Module 12: View

Lecture 23: Intel Compilers and Threading Tool

The Lecture Contains:

Previous Next

Module 12: View

Lecture 23: Intel Compilers and Threading Tool

Intel Thread Checker Output View

```
[majeti deepak@Majeti programs]$ gcc -g -pthread -O0 count.c -o count
[majeti deepak@Majeti programs]$ /opt/intel/itt/tcheck/bin/32/./tcheck_cl count
Intel(R) Thread Checker 3.1 command line instrumentation driver (27583)
Copyright (c) 2007 Intel Corporation. All rights reserved.
Building project
Instrumenting
 25% count           ( All Functions ):..
 75% libc-2.8.so     ( Minimal ):....
100% libpthread-2.8.so ( Minimal ):..

Running:  /media/STUDIES/sem-2/parallel_summer course/inteltools_ppt/programs/count


Application finished
```

ID	Short Description	Severity	Count	Context[Best]	Description	1st Access[Best]	2nd Access[Best]
1	Read -> Write data race	Error	1	"count.c":5	Memory write at "count.c":6 conflicts with a prior memory read at "count.c":6 (anti dependence)	"count.c":6	"count.c":6
2	Write -> Read data race	Error	1	"count.c":5	Memory read at "count.c":6 conflicts with a prior memory write at "count.c":6 (flow dependence)	"count.c":6	"count.c":6
3	Write -> Write data-race	Error	1	"count.c":5	Memory write at "count.c":6 conflicts with a prior memory write at "count.c":6 (output dependence)	"count.c":6	"count.c":6
4	Thread termination	Information	1	WholeProgram1	Thread termination at "count.c":17 includes stack allocation of 10.004 MB and use of 3.918 KB	"count.c":17	"count.c":17
5	Thread termination	Information	1	WholeProgram2	Thread termination at "count.c":17 includes stack allocation of 10.004 MB and use of 3.910 KB	"count.c":17	"count.c":17

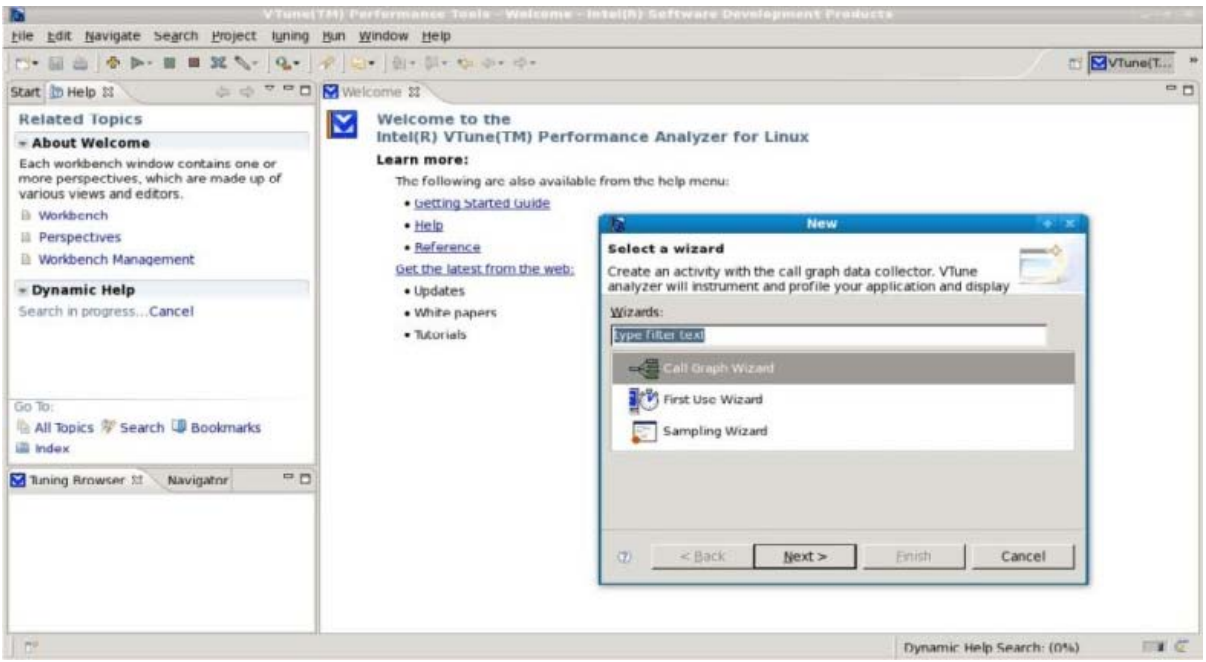
Intel Vtune Performance Analyzer

- Low Overhead Sampling Proling (sampling wizard).
- Call Graph Proling (call graph wizard).
- Counter Monitor
- Intel Tuning Assistant.
- New Events for Tuning Multicore Processors.

Previous Next

Module 12: View
Lecture 23: Intel Compilers and Threading Tool

View



Example Code

funcs.c

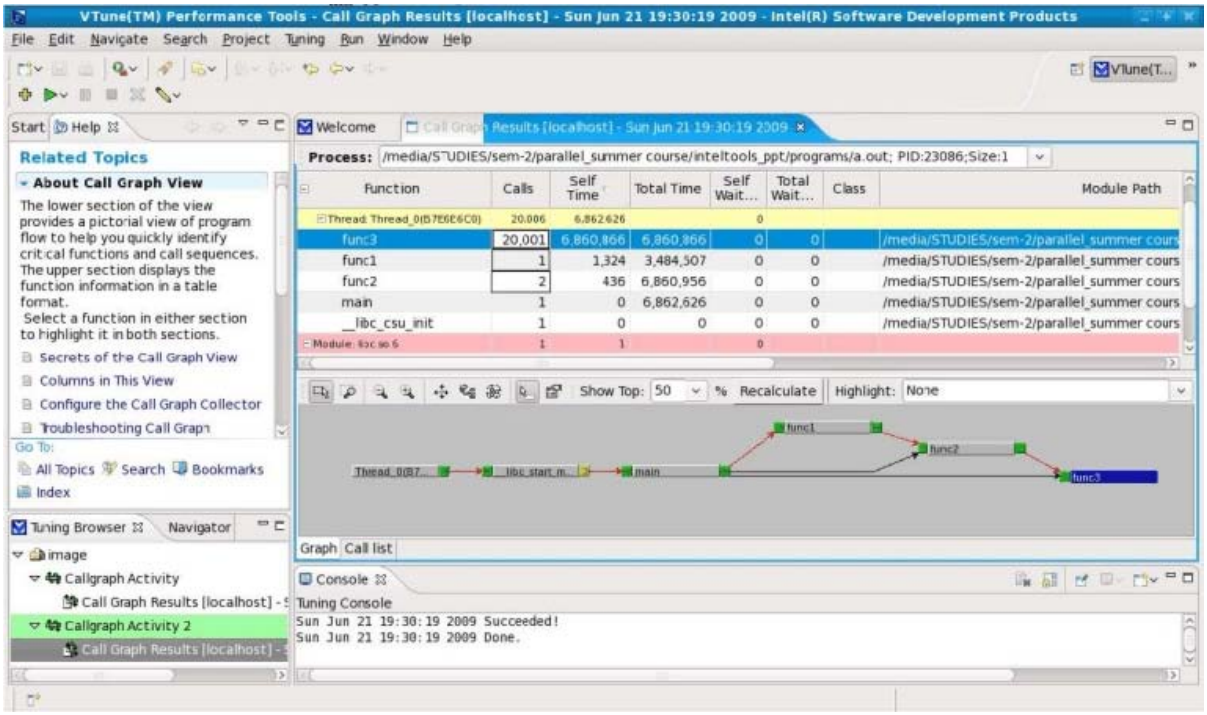
```
void func3()
{
        int k;
        for(k=0;k<100000;k++);
}
void func2()
{
        int j;
        for(j=0;j<10000;j++)
                func3();
}
void func1()
{
        int i;
        func2();
        for(i=0;i<100000;i++);
}
int main()
{
        func1();
        func2();
        func3();
}
```

Previous Next

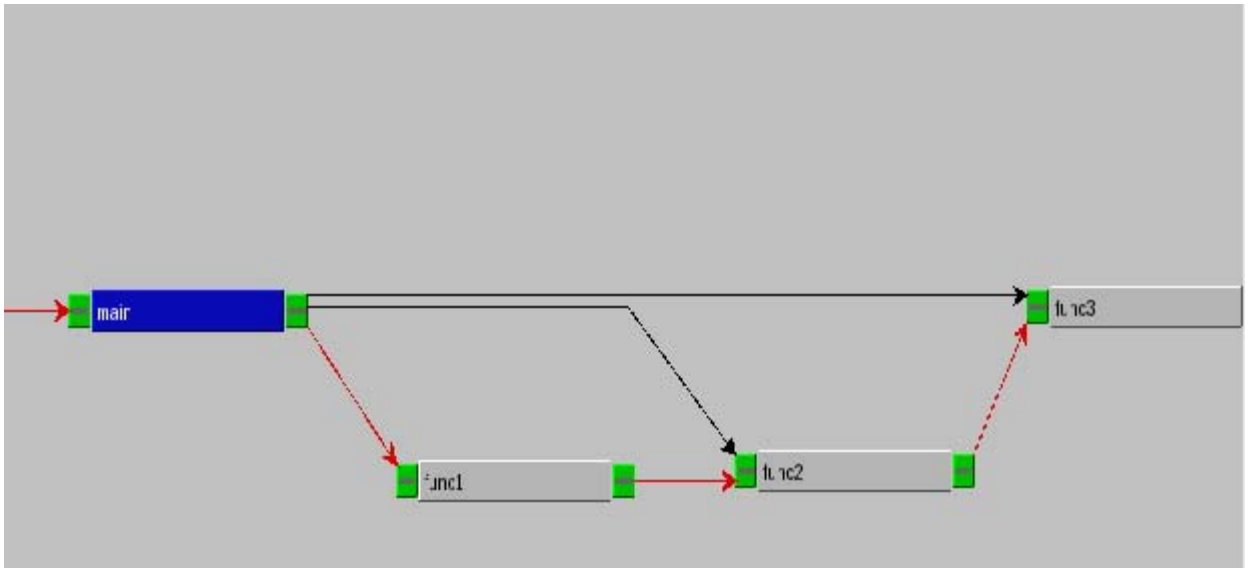
Module 12: View

Lecture 23: Intel Compilers and Threading Tool

Call Graph Wizard View



Call Graph View

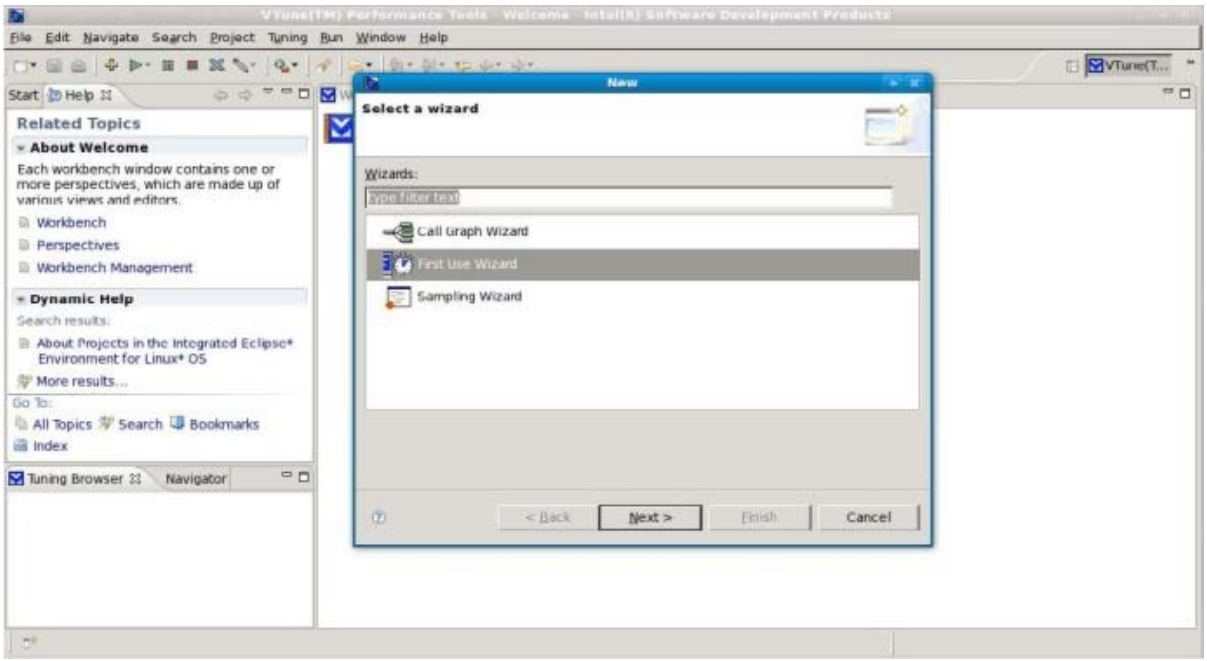


Previous Next

Call Statistics View

Function	Calls	Self Time	Total Time	Self Wait...	Total Wait...
Module: a.out	20,005	6,862,626		0	
Thread: Thread_0(B7E6E6C0)	20,005	6,862,626		0	
func3	20,001	6,860,866	6,860,866	0	0
func1	1	1,324	3,484,507	0	0
func2	2	436	6,850,956	0	0
main	1	0	6,862,626	0	0

First Use Wizard View



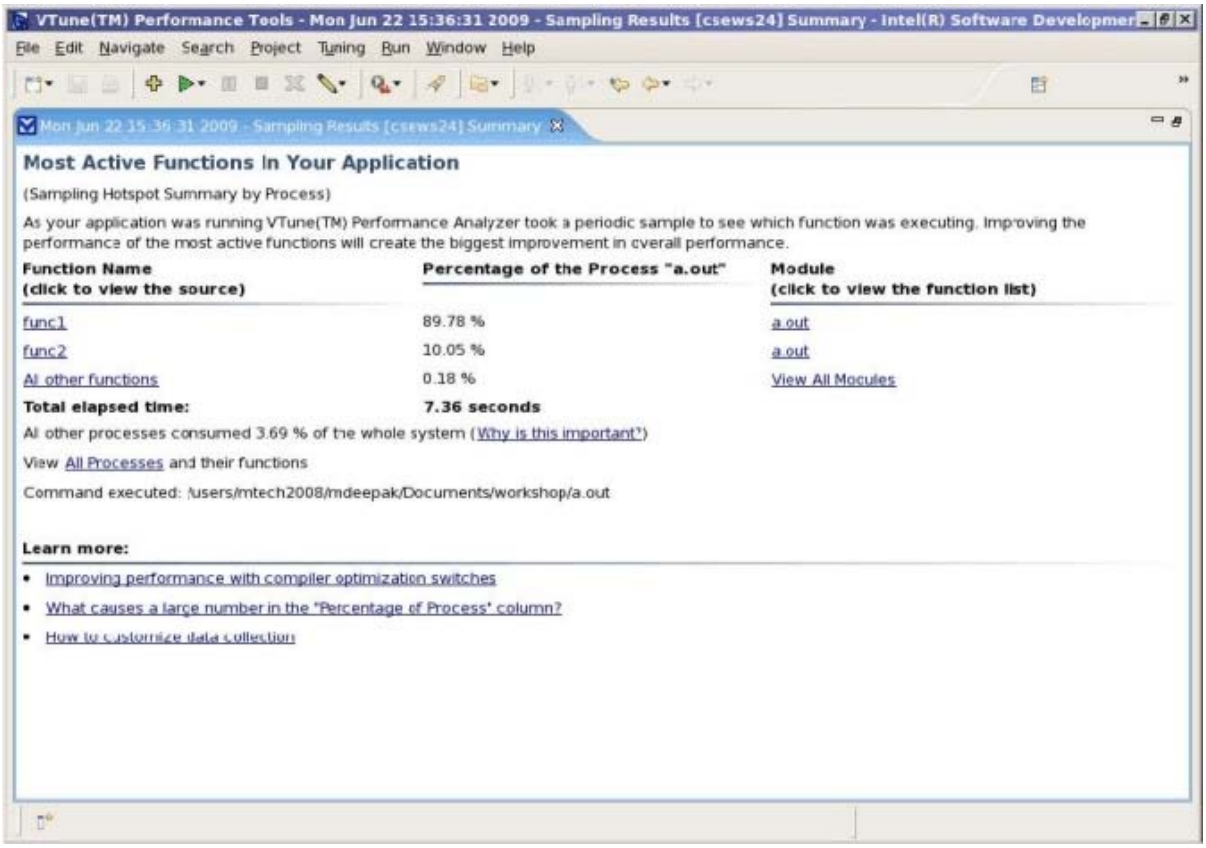
Previous Next

Example Code

Work.c

```
int array[10001];
void func2()
{
        int i,j;
        for(i=1;i<10000;i++)
                for(j=1;j<10000;j++)
                        array[j]=array[j-1]+array[j]+array[j+1];
}
void func1()
{
        int i,j,k;
        for(i=1;i<10;i++)
                for(j=1;j<10000;j++)
                        for(k=1;k<10000;k++)
                                array[k]=array[k-1]+array[k]+array[k+1];
}
int main()
{
        func1();
        func2();
}
```

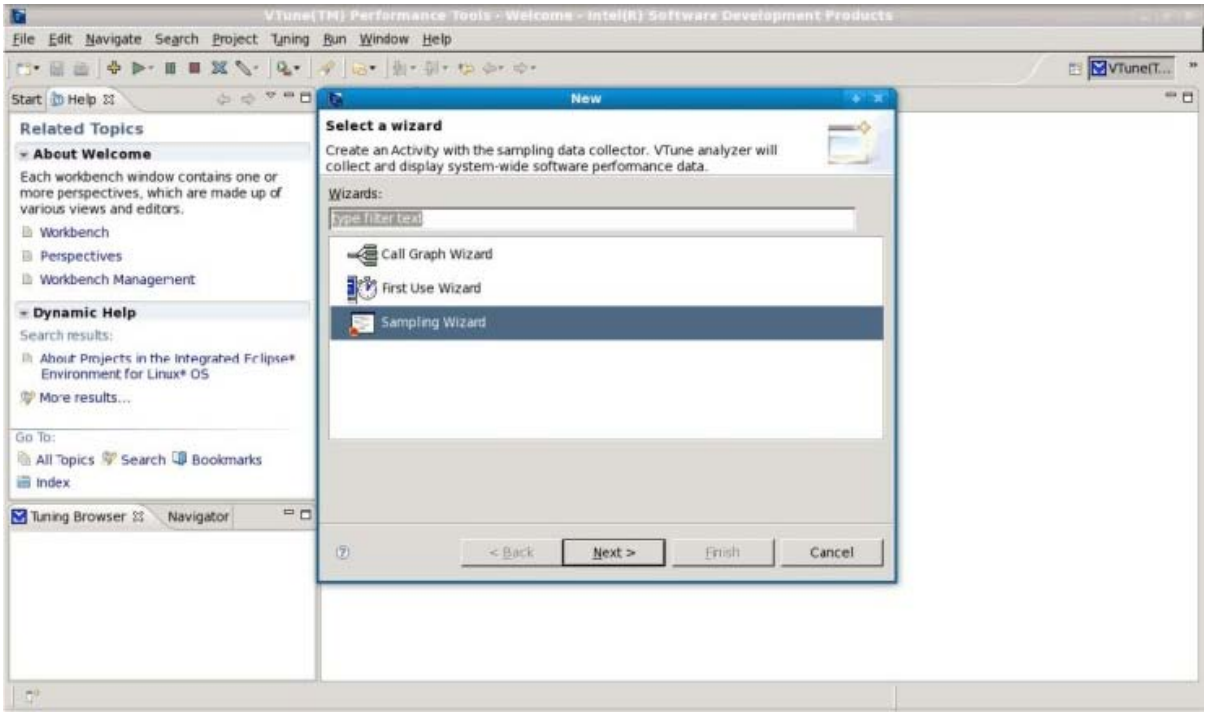
First Use Wizard Output View

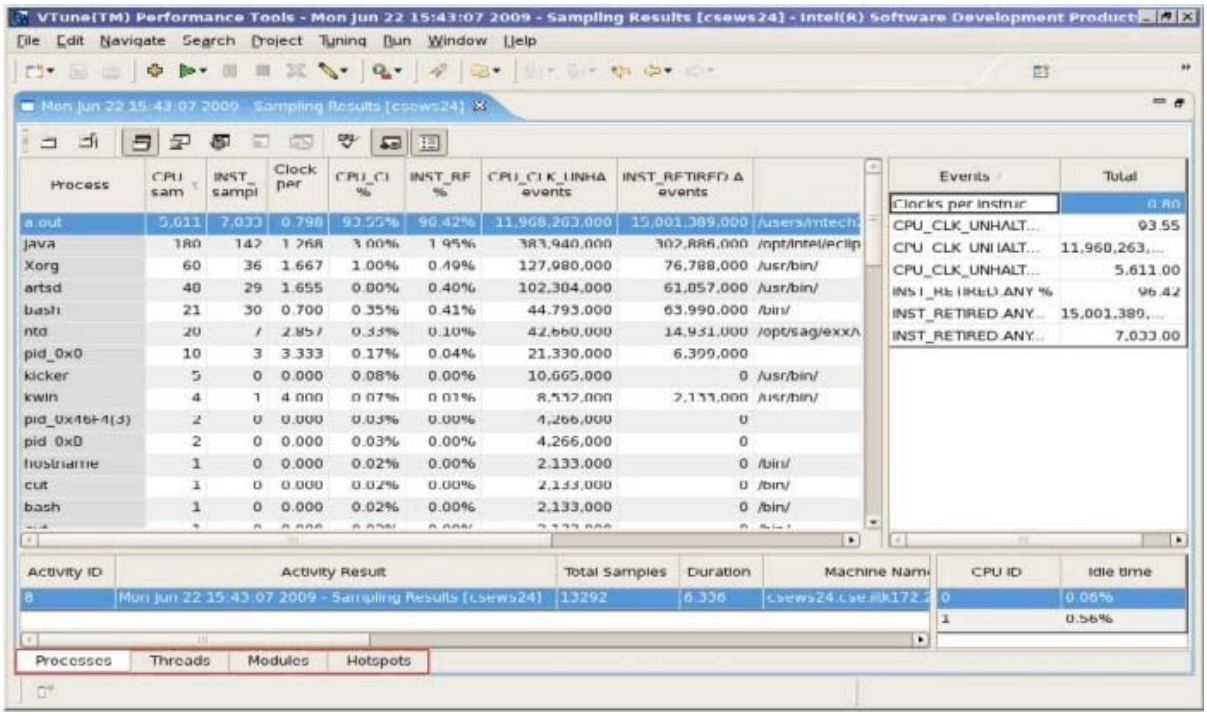


Module 12: View

Lecture 23: Intel Compilers and Threading Tool

Sampling Wizard View



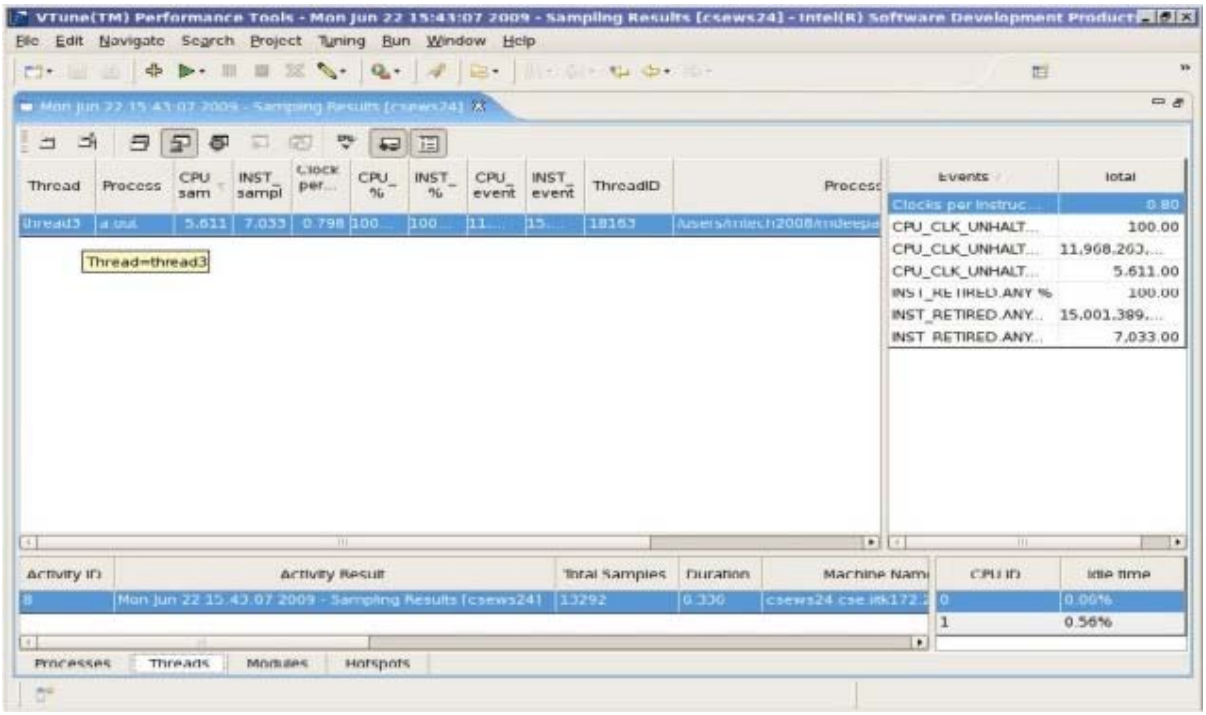


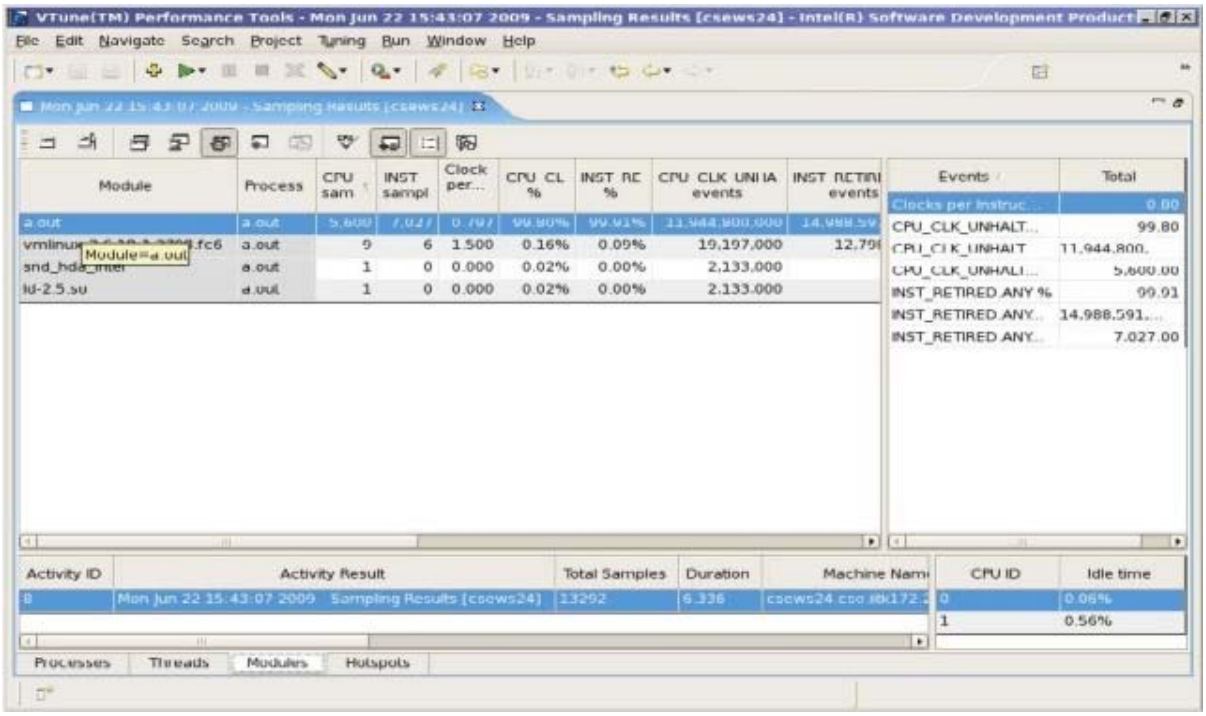
Previous Next

Module 12: View

Lecture 23: Intel Compilers and Threading Tool

Sampling Wizard View



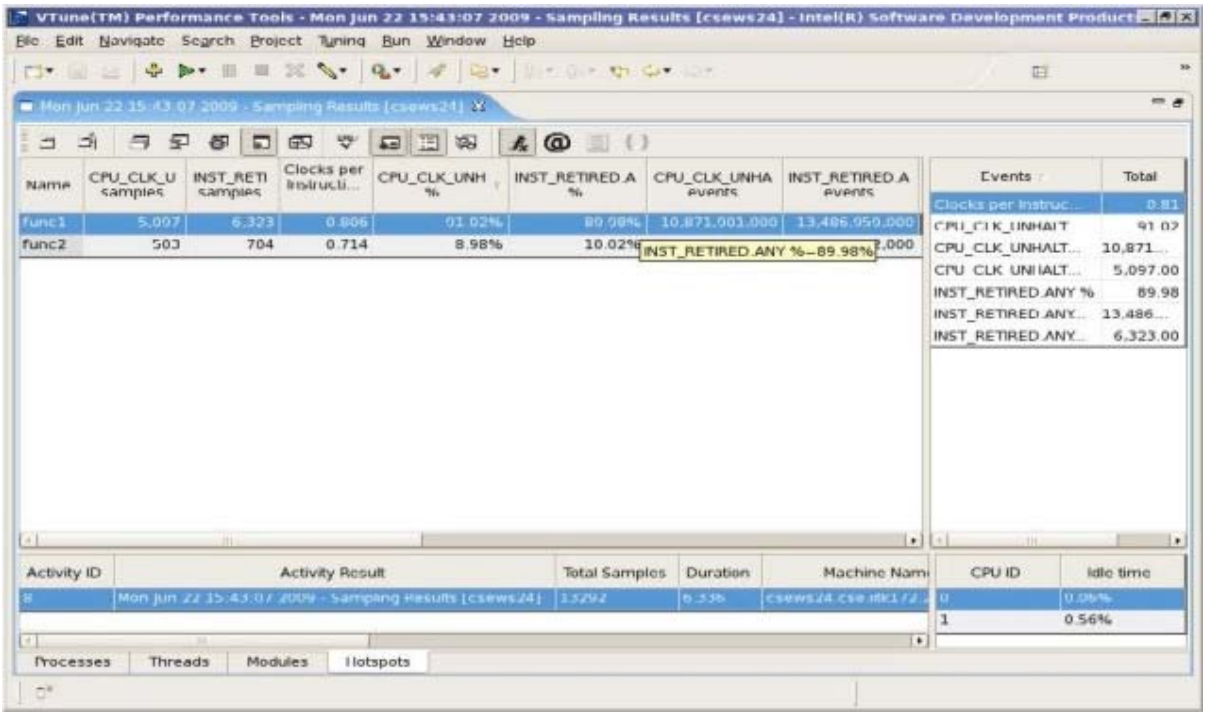


Previous Next

Module 12: View

Lecture 23: Intel Compilers and Threading Tool

Sampling Wizard View



Intel Vtune Tips

- The time displayed by the First Use Wizard is the real time (It also includes context switches).If there are other programs running in parallel, the result may be inaccurate.
- Use the Sampling Wizard for the system time (Actual time taken).
- Always take results for a series of runs and observe the median.

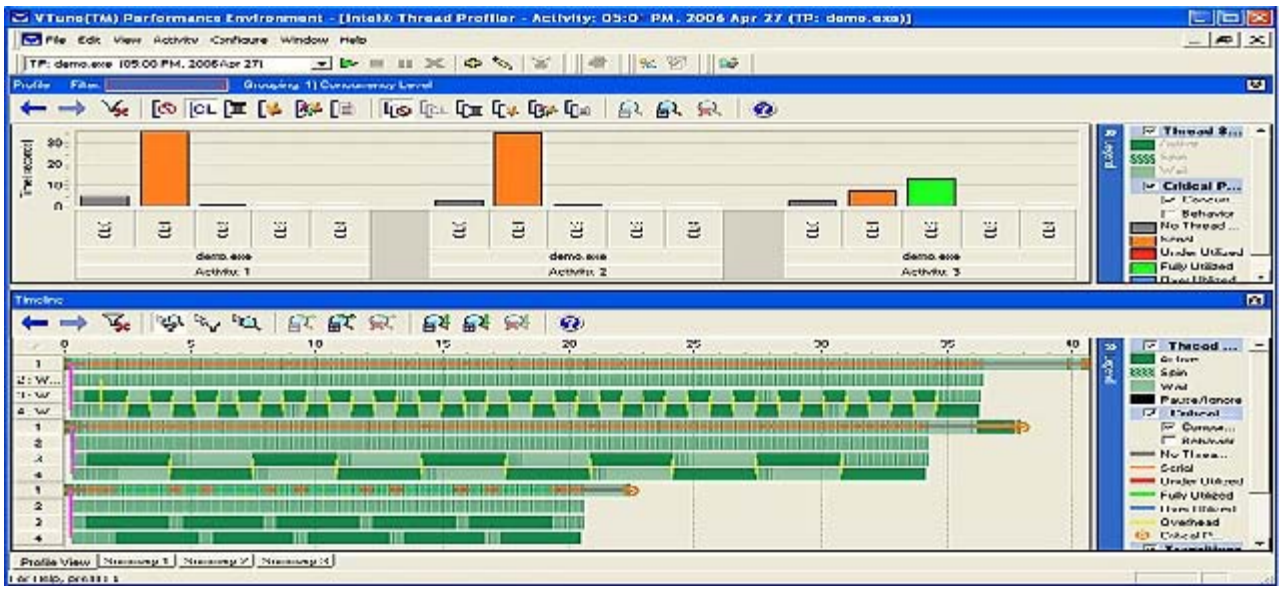
Intel Thread Proler

- Visualize Threaded Application Behavior.
- Identify Parallel Performance Issues.
- Find the best sections of code to optimize for sequential performance and for threaded performance.
- Check scalability by varying the number of processors/threads.
- GUI available only for Windows.
- Text mode patch available for Linux

Previous Next

Module 12: View
Lecture 23: Intel Compilers and Threading Tool

View



Example Code

```
#include<omp.h>

int main(){

        int x[1000],i,j;
        int a[1000];

#pragma omp parallel private(i,j)
#pragma omp sections
{
#pragma omp section
        for(i=0;i<100000;i++){
                for(j=0;j<10000;j++){
                        x[i%1000]=x[j%1000];
                }
        }
#pragma omp section
        for(i=0;i<100;i++){
                for(j=0;j<100;j++){
                        a[i%1000]=a[j%1000];
                }
        }
}
}
```

Previous Next

Intel Proler View on Linux

```
[majeti_deepak@Majeti programs]$ gcc -O0 -fopenmp section.c -o section
[majeti_deepak@Majeti programs]$ /opt/intel/itt/tprofile/bin/32/tprofile_cl  section
Building project
Instrumenting
 16% section         ( API Imports ):..
 50% libc-2.8.so     ( Minimal ):....
 66% libgomp.so.1.0.0 ( API Imports ):..
 83% libpthread-2.8.so ( Minimal ):..
100% librt-2.8.so    ( Minimal ):..

Running:  /media/STUDIES/scm-2/parallel_summer course/inteltools_ppt/programs/section


Application finished

Intel(R) Thread Profiler  3.1  Summary Report
application:          /media/STUDIES/scm-2/parallel_summer course/inteltools_ppt/programs/section
collection:           Mon Jun 22 11:54:36 2009
runtime:              10.6452s
# of processors:      2
# of threads:         2
# of waits:           1
wait frequency:       0.0939393
average concurrency:  1.99975
Concurrency:
   0 [..........] 0%       0
   1 [..........] 0.00875% 0.000931005
   2 [##########] 100%     10.6424


For further analysis, the output in the threadprofiler directory
can be viewed in the GUI (available on Microsoft* Windows*)

  1. Copy the contents of the threadprofiler directory to the Windows
     machine, or use a network-mounted drive.
  2. From the Intel(R) Thread Profiler GUI, use File->Open File and select
     the tprofile.<pid>.tp file.

* Other names and brands are the property of their respective owners.
```

Previous Next

End of Lecture

Questions?

References

- Intel ® Thread Checker for Linux
  http://software.intel.com/en-us/articles/
  intel-thread-checker-documentation
- VTune(TM) Performance Analyzer for Linux
  http://software.intel.com/file/6734
- Intel ® Software Technical Documentation
  http://software.intel.com/en-us/articles/
  intel-software-technical-documentation/
- Intel tools tutorial
  http://docs.notur.no/uit/archive/HPCiA07/hpcia07-documents/
  intel-tools-tutorial
- M. Herily, N. Shavit
  The Art of Multiprocessor Programming
- T. G. Mattson,B. A. Sanders, B. L. Masingill
  Patterns for Parallel Programming

Previous Next