

The Lecture Contains:

- ☰ Intel Compilers and Threading Tool
- ☰ Intel Vtune Performance Analyzer
- ☰ View
- ☰ Example Code
- ☰ Call Graph Wizard View
- ☰ Call Graph View
- ☰ Call Statistics View
- ☰ First Use Wizard View
- ☰ Example Code
- ☰ First Use Wizard Output View
- ☰ Sampling Wizard View
- ☰ Intel Vtune Tips
- ☰ Intel Thread Proler

◀ 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

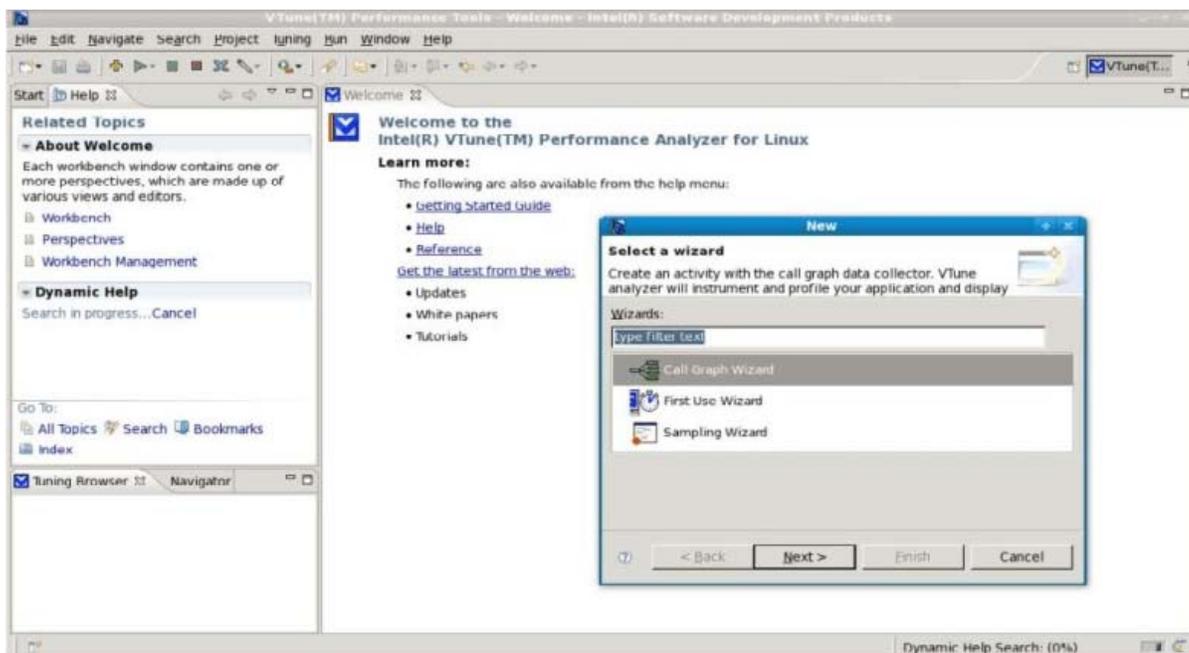
ID	Short Description	Severity	Context	Description	1st Access	2nd Access
1	Read -> Write data-race	Error	1	"count.c":5 Memory write at "count.c":6 conflicts with a prior memory read at "count.c":6 (anti dependence)	"count.c":6	"count.c":6
2	Write -> Read data-race	Error	1	"count.c":5 Memory read at "count.c":6 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	Whole program 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	2	Whole program 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

Intel Vtune Performance Analyzer

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

◀ Previous Next ▶

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 ();
}

```

Module 12: View

Lecture 23: Intel Compilers and Threading Tool

Call Graph Wizard View

Process: /media/STUDIES/sem-2/parallel_summer course/inteltools_ppt/programs/a.out; PID:23086;Size:1

Function	Calls	Self Time	Total Time	Self Wait...	Total Wait...	Class	Module Path
Thread_Thread_0(B7E6E6C0)	20,006	6,862,626		0			
func3	20,001	6,860,866	6,860,866	0	0		/media/STUDIES/sem-2/parallel_summer cours
func1	1	1,324	3,484,507	0	0		/media/STUDIES/sem-2/parallel_summer cours
func2	2	436	6,860,956	0	0		/media/STUDIES/sem-2/parallel_summer cours
main	1	0	6,862,626	0	0		/media/STUDIES/sem-2/parallel_summer cours
_libc_csu_init	1	0	0	0	0		/media/STUDIES/sem-2/parallel_summer cours
Module: libc.so.6	1	1	0				

Graph Call list

```

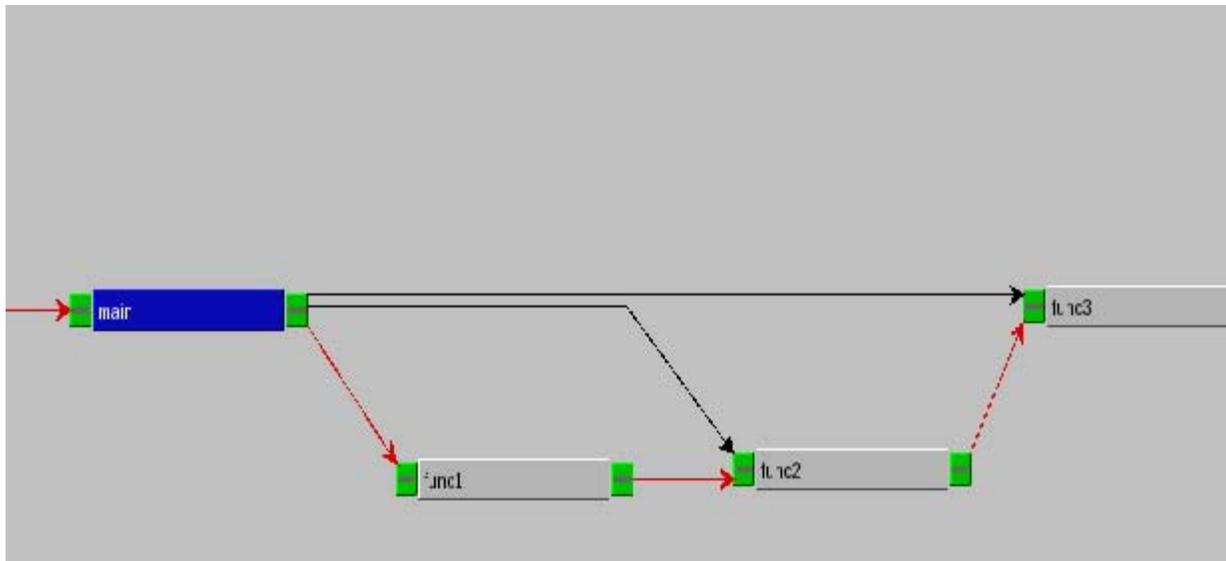
graph LR
    Thread_0(B7) --> libc_start_m
    libc_start_m --> main
    main --> func1
    main --> func2
    func1 --> func3
    func2 --> func3
  
```

Console

```

Tuning Console
Sun Jun 21 19:30:19 2009 Succeeded!
Sun Jun 21 19:30:19 2009 Done.
  
```

Call Graph View



◀ Previous Next ▶

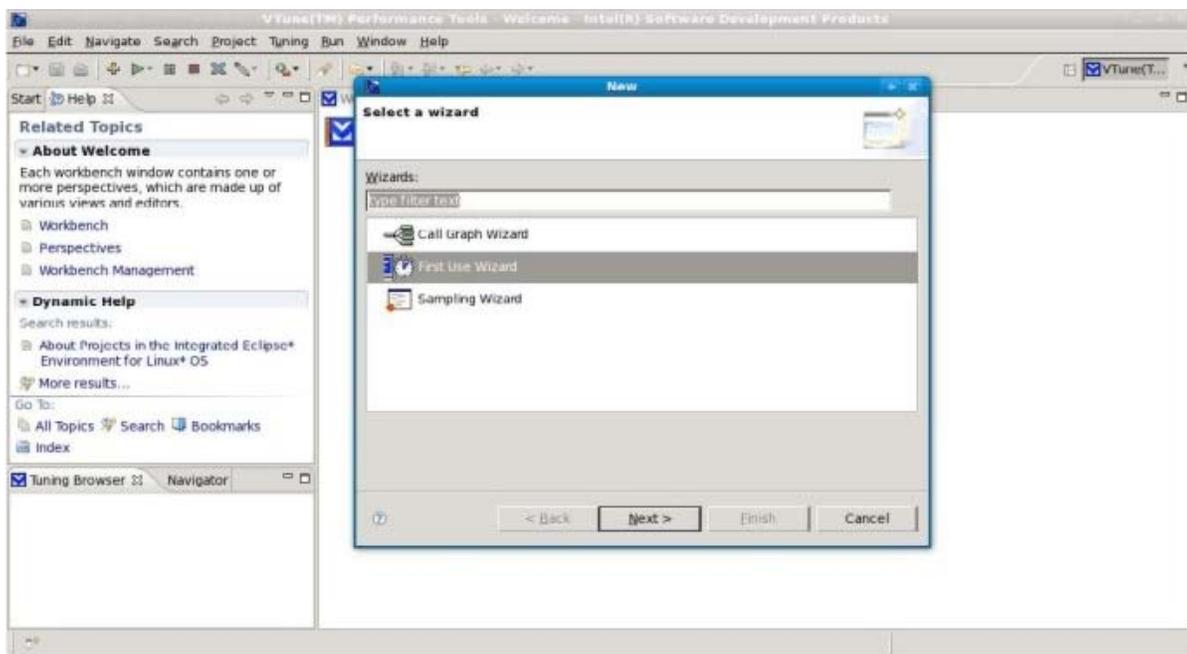
Module 12: View

Lecture 23: Intel Compilers and Threading Tool

Call Statistics View

Function	Calls	Self Time	Total Time	Self Wait...	Total Wait...
Module: a.out	20,035	6,862,626		0	
Thread: Thread_0(B7E6E6C0)	20,035	6,862,626		0	
func3	20,001	6,360,866	6,360,866	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

VTune(TM) Performance Tools - Mon Jun 22 15:36:31 2009 - Sampling Results [csews24] Summary - Intel(R) Software Developer

File Edit Navigate Search Project Tuning Run Window Help

Mon Jun 22 15:36:31 2009 - Sampling Results [csews24] 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 (click to view the source)	Percentage of the Process "a.out"	Module (click to view the function list)
func1	89.78 %	a.out
func2	10.05 %	a.out
All other functions	0.18 %	View All Modules

Total elapsed time: 7.36 seconds

All other processes consumed 3.69 % of the whole system ([Why is this important?](#))

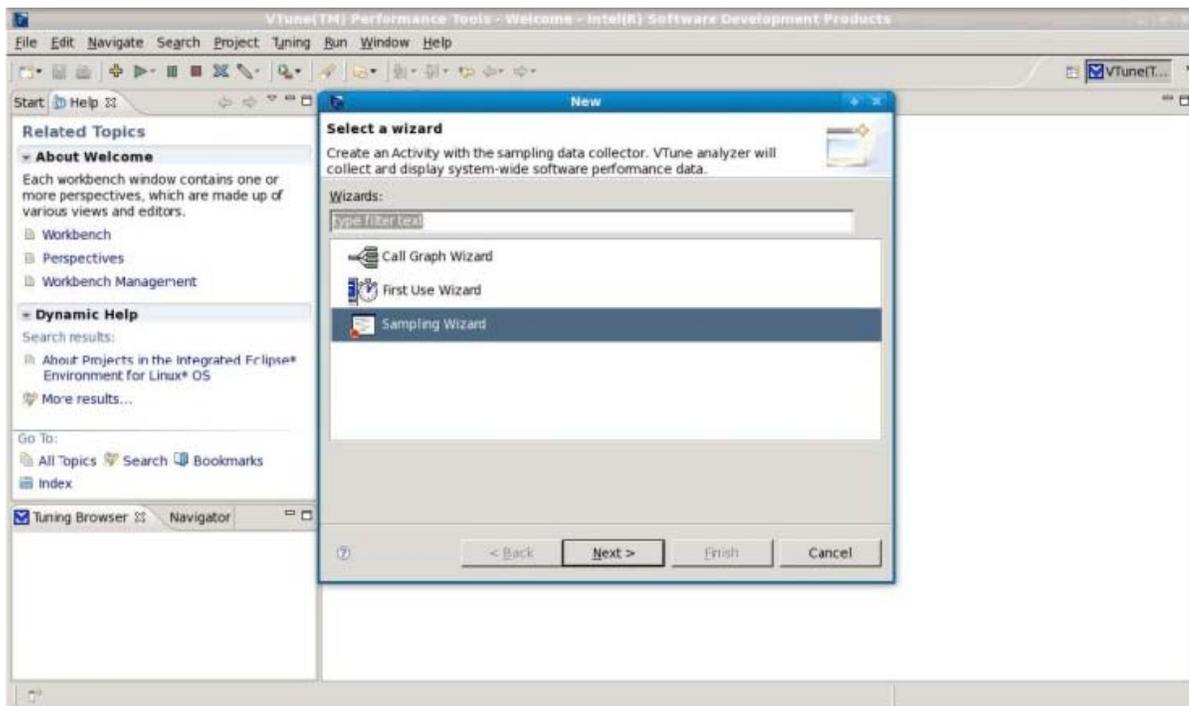
View [All Processes](#) and their functions

Command executed: /users/mtech2008/mdeepak/Documents/workshop/a.out

Learn more:

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

Sampling Wizard View



Process	CPU sam	INST_sampl	Clock per	CPU_CL %	INST_RF %	CPU_CLK_UNHA events	INST_RETIRED A events	
a.out	5,011	7,033	0.798	93.55%	90.42%	11,968,263.000	15,001,389.000	/usr/intel/...
java	180	147	1.268	3.00%	1.95%	383,940.000	307,886.000	/opt/intel/erlip
Xorg	60	36	1.667	1.00%	0.49%	127,980.000	76,788.000	/usr/bin/
artsd	40	29	1.655	0.80%	0.40%	102,304.000	61,057.000	/usr/bin/
bash	21	30	0.700	0.35%	0.41%	44,793.000	63,990.000	/bin/
nto	20	7	2.857	0.33%	0.10%	42,660.000	14,951.000	/opt/sag/exxa
pid_0x0	10	3	3.333	0.17%	0.04%	21,330.000	6,399.000	
kicker	5	0	0.000	0.08%	0.00%	10,665.000	0	/usr/bin/
kwin	4	1	4.000	0.07%	0.03%	8,512.000	2,133.000	/usr/bin/
pid_0x16b413	2	0	0.000	0.03%	0.00%	4,266.000	0	
pid_0x0	2	0	0.000	0.03%	0.00%	4,266.000	0	
libstrptime	1	0	0.000	0.02%	0.00%	2,133.000	0	/bin/
cut	1	0	0.000	0.02%	0.00%	2,133.000	0	/bin/
bash	1	0	0.000	0.02%	0.00%	2,133.000	0	/bin/

Events	Total
Clocks per Instruc	0.80
CPU_CLK_UNHALT...	93.55
CPU_CLK_UNHALT...	11,960,263,...
CPU_CLK_UNHALT...	5,611.00
INST_RETIRED ANY %	96.42
INST_RETIRED ANY...	15,001,389,...
INST_RETIRED ANY...	7,033.00

Activity ID	Activity Result	Total Samples	Duration	Machine Name	CPU ID	Idle time
8	Mon Jun 22 15:43:07 2009 - Sampling Results [csews24]	13292	6.336	csews24.Cse@R172.2	0	0.06%
					1	0.56%

Module 12: View

Lecture 23: Intel Compilers and Threading Tool

Sampling Wizard View

The screenshot shows the VTune Performance Tools interface. The main window displays the Sampling Wizard View for Thread=thread3. The top table lists performance metrics for the thread:

Thread	Process	CPU sam	INST sampl	Clock per...	CPU %	INST %	CPU event	INST event	ThreadID	Process
thread3	a.out	5.611	7.033	0.798	100	100	11	15	18163	Intel(R) Intel(R)2008/Intel(R) Deep

The right-hand pane shows a list of events and their total counts:

Events	Total
Clocks per Instruc...	0.80
CPU_CLK_UNHALT...	100.00
CPU_CLK_UNHALT...	11,968,263...
CPU_CLK_UNHALT...	5.611.00
INS_RETIRED ANY %	100.00
INST_RETIRED ANY...	15,001,389...
INST_RETIRED ANY...	7,033.00

The bottom pane shows the Activity Results table:

Activity ID	Activity Result	Total Samples	Duration	Machine Name	CPU ID	Idle time
8	Mon Jun 22 15:43:07 2009 - Sampling Results [csews24]	13292	0.330	csews24.cse.iitk.172...	0	0.06%
					1	0.56%

The bottom navigation bar includes: Processes, Threads, Modules, Hotspots.

The screenshot shows the VTune Performance Tools interface. The main window displays the Sampling Wizard View for Module=a.out. The top table lists performance metrics for the module:

Module	Process	CPU sam	INST sampl	Clock per...	CPU CL %	INST RE %	CPU CLK UNHALT events	INST RETIR events
a.out	a.out	5.600	7.027	0.797	99.90%	99.91%	11,944,800,000	14,988,591
vmlinux.fc6	a.out	9	6	1.500	0.16%	0.09%	19,197,000	12,794
snd_hdac_hmi	a.out	1	0	0.000	0.02%	0.00%	2,133,000	
lu-2.5.su	a.out	1	0	0.000	0.02%	0.00%	2,133,000	

The right-hand pane shows a list of events and their total counts:

Events	Total
Clocks per Instruc...	0.00
CPU_CLK_UNHALT...	99.80
CPU_CLK_UNHALT...	11,944,800...
CPU_CLK_UNHALT...	5,600.00
INST_RETIRED ANY %	99.91
INST_RETIRED ANY...	14,988,591...
INST_RETIRED ANY...	7,027.00

The bottom pane shows the Activity Results table:

Activity ID	Activity Result	Total Samples	Duration	Machine Name	CPU ID	Idle time
8	Mon Jun 22 15:43:07 2009 - Sampling Results [csews24]	13292	0.336	csews24.cse.iitk.172...	0	0.06%
					1	0.56%

The bottom navigation bar includes: Processes, Threads, Modules, Hotspots.

◀ Previous Next ▶

Sampling Wizard View

Name	CPU_CLK_U samples	INST_RETI samples	Clocks per Instruct...	CPU_CLK_UNH %	INST_RETIRED A %	CPU_CLK_UNHA events	INST_RETIRED A events
func1	5,097	6,323	0.806	01.02%	89.98%	10,871,001,000	13,486,050,000
func2	503	704	0.714	8.98%	10.02%	10,871,001,000	13,486,050,000

Events	Total
Clocks per Instruc...	0.81
CPU_CLK_UNHAIT	41.07
CPU_CLK_UNHALT...	10,871...
CPU_CLK_UNHALT...	5,097.00
INST_RETIRED_ANY %	89.98
INST_RETIRED_ANY...	13,486...
INST_RETIRED_ANY...	6,323.00

Activity ID	Activity Result	Total Samples	Duration	Machine Name	CPU ID	Idle time
8	Mon Jun 22 15:43:07 2009 - Sampling Results [csews24]	1,529,2	0.33s	csews24.cse.rkl.fz	0	0.06%
					1	0.56%

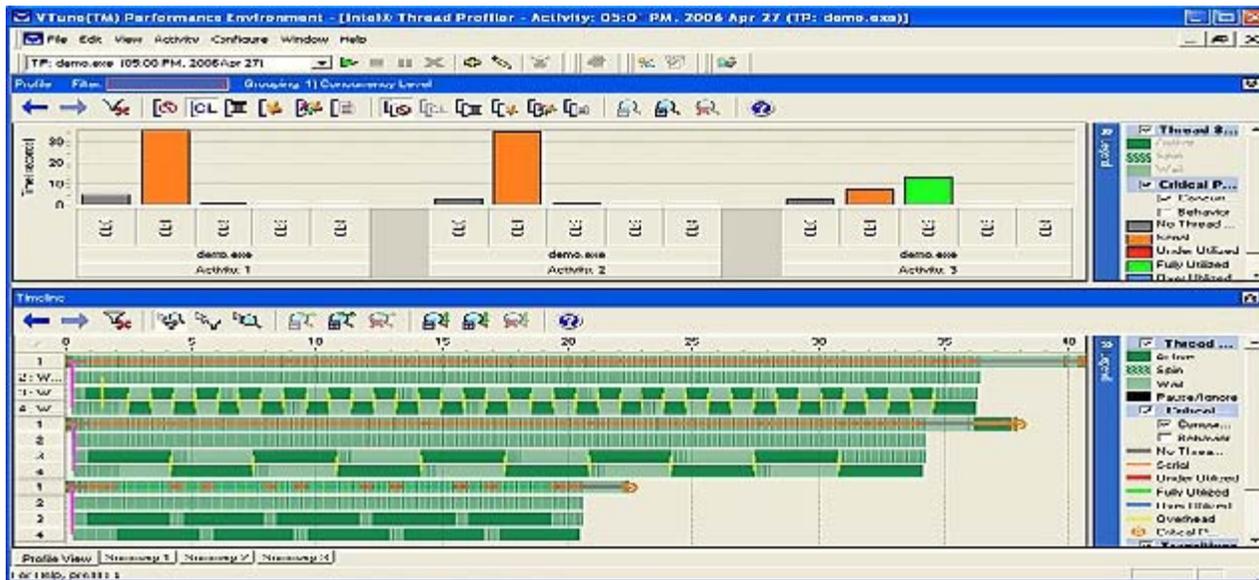
Intel Vtune Tips

- 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

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];
            }
        }
    }
}
```

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/sem 2/parallel_summer course/inteltools_ppt/programs/section
```

Application finished

```
Intel(R) Thread Profiler 3.1 Summary Report
application:      /media/STUDIES/sem-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** 