, context, line, variable, and call stack in the source code to aid analysis and repair of bugs
- ❖ Identify nearly impossible-to-find data races and deadlocks using an advanced error detection engine. Helps to reduce untraceable errors.
- ❖ Instrumental for effective design of threaded applications
- ❖ Errors do not need to actually occur to be detected. Make the code as more robust

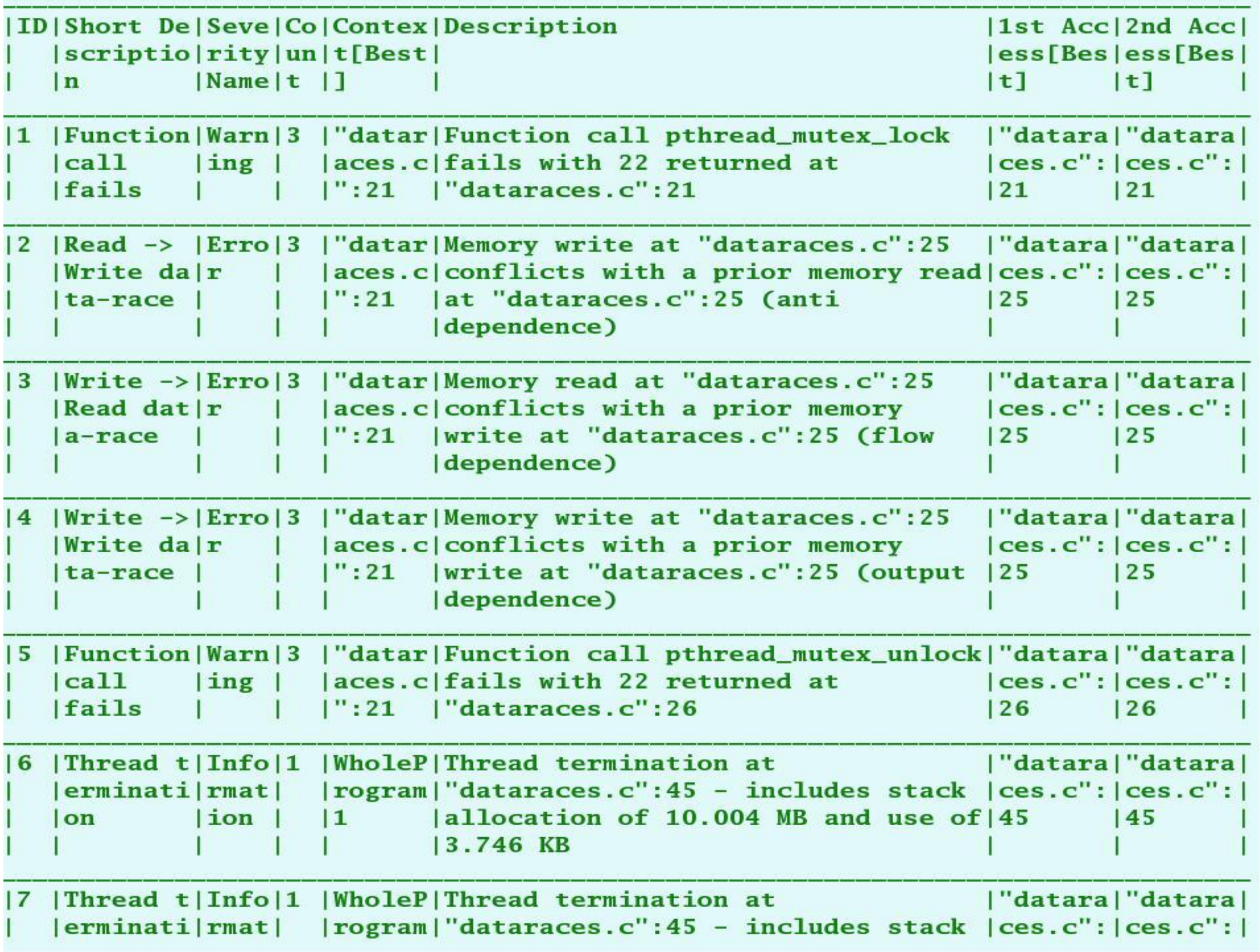
# Intel Thread Checker : Case Study

```
#define   NTHREADS  4
int   globalX = 0;        pthread_mutex_t cs;
```

```
int main (int argc, char *argv[])
{
    pthread_t h[NTHREADS];
    int rc;
    int i;

    pthread_mutex_init (&cs, 0);
    for (i = 0; i < NTHREADS; i++)  {
     rc = pthread_create (&h[i], 0
                        , increment, 0);
    }

    for (i = 0; i < NTHREADS; i++)  {
        rc = pthread_join (h[i], 0);
    }
}
```

```
void * increment (void *arg)
              {
  pthread_mutex_lock (&cs);
         globalX++;
pthread_mutex_unlock (&cs);

pthread_mutex_destroy (&cs);

         return 0;
              }
```

# Intel Thread Checker : Output



```
-------------------------------------------------------------------------------
|ID|Short De|Seve|Co|Contex|Description                       |1st Acc|2nd Acc|
|  |scriptio|rity|un|t[Best|                                  |ess[Bes|ess[Bes|
|  |n       |Name|t |]     |                                  |t]     |t]     |
-------------------------------------------------------------------------------
|1 |Function|Warn|3 |"datar|Function call pthread_mutex_lock  |"datara|"datara|
|  |call    |ing |  |aces.c|fails with 22 returned at         |ces.c":|ces.c":|
|  |fails   |    |  |":21  |"dataraces.c":21                  |21     |21     |
-------------------------------------------------------------------------------
|2 |Read -> |Erro|3 |"datar|Memory write at "dataraces.c":25  |"datara|"datara|
|  |Write da|r   |  |aces.c|conflicts with a prior memory read|ces.c":|ces.c":|
|  |ta-race |    |  |":21  |at "dataraces.c":25 (anti         |25     |25     |
|  |        |    |  |      |dependence)                       |       |       |
-------------------------------------------------------------------------------
|3 |Write ->|Erro|3 |"datar|Memory read at "dataraces.c":25   |"datara|"datara|
|  |Read dat|r   |  |aces.c|conflicts with a prior memory     |ces.c":|ces.c":|
|  |a-race  |    |  |":21  |write at "dataraces.c":25 (flow   |25     |25     |
|  |        |    |  |      |dependence)                       |       |       |
-------------------------------------------------------------------------------
|4 |Write ->|Erro|3 |"datar|Memory write at "dataraces.c":25  |"datara|"datara|
|  |Write da|r   |  |aces.c|conflicts with a prior memory     |ces.c":|ces.c":|
|  |ta-race |    |  |":21  |write at "dataraces.c":25 (output |25     |25     |
|  |        |    |  |      |dependence)                       |       |       |
-------------------------------------------------------------------------------
|5 |Function|Warn|3 |"datar|Function call pthread_mutex_unlock|"datara|"datara|
|  |call    |ing |  |aces.c|fails with 22 returned at         |ces.c":|ces.c":|
|  |fails   |    |  |":21  |"dataraces.c":26                  |26     |26     |
-------------------------------------------------------------------------------
|6 |Thread t|Info|1 |WholeP|Thread termination at             |"datara|"datara|
|  |erminati|rmat|  |rogram|"dataraces.c":45 - includes stack |ces.c":|ces.c":|
|  |on      |ion |  |1     |allocation of 10.004 MB and use of|45     |45     |
|  |        |    |  |      |3.746 KB                          |       |       |
-------------------------------------------------------------------------------
|7 |Thread t|Info|1 |WholeP|Thread termination at             |"datara|"datara|
|  |erminati|rmat|  |rogram|"dataraces.c":45 - includes stack |ces.c":|ces.c":|
```

# Intel Thread Checker : Whats Gone Wrong

```
#define   NTHREADS  4
int   globalX = 0;        pthread_mutex_t cs;
```

Declearing mutex at global location

```
int main (int argc, char *argv[])
{
   pthread_t h[NTHREADS];
   int rc;
   int i;

   pthread_mutex_init (&cs, 0);
   for (i = 0; i < NTHREADS; i++)  {
      rc = pthread_create (&h[i],
increment, 0);
   }

   for (i = 0; i < NTHREADS; i++)  {
      rc = pthread_join (h[i], 0);
   }
}
```

Initializing mutex with in main thread

# Intel Thread Checker : Whats Gone Wrong

```
#define   NTHREADS  4
int   globalX = 0;         pthread_mutex_t cs;
```

Destroying mutex within thread after completion of execution

```
void * increment (void *arg)
{
   pthread_mutex_lock (&cs);
   globalX++;
   pthread_mutex_unlock (&cs);

   pthread_mutex_destroy (&cs);

   return 0;
}
```

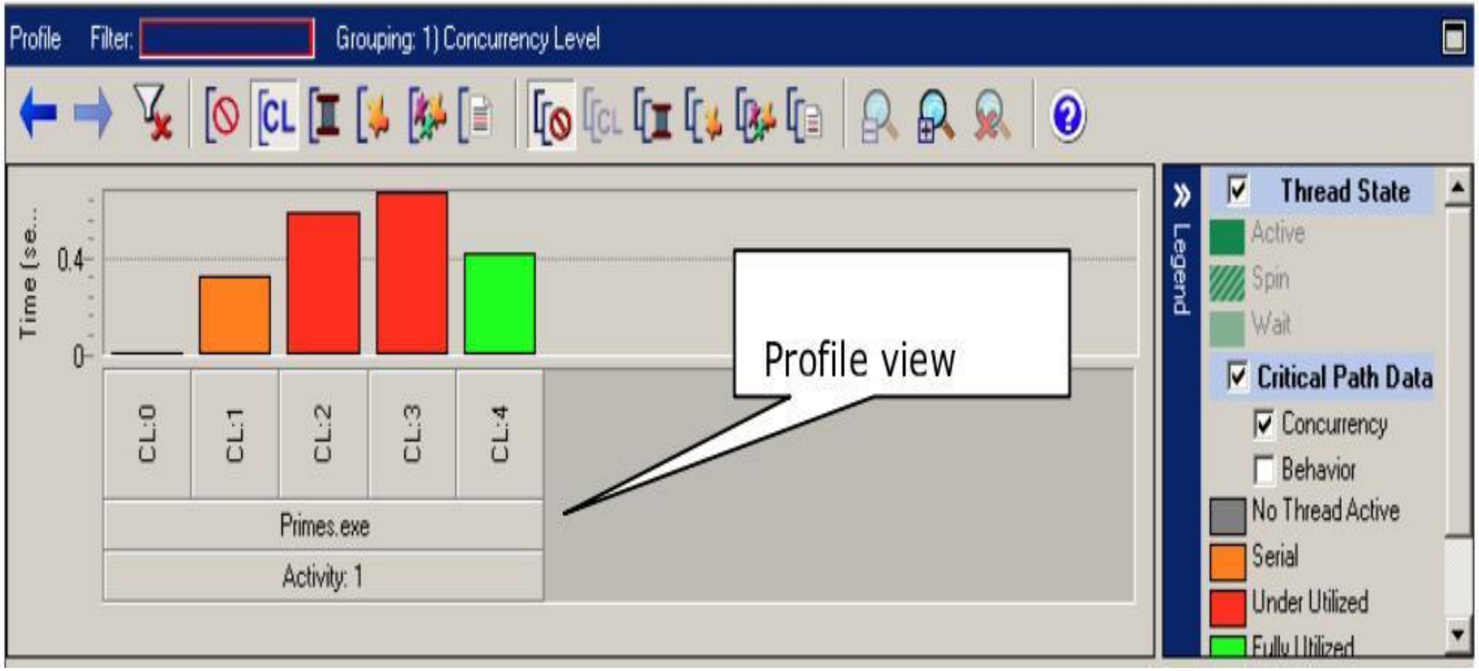
# Intel Thread Profiler

Intel® Thread Profiler helps you to improve the performance of applications threaded with Windows API, OpenMP, or POSIX threads (Pthreads).

- ❖ Identify bottlenecks that limit the parallel performance of your multi threaded application.
- ❖ Locate synchronization delays, stalled threads, excessive blocking time, and ineffective utilization of processors.
- ❖ Find the best sections of code to optimize for sequential performance and for threaded performance.
- ❖ Compare scalability across different numbers of processors or using different threading methods.

# Intel Thread Profiler : Profiler View

The Profile view (on top) displays a high-level summary of the time spent on the critical path, decomposed into time categories.


Profile Filter: Grouping: 1) Concurrency Level

☑ Thread State

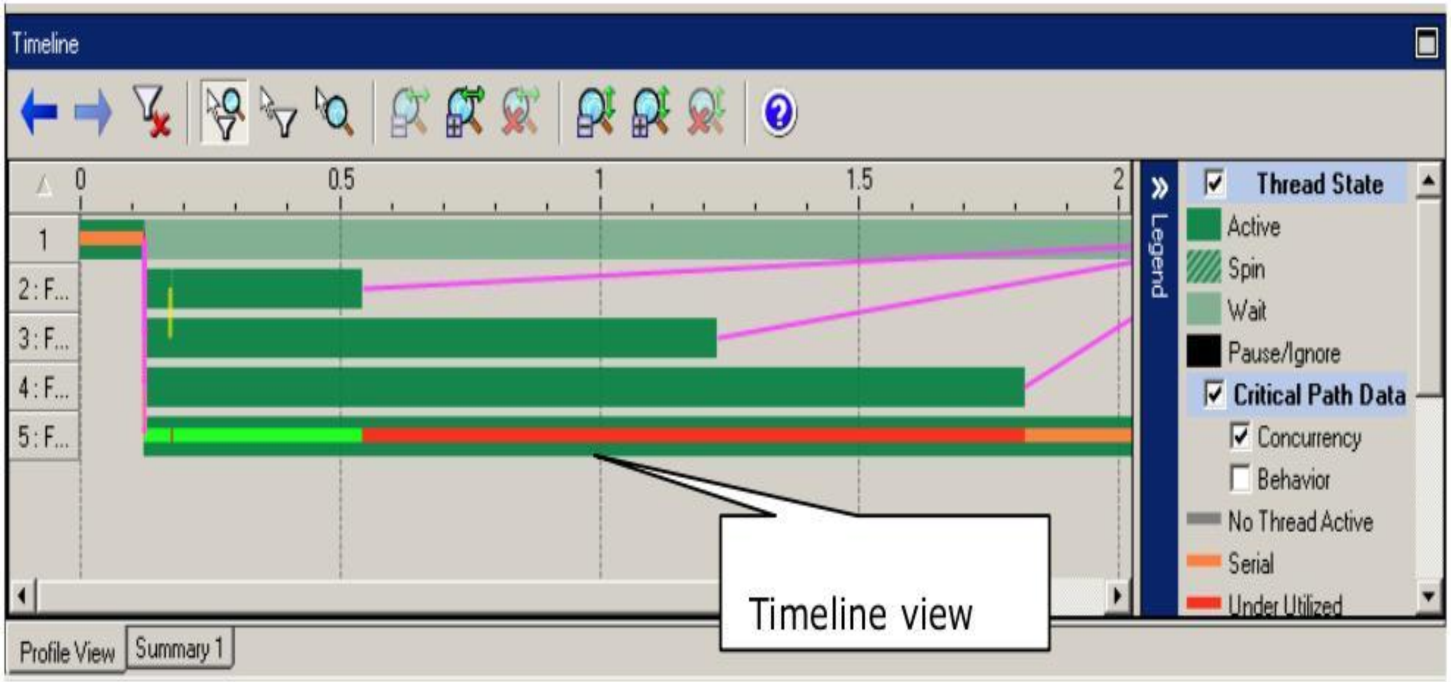
☑ Critical Path Data

☑ Concurrency

□ Behavior

# Intel Thread Profiler : Time Line View

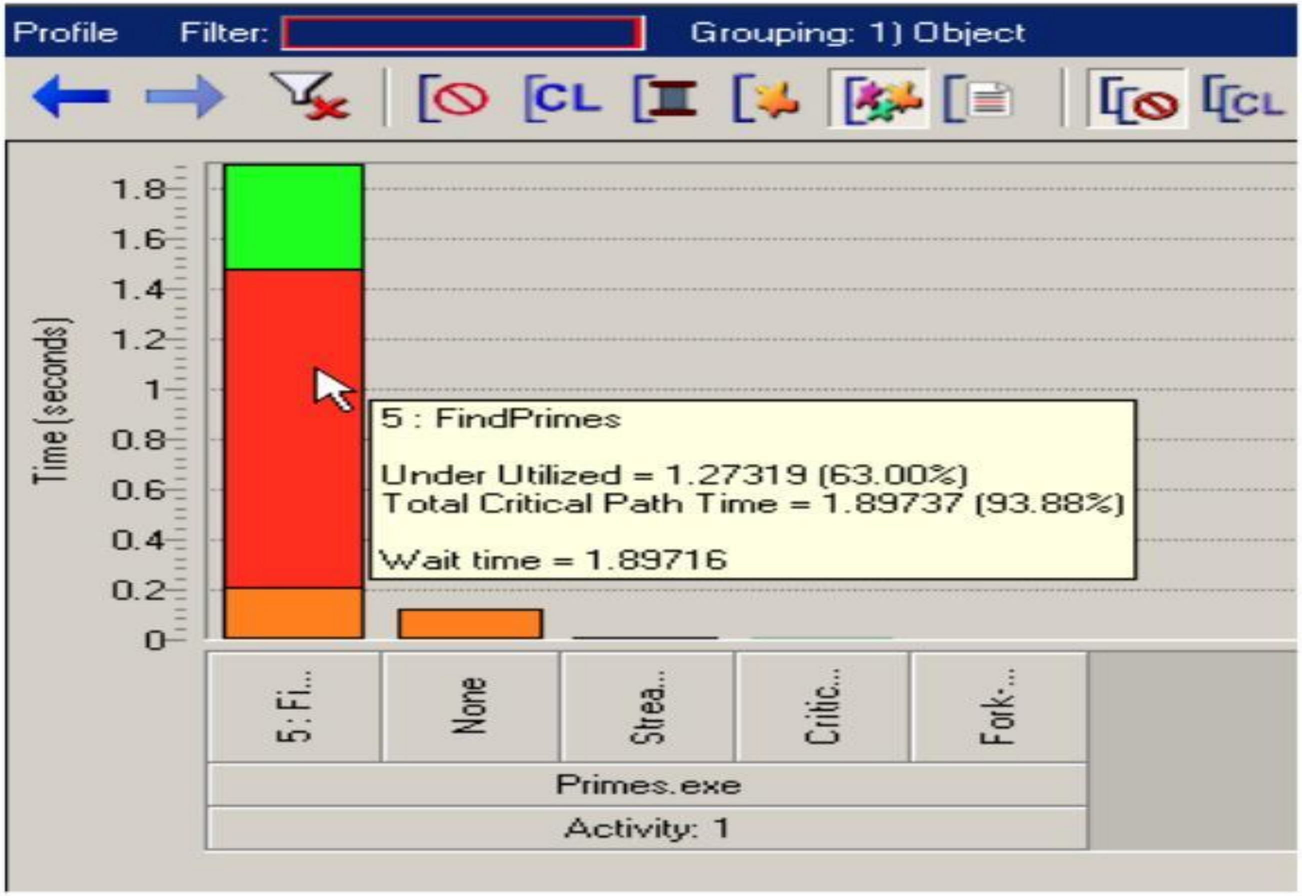
The Timeline view (on bottom) illustrates the behavior of your program over time.



Timeline view

# Intel Thread Profiler : Case Study

In the following case, the majority of time was spent in under utilized (red) time.



5 : FindPrimes

Under Utilized = 1.27319 (63.00%)
Total Critical Path Time = 1.89737 (93.88%)

Wait time = 1.89716

# Intel Vtune Performance Analyzer

The VTune™ Performance Analyzer provides information on the performance of your code. The VTune analyzer shows you the performance issues, enabling you to focus your tuning effort and get the best performance boost in the least amount of time.

- Locate a performance issue
- Revise the code to remove the issue
- Compare the performance of the new code with the initial code

# Intel VTune Performance Analyzer

Three different wizard is provided to analyze an application using VTune™ Performance Analyzer

- First Use Wizard
- Sampling wizerd
- Call Graph Wizard

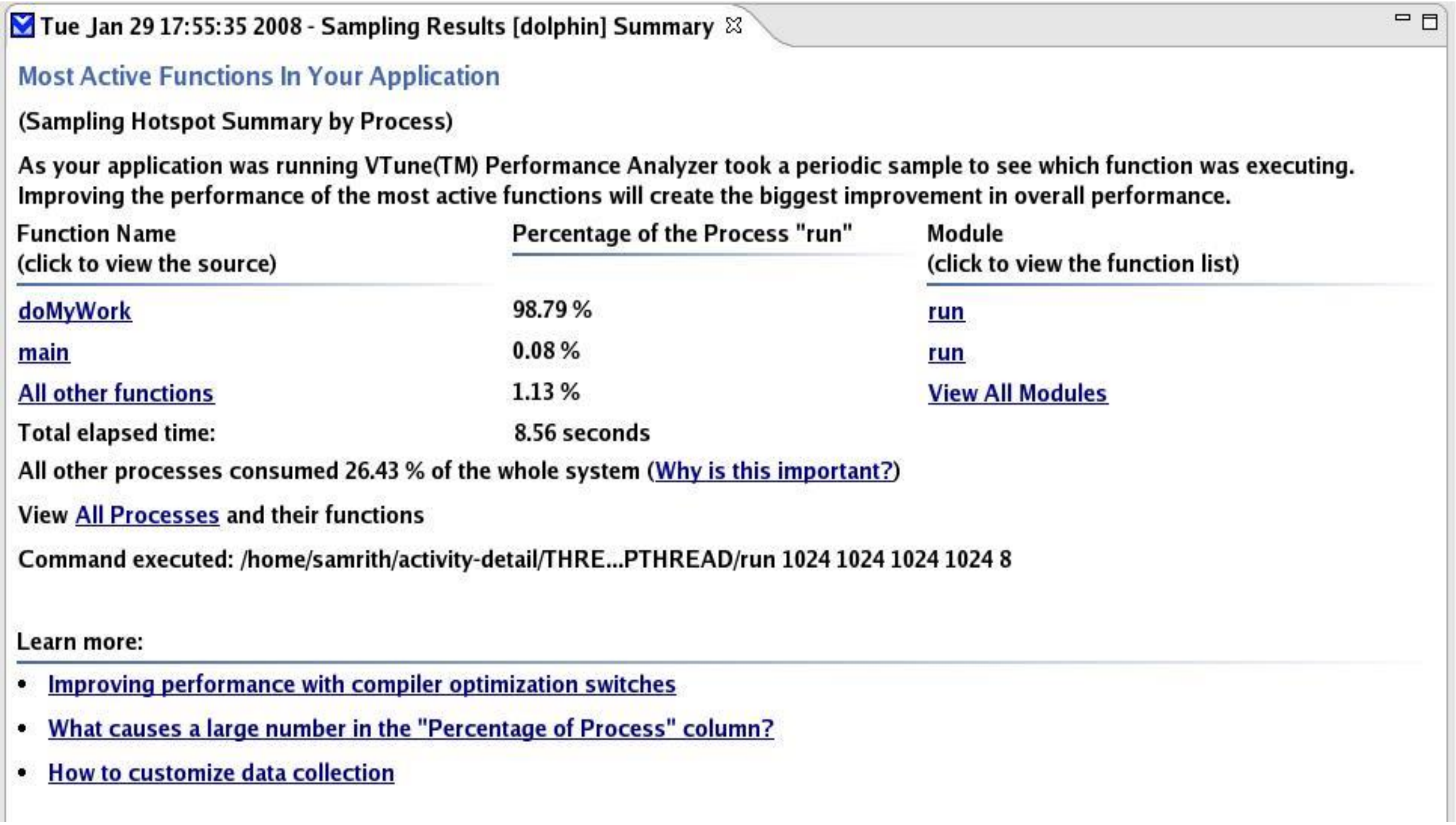
# VTune : First User Wizard

The first use wizard creates and runs a performance tuning Activity. After the Activity run is complete, a Summary view displays, showing the five most active functions in your application.

- The Activity runs the sampling collector
- Collects data on the Clock ticks processor event.
- Calculate percentage of processor time spent in each module of your application.

# VTune : First User Wizard

First Use Wizard’s output of analyzing Matrix Matrix Multiplication Code with Posix Thread



Tue Jan 29 17:55:35 2008 - Sampling Results [dolphin] Summary

Most Active Functions In Your Application

(Sampling Hotspot Summary by Process)

As your application was running VTune(TM) Performance Analyzer took a periodic sample to see which function was executing. Improving the performance of the most active functions will create the biggest improvement in overall performance.

| Function Name<br>(click to view the source) | Percentage of the Process "run" | Module<br>(click to view the function list) |
|---|---|---|
| doMyWork | 98.79 % | run |
| main | 0.08 % | run |
| All other functions | 1.13 % | View All Modules |
| Total elapsed time: | 8.56 seconds | |

All other processes consumed 26.43 % of the whole system (Why is this important?)

View All Processes and their functions

Command executed: /home/samrith/activity-detail/THRE...PTHREAD/run 1024 1024 1024 1024 8

Learn more:

- Improving performance with compiler optimization switches
- What causes a large number in the "Percentage of Process" column?
- How to customize data collection

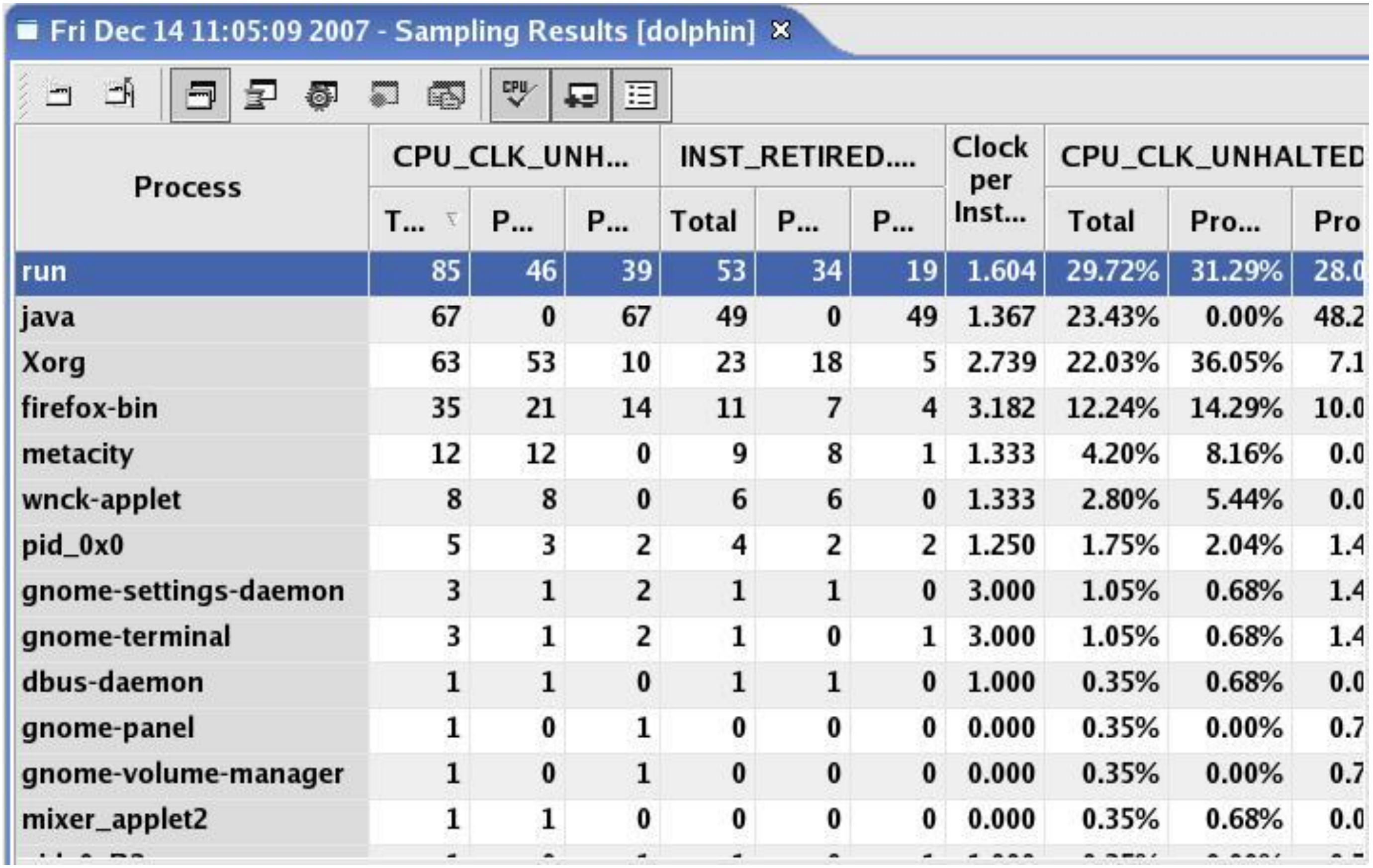
# VTune : Sampling Wizard

The VTune Performance Analyzer's sampling collector collects system-wide data.

- ❖ Sampling data collection is a non-intrusive process.
- ❖ Collect sampling data of active processes on your system
- ❖ The VTune analyzer is meant to be a statistical sampling tool and is not meant to sample after every instruction.

# VTune : Sampling Wizard

The VTune(TM) Performance Analyzer's sampling collector collects system-wide data and display in the following picture.

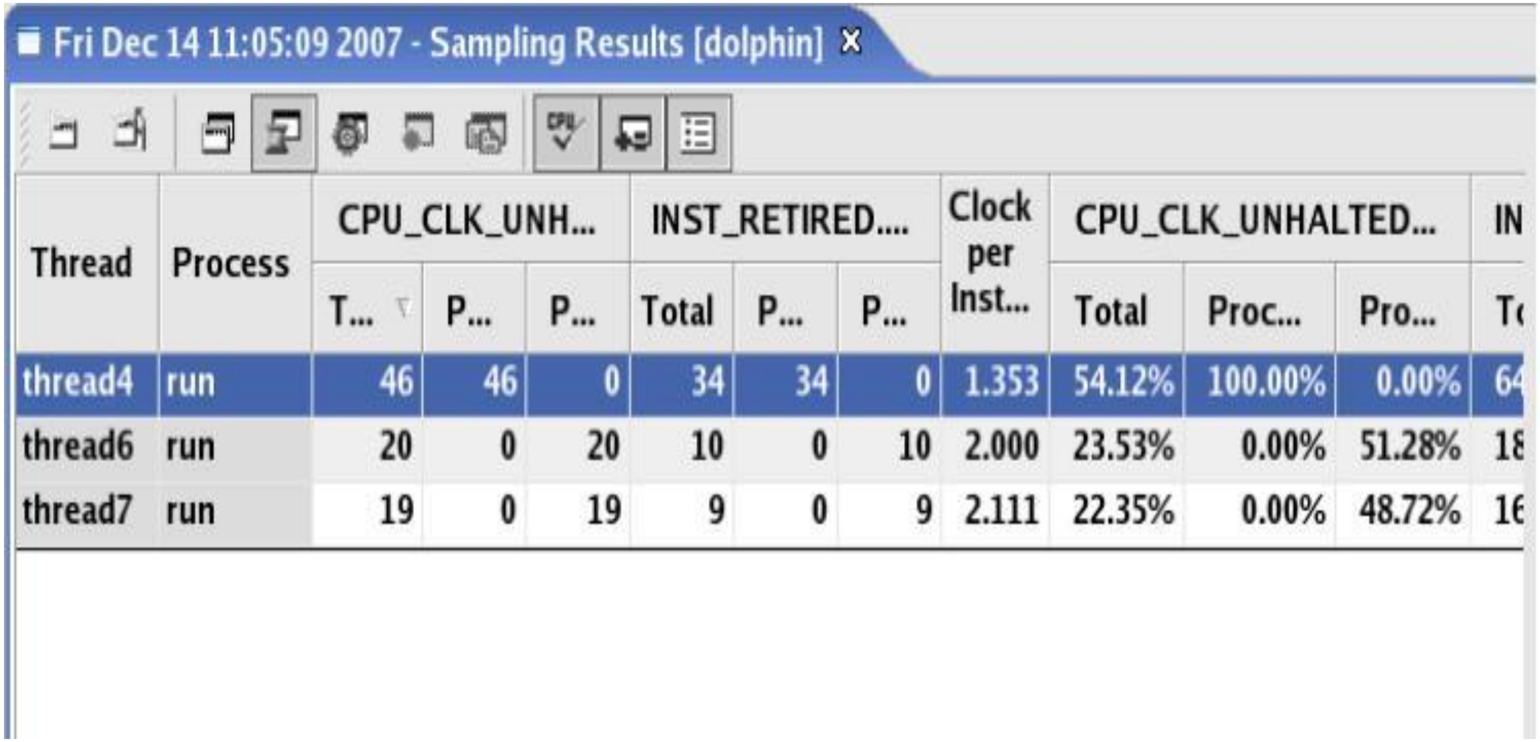


Fri Dec 14 11:05:09 2007 - Sampling Results [dolphin]

| Process | CPU_CLK_UNH... | | | INST_RETIRED.... | | | Clock per Inst... | CPU_CLK_UNHALTED | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | T... | P... | P... | Total | P... | P... | | Total | Pro... | Pro |
| run | 85 | 46 | 39 | 53 | 34 | 19 | 1.604 | 29.72% | 31.29% | 28.0 |
| java | 67 | 0 | 67 | 49 | 0 | 49 | 1.367 | 23.43% | 0.00% | 48.2 |
| Xorg | 63 | 53 | 10 | 23 | 18 | 5 | 2.739 | 22.03% | 36.05% | 7.1 |
| firefox-bin | 35 | 21 | 14 | 11 | 7 | 4 | 3.182 | 12.24% | 14.29% | 10.0 |
| metacity | 12 | 12 | 0 | 9 | 8 | 1 | 1.333 | 4.20% | 8.16% | 0.0 |
| wnck-applet | 8 | 8 | 0 | 6 | 6 | 0 | 1.333 | 2.80% | 5.44% | 0.0 |
| pid_0x0 | 5 | 3 | 2 | 4 | 2 | 2 | 1.250 | 1.75% | 2.04% | 1.4 |
| gnome-settings-daemon | 3 | 1 | 2 | 1 | 1 | 0 | 3.000 | 1.05% | 0.68% | 1.4 |
| gnome-terminal | 3 | 1 | 2 | 1 | 0 | 1 | 3.000 | 1.05% | 0.68% | 1.4 |
| dbus-daemon | 1 | 1 | 0 | 1 | 1 | 0 | 1.000 | 0.35% | 0.68% | 0.0 |
| gnome-panel | 1 | 0 | 1 | 0 | 0 | 0 | 0.000 | 0.35% | 0.00% | 0.7 |
| gnome-volume-manager | 1 | 0 | 1 | 0 | 0 | 0 | 0.000 | 0.35% | 0.00% | 0.7 |
| mixer_applet2 | 1 | 1 | 0 | 0 | 0 | 0 | 0.000 | 0.35% | 0.68% | 0.0 |
| [illegible] | 1 | 0 | 1 | 1 | 0 | 1 | 1.000 | 0.35% | 0.00% | 0.7 |

# VTune : Sampling Wizard

Display Sampling Information for specific process

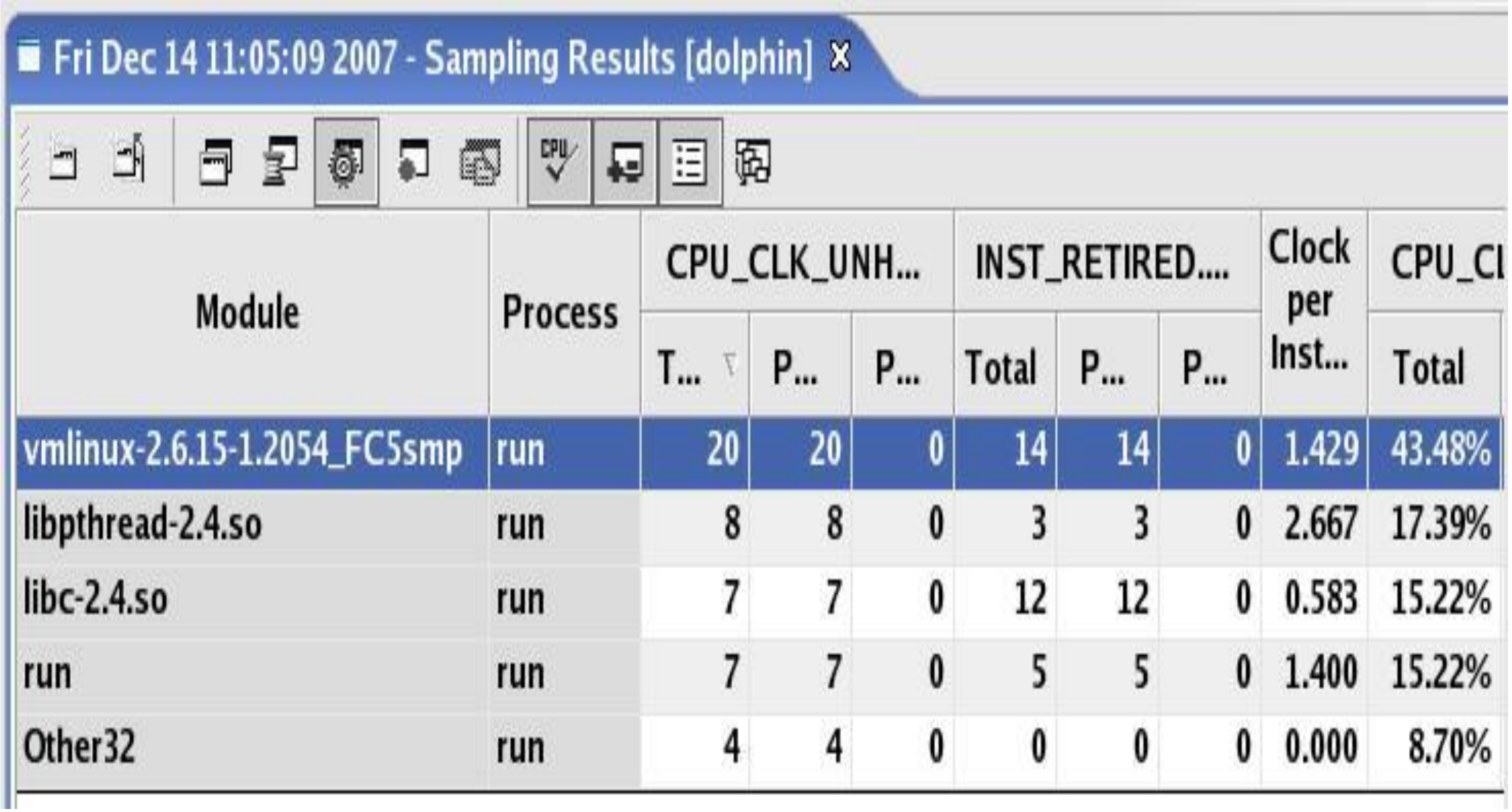


Fri Dec 14 11:05:09 2007 - Sampling Results [dolphin]

| Thread | Process | CPU_CLK_UNH... | | | INST_RETIRED.... | | | Clock per Inst... | CPU_CLK_UNHALTED... | | | IN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | T... | P... | P... | Total | P... | P... | | Total | Proc... | Pro... | To |
| thread4 | run | 46 | 46 | 0 | 34 | 34 | 0 | 1.353 | 54.12% | 100.00% | 0.00% | 64 |
| thread6 | run | 20 | 0 | 20 | 10 | 0 | 10 | 2.000 | 23.53% | 0.00% | 51.28% | 18 |
| thread7 | run | 19 | 0 | 19 | 9 | 0 | 9 | 2.111 | 22.35% | 0.00% | 48.72% | 16 |

# VTune : Sampling Wizard

Display Sampling information of specific modules of specific thread

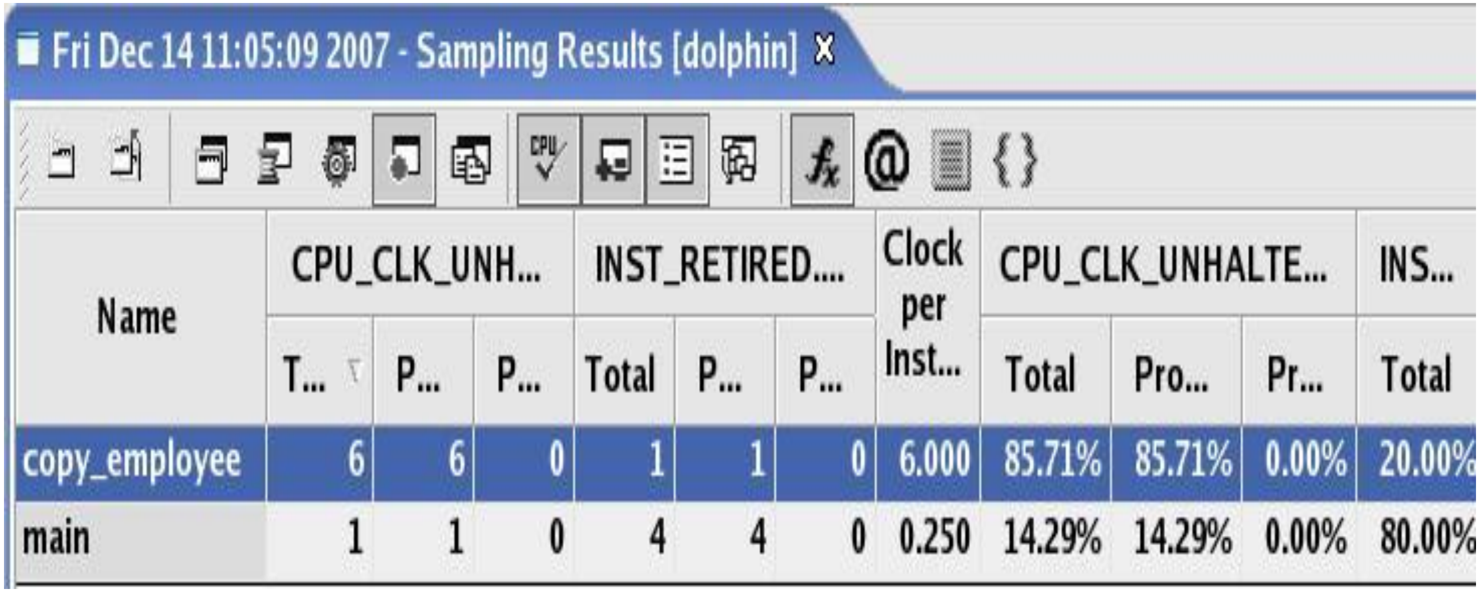


Fri Dec 14 11:05:09 2007 - Sampling Results [dolphin]

| Module | Process | CPU_CLK_UNH... | | | INST_RETIRED.... | | | Clock per Inst... | CPU_Cl |
|---|---|---|---|---|---|---|---|---|---|
| | | T... | P... | P... | Total | P... | P... | | Total |
| vmlinux-2.6.15-1.2054_FC5smp | run | 20 | 20 | 0 | 14 | 14 | 0 | 1.429 | 43.48% |
| libpthread-2.4.so | run | 8 | 8 | 0 | 3 | 3 | 0 | 2.667 | 17.39% |
| libc-2.4.so | run | 7 | 7 | 0 | 12 | 12 | 0 | 0.583 | 15.22% |
| run | run | 7 | 7 | 0 | 5 | 5 | 0 | 1.400 | 15.22% |
| Other32 | run | 4 | 4 | 0 | 0 | 0 | 0 | 0.000 | 8.70% |

# VTune : Sampling Wizard

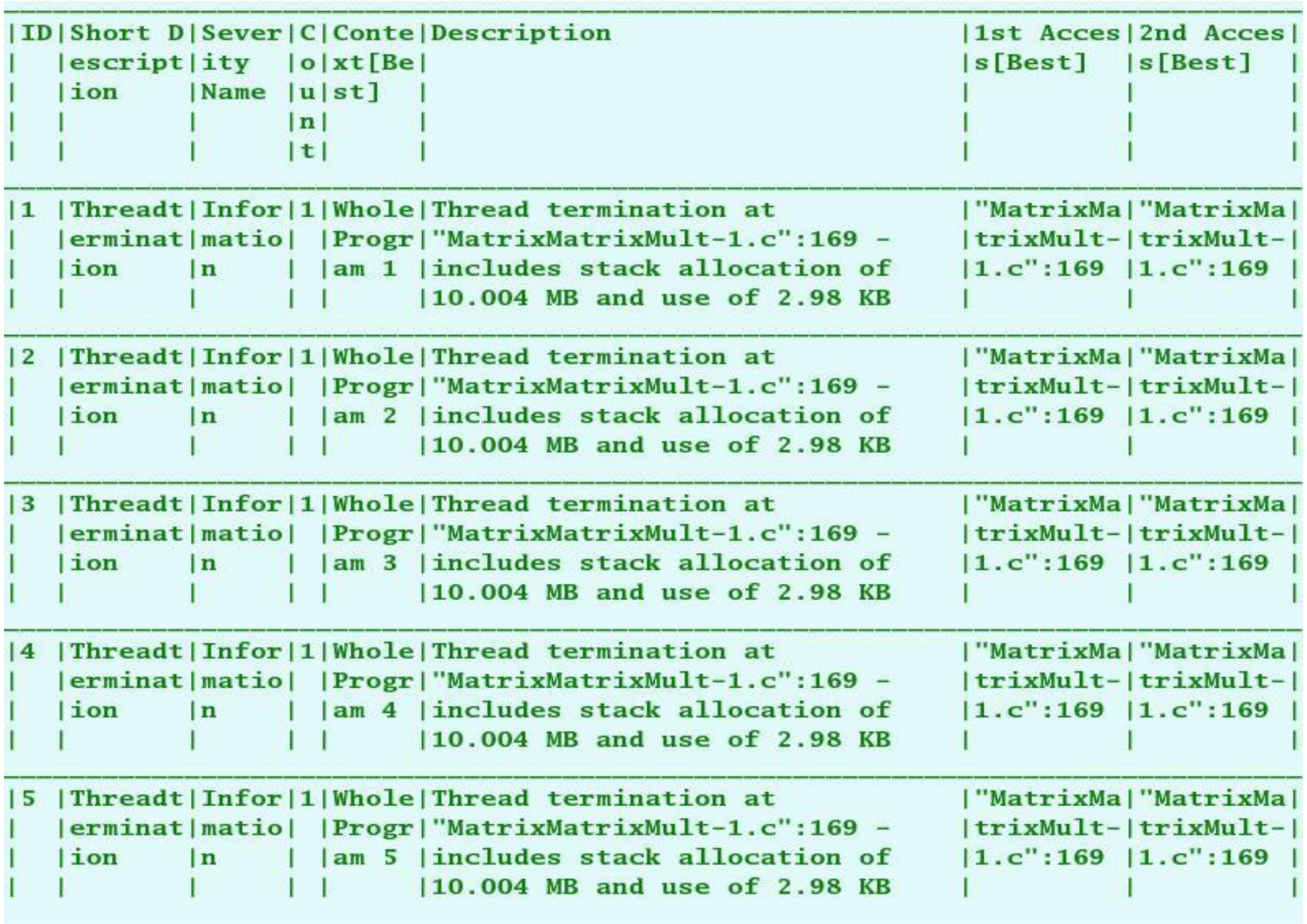
Display Sampling information of specific function of specific module of specific thread.



Fri Dec 14 11:05:09 2007 - Sampling Results [dolphin]

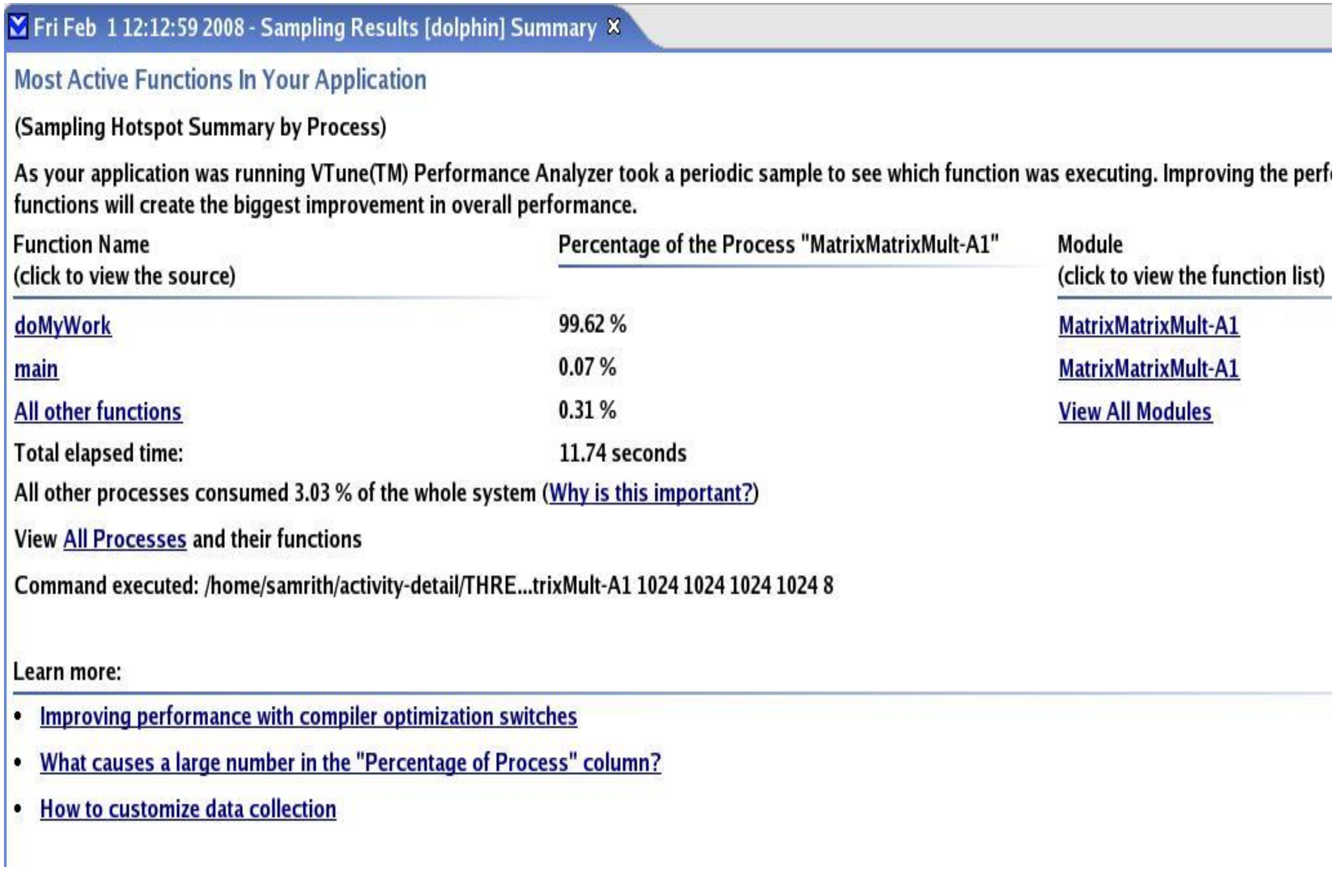
| Name | CPU_CLK_UNH... | | | INST_RETIRED.... | | | Clock per Inst... | CPU_CLK_UNHALTE... | | | INS... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | T... | P... | P... | Total | P... | P... | | Total | Pro... | Pr... | Total |
| copy_employee | 6 | 6 | 0 | 1 | 1 | 0 | 6.000 | 85.71% | 85.71% | 0.00% | 20.00% |
| main | 1 | 1 | 0 | 4 | 4 | 0 | 0.250 | 14.29% | 14.29% | 0.00% | 80.00% |

# VTune : Case Study : Cont…



```
------------------------------------------------------------------------------
|ID|Short D|Sever|C|Conte|Description                    |1st Acces|2nd Acces|
|  |escript|ity  |o|xt[Be|                               |s[Best]  |s[Best]  |
|  |ion    |Name |u|st]  |                               |         |         |
|  |       |     |n|     |                               |         |         |
|  |       |     |t|     |                               |         |         |
------------------------------------------------------------------------------
|1 |Threadt|Infor|1|Whole|Thread termination at          |"MatrixMa|"MatrixMa|
|  |erminat|matio| |Progr|"MatrixMatrixMult-1.c":169 -   |trixMult-|trixMult-|
|  |ion    |n    | |am 1 |includes stack allocation of   |1.c":169 |1.c":169 |
|  |       |     | |     |10.004 MB and use of 2.98 KB   |         |         |
------------------------------------------------------------------------------
|2 |Threadt|Infor|1|Whole|Thread termination at          |"MatrixMa|"MatrixMa|
|  |erminat|matio| |Progr|"MatrixMatrixMult-1.c":169 -   |trixMult-|trixMult-|
|  |ion    |n    | |am 2 |includes stack allocation of   |1.c":169 |1.c":169 |
|  |       |     | |     |10.004 MB and use of 2.98 KB   |         |         |
------------------------------------------------------------------------------
|3 |Threadt|Infor|1|Whole|Thread termination at          |"MatrixMa|"MatrixMa|
|  |erminat|matio| |Progr|"MatrixMatrixMult-1.c":169 -   |trixMult-|trixMult-|
|  |ion    |n    | |am 3 |includes stack allocation of   |1.c":169 |1.c":169 |
|  |       |     | |     |10.004 MB and use of 2.98 KB   |         |         |
------------------------------------------------------------------------------
|4 |Threadt|Infor|1|Whole|Thread termination at          |"MatrixMa|"MatrixMa|
|  |erminat|matio| |Progr|"MatrixMatrixMult-1.c":169 -   |trixMult-|trixMult-|
|  |ion    |n    | |am 4 |includes stack allocation of   |1.c":169 |1.c":169 |
|  |       |     | |     |10.004 MB and use of 2.98 KB   |         |         |
------------------------------------------------------------------------------
|5 |Threadt|Infor|1|Whole|Thread termination at          |"MatrixMa|"MatrixMa|
|  |erminat|matio| |Progr|"MatrixMatrixMult-1.c":169 -   |trixMult-|trixMult-|
|  |ion    |n    | |am 5 |includes stack allocation of   |1.c":169 |1.c":169 |
|  |       |     | |     |10.004 MB and use of 2.98 KB   |         |         |
```

# VTune : Case Study : Cont…



Fri Feb 1 12:12:59 2008 - Sampling Results [dolphin] Summary

Most Active Functions In Your Application

(Sampling Hotspot Summary by Process)

As your application was running VTune(TM) Performance Analyzer took a periodic sample to see which function was executing. Improving the perf functions will create the biggest improvement in overall performance.

| Function Name<br>(click to view the source) | Percentage of the Process "MatrixMatrixMult-A1" | Module<br>(click to view the function list) |
|---|---|---|
| doMyWork | 99.62 % | MatrixMatrixMult-A1 |
| main | 0.07 % | MatrixMatrixMult-A1 |
| All other functions | 0.31 % | View All Modules |
| Total elapsed time: | 11.74 seconds | |

All other processes consumed 3.03 % of the whole system (Why is this important?)

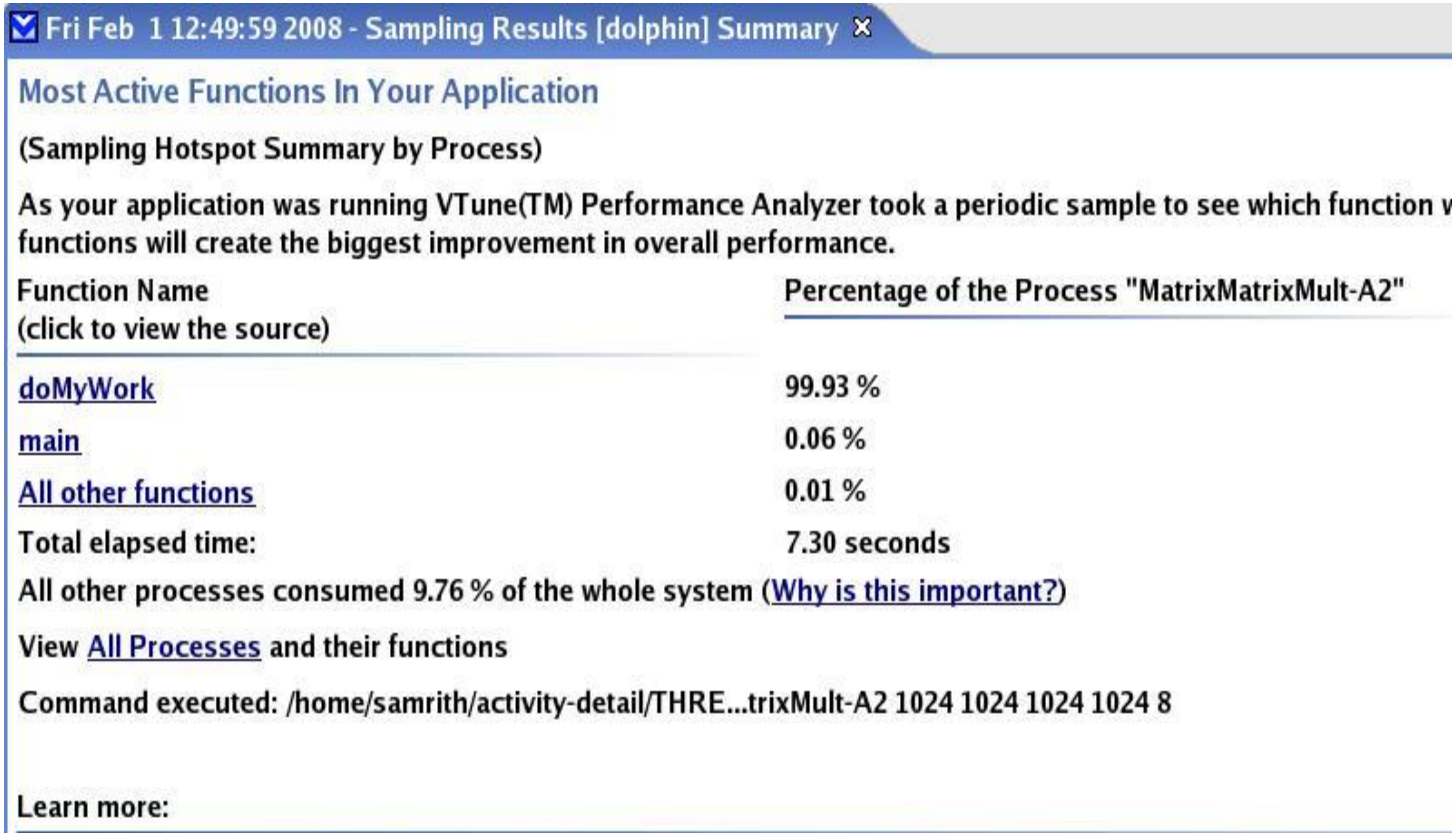
View All Processes and their functions

Command executed: /home/samrith/activity-detail/THRE...trixMult-A1 1024 1024 1024 1024 8

Learn more:

- Improving performance with compiler optimization switches
- What causes a large number in the "Percentage of Process" column?
- How to customize data collection

# VTune : Case Study : Cont…



Fri Feb 1 12:49:59 2008 - Sampling Results [dolphin] Summary

Most Active Functions In Your Application

(Sampling Hotspot Summary by Process)

As your application was running VTune(TM) Performance Analyzer took a periodic sample to see which function w
functions will create the biggest improvement in overall performance.

| Function Name<br>(click to view the source) | Percentage of the Process "MatrixMatrixMult-A2" |
|---|---|
| doMyWork | 99.93 % |
| main | 0.06 % |
| All other functions | 0.01 % |
| Total elapsed time: | 7.30 seconds |

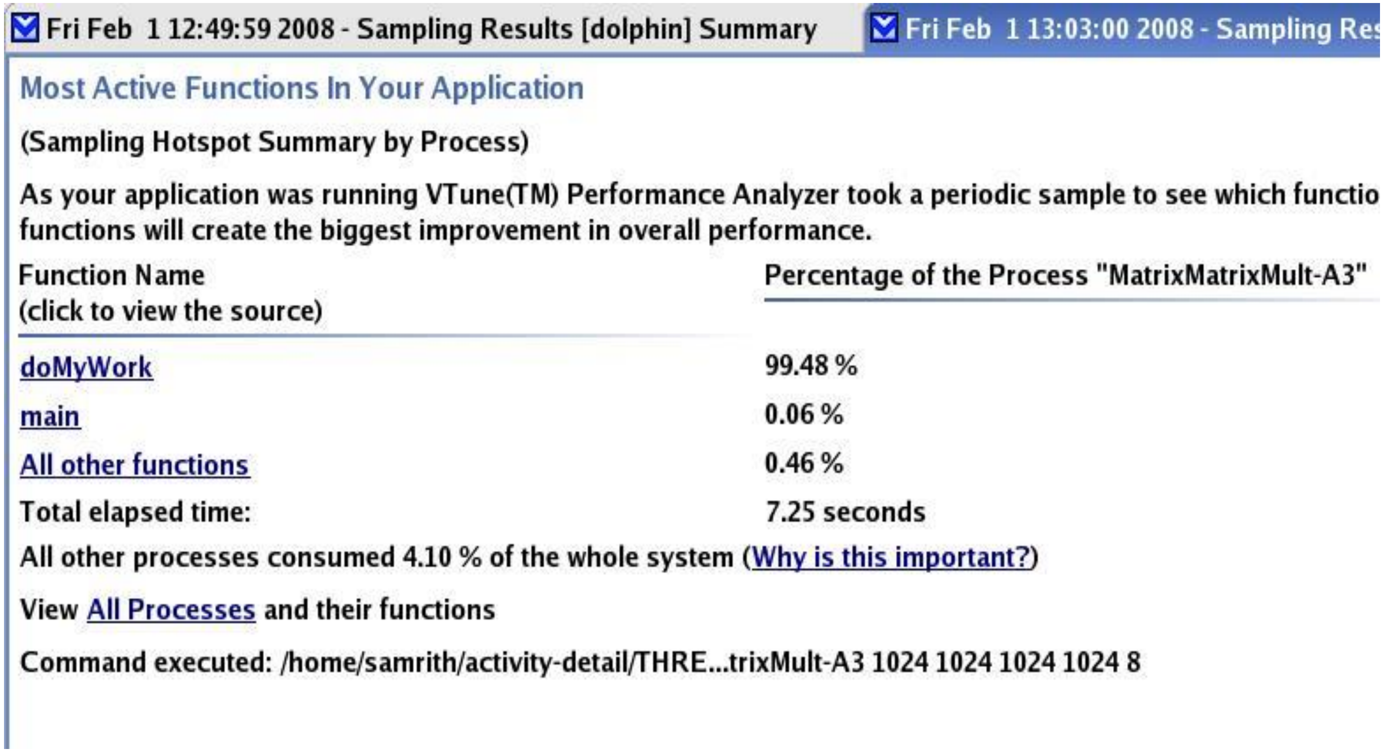
All other processes consumed 9.76 % of the whole system (Why is this important?)

View All Processes and their functions

Command executed: /home/samrith/activity-detail/THRE...trixMult-A2 1024 1024 1024 1024 8

Learn more:

# VTune : Case Study : Cont…



Fri Feb 1 12:49:59 2008 - Sampling Results [dolphin] Summary
Fri Feb 1 13:03:00 2008 - Sampling Res

Most Active Functions In Your Application

(Sampling Hotspot Summary by Process)

As your application was running VTune(TM) Performance Analyzer took a periodic sample to see which functio functions will create the biggest improvement in overall performance.

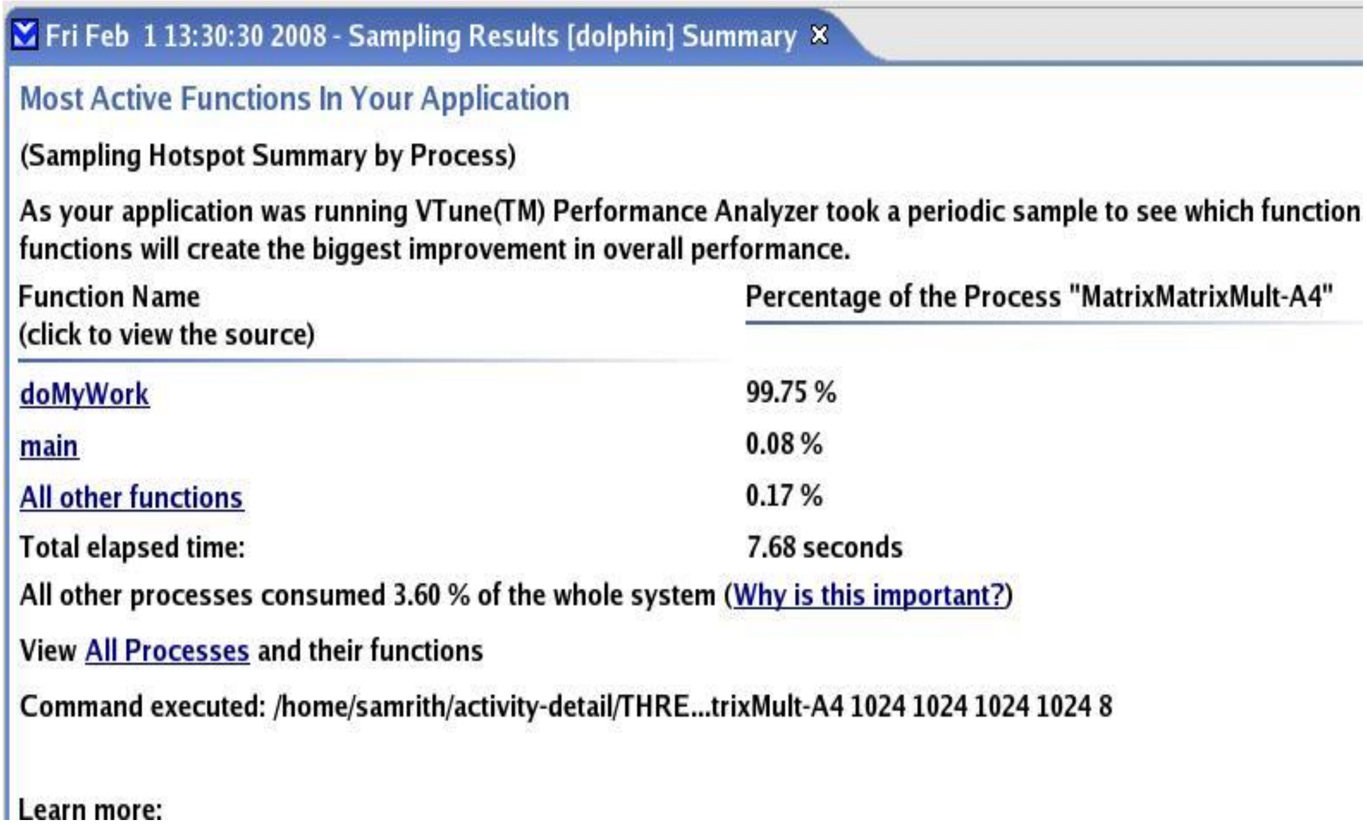
| Function Name<br>(click to view the source) | Percentage of the Process "MatrixMatrixMult-A3" |
|---|---|
| doMyWork | 99.48 % |
| main | 0.06 % |
| All other functions | 0.46 % |
| Total elapsed time: | 7.25 seconds |

All other processes consumed 4.10 % of the whole system (Why is this important?)

View All Processes and their functions

Command executed: /home/samrith/activity-detail/THRE...trixMult-A3 1024 1024 1024 1024 8

# VTune : Case Study : Cont…



Fri Feb 1 13:30:30 2008 - Sampling Results [dolphin] Summary

Most Active Functions In Your Application

(Sampling Hotspot Summary by Process)

As your application was running VTune(TM) Performance Analyzer took a periodic sample to see which function functions will create the biggest improvement in overall performance.

| Function Name<br>(click to view the source) | Percentage of the Process "MatrixMatrixMult-A4" |
|---|---|
| doMyWork | 99.75 % |
| main | 0.08 % |
| All other functions | 0.17 % |
| Total elapsed time: | 7.68 seconds |

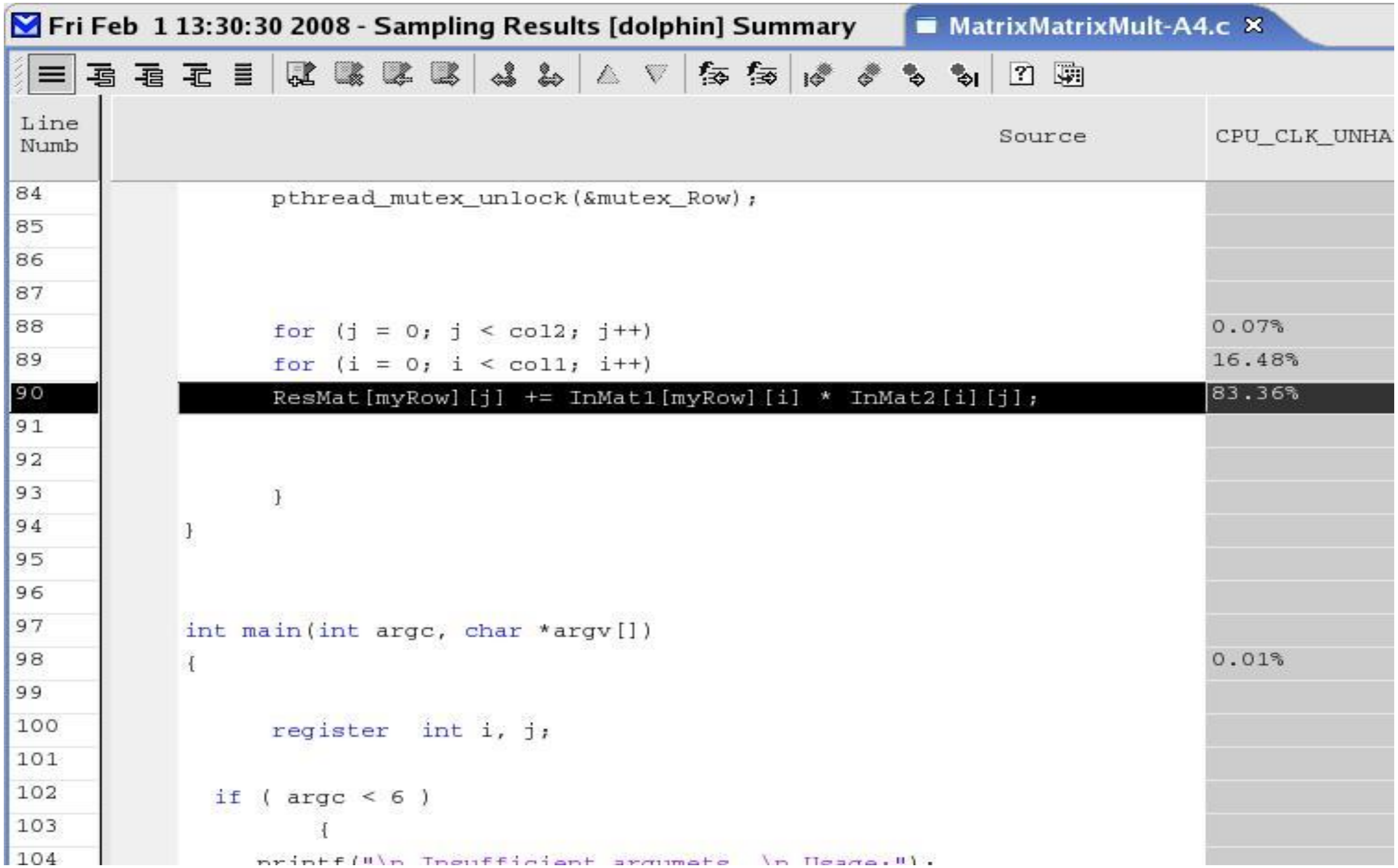
All other processes consumed 3.60 % of the whole system (Why is this important?)

View All Processes and their functions

Command executed: /home/samrith/activity-detail/THRE...trixMult-A4 1024 1024 1024 1024 8

Learn more:

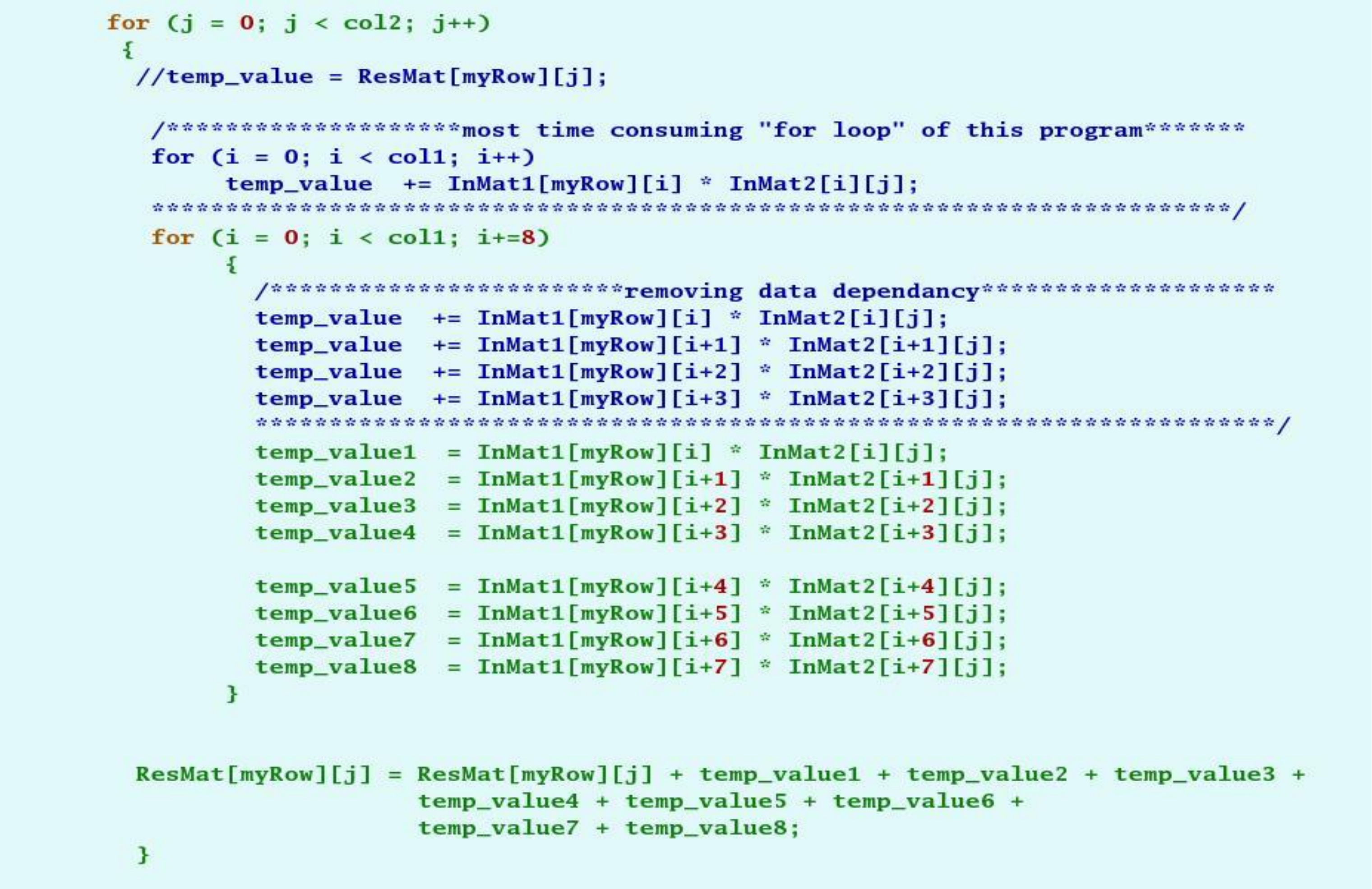
# VTune : Case Study : Cont…



Fri Feb  1 13:30:30 2008 - Sampling Results [dolphin] Summary
MatrixMatrixMult-A4.c
Line Numb
Source
CPU_CLK_UNHA

```
        pthread_mutex_unlock(&mutex_Row);



        for (j = 0; j < col2; j++)                                    0.07%
        for (i = 0; i < col1; i++)                                   16.48%
        ResMat[myRow][j] += InMat1[myRow][i] * InMat2[i][j];         83.36%


        }
}


int main(int argc, char *argv[])
{                                                                      0.01%

        register  int i, j;

   if ( argc < 6 )
            {
      printf("\n Insufficient argumets  \n Usage:");
```

# VTune : Case Study : So What is Solution



```
for (j = 0; j < col2; j++)
 {
  //temp_value = ResMat[myRow][j];

   /********************most time consuming "for loop" of this program*******
   for (i = 0; i < col1; i++)
        temp_value  += InMat1[myRow][i] * InMat2[i][j];
   ************************************************************************/
   for (i = 0; i < col1; i+=8)
        {
          /************************removing data dependancy********************
          temp_value  += InMat1[myRow][i] * InMat2[i][j];
          temp_value  += InMat1[myRow][i+1] * InMat2[i+1][j];
          temp_value  += InMat1[myRow][i+2] * InMat2[i+2][j];
          temp_value  += InMat1[myRow][i+3] * InMat2[i+3][j];
          ********************************************************************/
          temp_value1  = InMat1[myRow][i] * InMat2[i][j];
          temp_value2  = InMat1[myRow][i+1] * InMat2[i+1][j];
          temp_value3  = InMat1[myRow][i+2] * InMat2[i+2][j];
          temp_value4  = InMat1[myRow][i+3] * InMat2[i+3][j];

          temp_value5  = InMat1[myRow][i+4] * InMat2[i+4][j];
          temp_value6  = InMat1[myRow][i+5] * InMat2[i+5][j];
          temp_value7  = InMat1[myRow][i+6] * InMat2[i+6][j];
          temp_value8  = InMat1[myRow][i+7] * InMat2[i+7][j];
        }


  ResMat[myRow][j] = ResMat[myRow][j] + temp_value1 + temp_value2 + temp_value3 +
                     temp_value4 + temp_value5 + temp_value6 +
                     temp_value7 + temp_value8;
  }
```

# References

1. Andrews, Grogory R. **(2000),** Foundations of Multithreaded, Parallel, and Distributed Programming, Boston, MA : Addison-Wesley

2. Butenhof, David R **(1997),** Programming with POSIX Threads , Boston, MA : Addison Wesley Professional

3. Culler, David E., Jaswinder Pal Singh **(1999),** Parallel Computer Architecture - A Hardware/Software Approach , San Francsico, CA : Morgan Kaufmann

4. Grama Ananth, Anshul Gupts, George Karypis and Vipin Kumar **(2003),** Introduction to Parallel computing, Boston, MA : Addison-Wesley

5. Intel Corporation, **(2003),** Intel Hyper-Threading Technology, Technical User's Guide, Santa Clara CA : Intel Corporation Available at : http://www.intel.com

6. Shameem Akhter, Jason Roberts **(April 2006),** Multi-Core Programming - Increasing Performance through Software Multi-threading , Intel PRESS, Intel Corporation,

7. Bradford Nichols, Dick Buttlar and Jacqueline Proulx Farrell **(1996),** Pthread Programming O'Reilly and Associates, Newton, MA 02164,

8. James Reinders, Intel Threading Building Blocks – (**2007**) , O'REILLY series

9. Laurence T Yang & Minyi Guo (Editors), (**2006**) *High Performance Computing - Paradigm and Infrastructure* Wiley Series on Parallel and Distributed computing, Albert Y. Zomaya, Series Editor

10. Intel Threading Methodology ; Principles and Practices Version 2.0 copy right **(March 2003),** Intel Corporation

# References

11. William Gropp, Ewing Lusk, Rajeev Thakur **(1999),** Using MPI-2, Advanced Features of the Message-Passing Interface, The MIT Press..

12. Pacheco S. Peter, **(1992),** Parallel Programming with MPI, , University of Sanfrancisco, Morgan Kaufman Publishers, Inc., Sanfrancisco, California

13. Kai Hwang, Zhiwei Xu, (**1998**), Scalable Parallel Computing (Technology Architecture Programming), McGraw Hill New York.

14. Michael J. Quinn (**2004**), Parallel Programming in C with MPI and OpenMP McGraw-Hill International Editions, Computer Science Series, McGraw-Hill, Inc. Newyork

15. Andrews, Grogory R. **(2000),** Foundations of Multithreaded, Parallel, and Distributed Progrmaming, Boston, MA : Addison-Wesley

16. SunSoft. Solaris multithreaded programming guide. SunSoft Press, Mountainview, CA, **(1996),** Zomaya, editor. Parallel and Distributed Computing Handbook. McGraw-Hill,

17. Chandra, Rohit, Leonardo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald, and Ramesh Menon, **(2001)**,Parallel Programming in OpenMP San Fracncisco Moraan Kaufmann

18. S.Kieriman, D.Shah, and B.Smaalders **(1995),** Programming with Threads, SunSoft Press, Mountainview, CA. 1995

19. Mattson Tim, **(2002),** Nuts and Bolts of multi-threaded Programming Santa Clara, CA : Intel Corporation, Available at : http://www.intel.com

20. I. Foster **(1995,** Designing and Building Parallel Programs ; Concepts and tools for Parallel Software Engineering, Addison-Wesley (1995)

21. J.Dongarra, I.S. Duff, D. Sorensen, and H.V.Vorst **(1999),** Numerical Linear Algebra for High Performance Computers (Software, Environments, Tools) SIAM, 1999

# References

22. OpenMP C and C++ Application Program Interface, Version 1.0". **(1998),** OpenMP Architecture Review Board. October 1998

23. D. A. Lewine. *Posix Programmer's Guide:* **(1991),** Writing Portable Unix Programs with the Posix. 1 Standard. O'Reilly & Associates, 1991

24. Emery D. Berger, Kathryn S McKinley, Robert D Blumofe, Paul R.Wilson, *Hoard : A Scalable Memory Allocator for Multi-threaded Applications* ; The Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX). Cambridge, MA, November (**2000)**. Web site URL : http://www.hoard.org/

25. Marc Snir, Steve Otto, Steyen Huss-Lederman, David Walker and Jack Dongarra, (**1998**) *MPI-The Complete Reference: Volume 1, The MPI Core, second edition* [MCMPI-07].

26. William Gropp, Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir, and Marc Snir (**1998**) *MPI-The Complete Reference: Volume 2, The MPI-2 Extensions*

27. A. Zomaya, editor. Parallel and Distributed Computing Handbook. McGraw-Hill, **(1996)**

28. OpenMP C and C++ Application Program Interface, Version 2.5 (**May 2005**)”, From the OpenMP web site, URL **: http://www.openmp.org/**

29. Stokes, Jon 2002, Introduction to Multithreading, Super-threading and Hyperthreading; *Ars Technica*, The Art of Technology, October **(2002)**

30. Andrews Gregory R. 2000, Foundations of Multi-threaded, Parallel and Distributed Programming, Boston MA : Addison – Wesley (**2000)**

31. Deborah T. Marr , Frank Binns, David L. Hill, Glenn Hinton, David A Koufaty, J . Alan Miller, Michael Upton, “Hyperthreading, Technology Architecture and Microarchitecture, Intel (**2000)**

# Thank You

*Any questions ?*