Profiling OpenMP

Guy Tel-Zur
December 2014

- Test program: Fibonacci.c
- → Dynamic threads creation
- → Recursion
- Tools
- → Intel's VTune
- → Solaris(Sun(Oracle))Sutdio

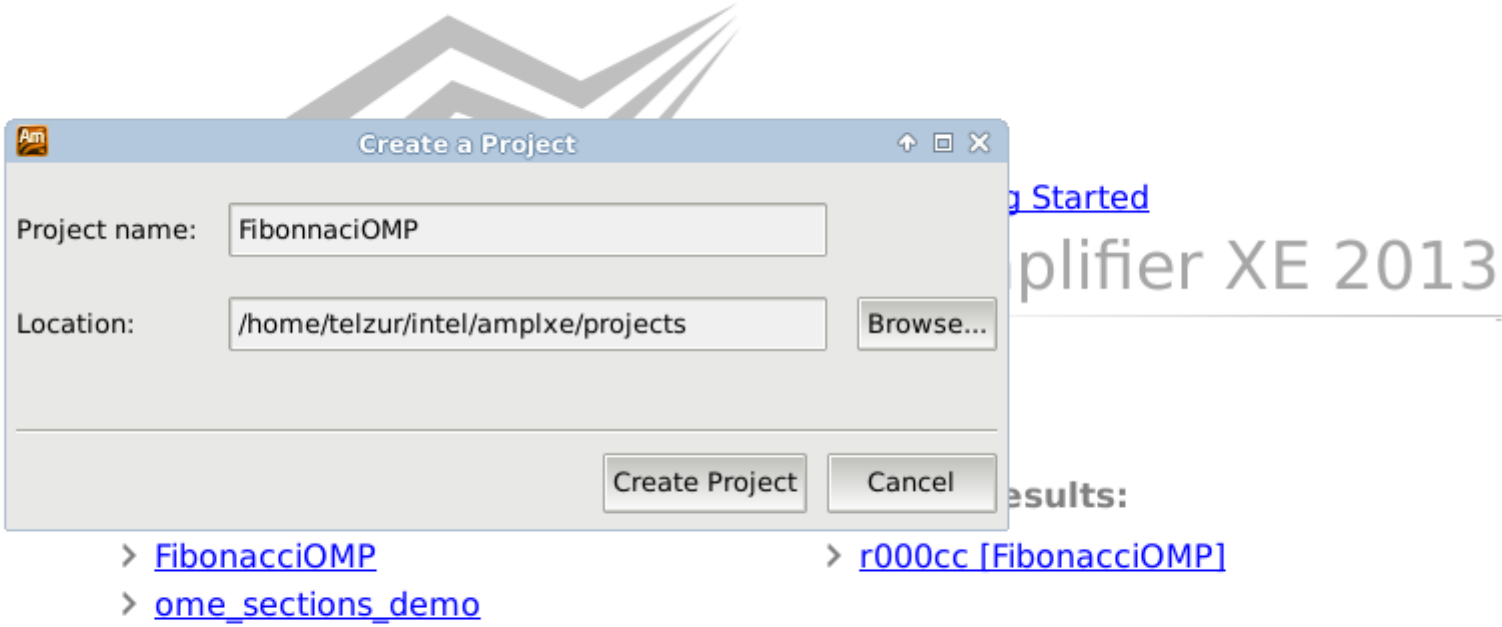
```
#include <stdio.h>
#include <omp.h>
int fib(int n)
{
  int i, j;
  if (n<2)
    return n;
  else
    {
       #pragma omp task shared(i) firstprivate(n)
       i=fib(n-1);
       #pragma omp task shared(j) firstprivate(n)
       j=fib(n-2);
       #pragma omp taskwait
       return i+j;
    }
}
int main()
{
  int n = 10;
  omp_set_dynamic(0);
  omp_set_num_threads(4);
  #pragma omp parallel shared(n)
  {
    #pragma omp single
    printf ("fib(%d) = %d\n", n, fib(n));
  }
}
```

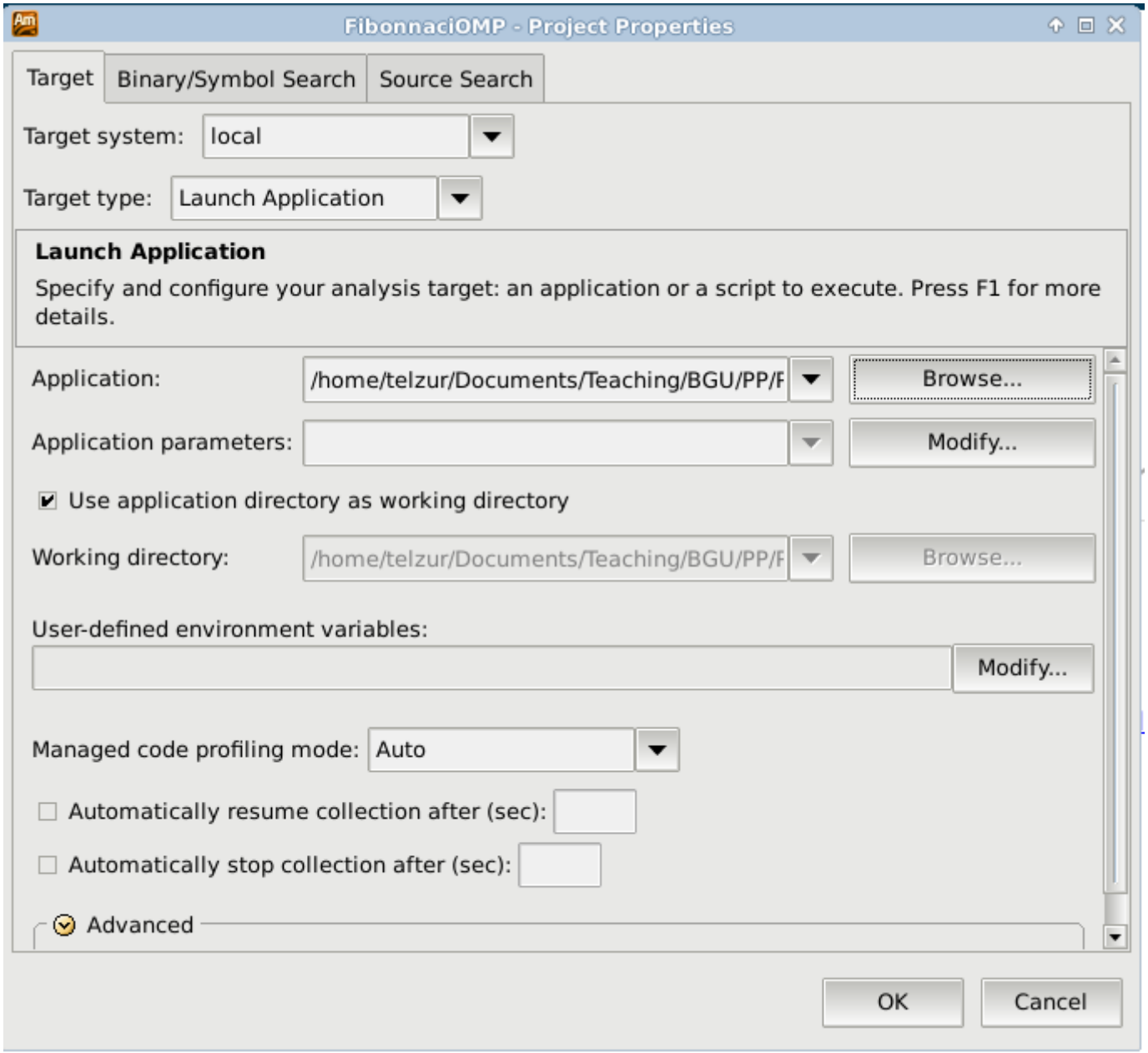
The task construct defines an explicit task. The encountering thread may immediately execute the task, or defer its execution. In the latter case, any thread in the team may be as signed the task

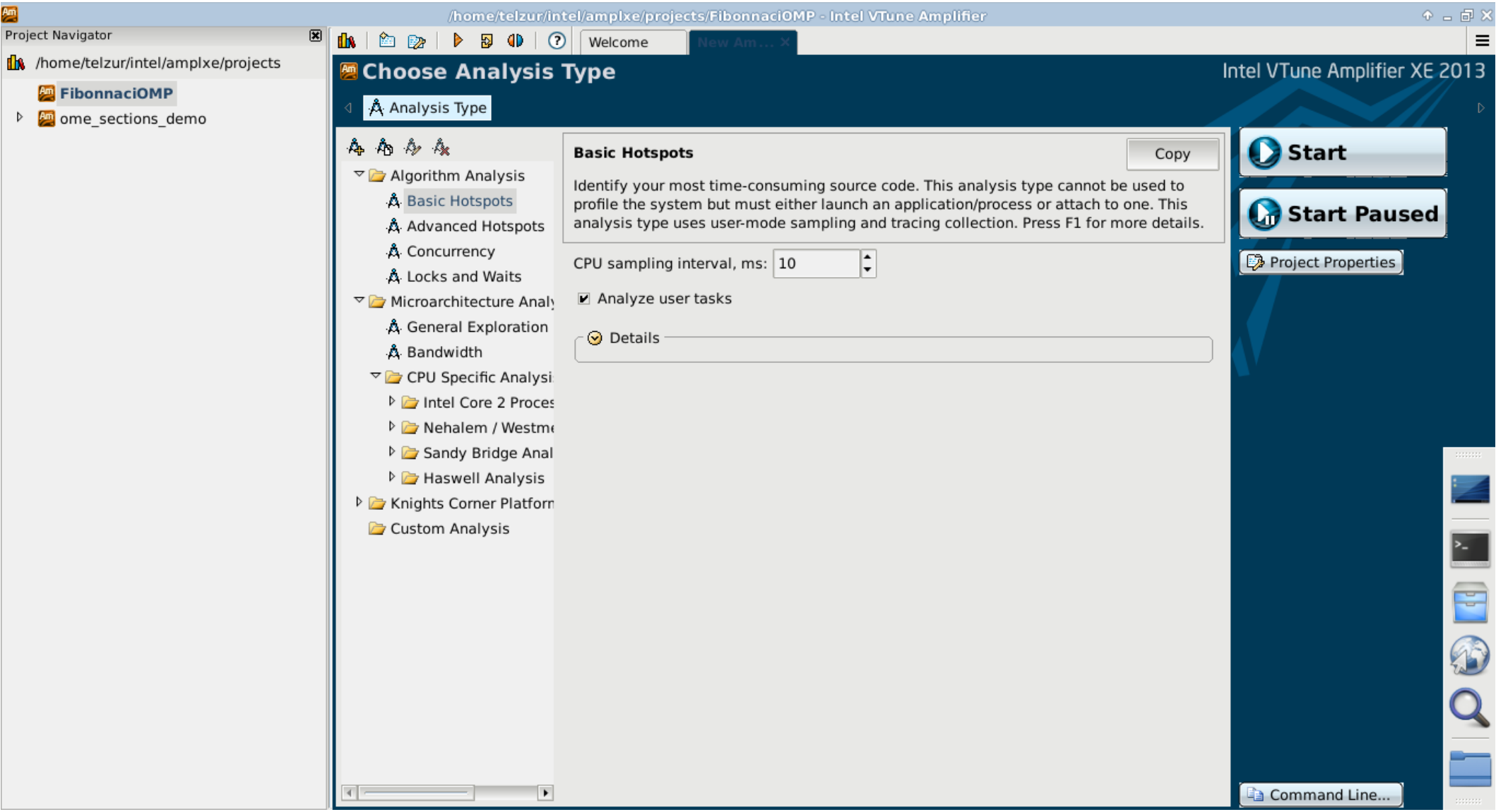
The taskwait construct specifies a wait on the completion of child tasks of the current task.

VTune

- **Prepare:**
  - `echo 0|sudo tee /proc/sys/kernel/yama/ptrace_scope`
  - `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/intel/composer_xe_2013_sp1.3.174/compiler/lib/intel64`
  - `export KMP_FORKJOIN_FRAMES=1`
- **Compile:** `/opt/intel/bin/icc -fopenmp -g -parallel-source-info=2 -o ./fibonacci_intel ./fibonacci_intel.c`
- Execute: `./fibonacci_intel`
- **Start VTune:** `/opt/intel/vtune_amplifier_xe_2013/bin64/amplxe-gui`


Create a Project
Project name:
FibonnaciOMP
Location:
/home/telzur/intel/amplxe/projects
Browse...
Create Project
Cancel
g Started
plifier XE 2013
esults:
FibonacciOMP
ome_sections_demo
r000cc [FibonacciOMP]


FibonnaciOMP - Project Properties
Target
Binary/Symbol Search
Source Search
Target system:
local
Target type:
Launch Application
Launch Application
Specify and configure your analysis target: an application or a script to execute. Press F1 for more details.
Application:
/home/telzur/Documents/Teaching/BGU/PP/F
Browse...
Application parameters:
Modify...
Use application directory as working directory
Working directory:
/home/telzur/Documents/Teaching/BGU/PP/F
Browse...
User-defined environment variables:
Modify...
Managed code profiling mode:
Auto
Automatically resume collection after (sec):
Automatically stop collection after (sec):
Advanced
OK
Cancel


/home/telzur/intel/amplxe/projects/FibonnaciOMP - Intel VTune Amplifier
Project Navigator
/home/telzur/intel/amplxe/projects
FibonnaciOMP
ome_sections_demo
Welcome
New Am...
Choose Analysis Type
Intel VTune Amplifier XE 2013
Analysis Type
Algorithm Analysis
Basic Hotspots
Advanced Hotspots
Concurrency
Locks and Waits
Microarchitecture Analy
General Exploration
Bandwidth
CPU Specific Analysi
Intel Core 2 Proces
Nehalem / Westme
Sandy Bridge Anal
Haswell Analysis
Knights Corner Platforn
Custom Analysis
Basic Hotspots
Copy
Identify your most time-consuming source code. This analysis type cannot be used to profile the system but must either launch an application/process or attach to one. This analysis type uses user-mode sampling and tracing collection. Press F1 for more details.
CPU sampling interval, ms: 10
Analyze user tasks
Details
Start
Start Paused
Project Properties
Command Line...

/home/telzur/intel/amplxe/projects/FibonnaciOMP - Intel VTune Amplifier

Project Navigator

/home/telzur/intel/amplxe/projects

- FibonnaciOMP
  - r000hs
- ome_sections_demo

Welcome | r000hs

Basic Hotspots Hotspots viewpoint (change) Intel VTune Amplifier XE 2013

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Tasks and Frames

Elapsed Time: 0.023s

Total Thread Count:
CPU Time:
Paused Time:
Frame Count:

Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

Application Command Line:	/home/telzur/Documents/Teaching/BGU/PP/PP2015A/lectures/08/code/fibonacci_intel
Operating System:	3.13.0-43-generic DISTRIB_ID=LinuxMint DISTRIB_RELEASE=17 DISTRIB_CODENAME=qiana DISTRIB_DESCRIPTION="Linux Mint 17 Qiana"
Computer Name:	LIFEBOOK
Result Size:	1 MB
Collection start time:	05:48:31 24/12/2014 UTC
Collection stop time:	05:48:32 24/12/2014 UTC

CPU

Name:	3rd generation Intel(R) Core(TM) Processor family
Frequency:	2.2 GHz
Logical CPU Count:	8

/home/telzur/intel/amplxe/projects/FibonnaciOMP - Intel VTune Amplifier

Project Navigator

/home/telzur/intel/amplxe/projects

- FibonnaciOMP
  - r000hs
- ome_sections_demo

Welcome | r000hs

Basic Hotspots Hotspots viewpoint (change) Intel VTune Amplifier XE 2013

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Tasks and Frames

Project Properties

Re-resolve

Import from CSV

Data collection is completed successfully

The result file '/home/telzur/intel/amplxe/projects/FibonnaciOMP/r000hs/r000hs.amplxe' is successfully created and added to the project FibonnaciOMP.

Finalization completed successfully

Result is ready for analysis.

Application Output

Command Line...

/home/telzur/intel/amplxe/projects/FibonnaciOMP - Intel VTune Amplifier

Welcome | r000hs | r001cc

Concurrency Hotspots by CPU Usage viewpoint (change)

Intel VTune Amplifier XE 2013

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Tasks and Frames

Grouping: Function / Call Stack

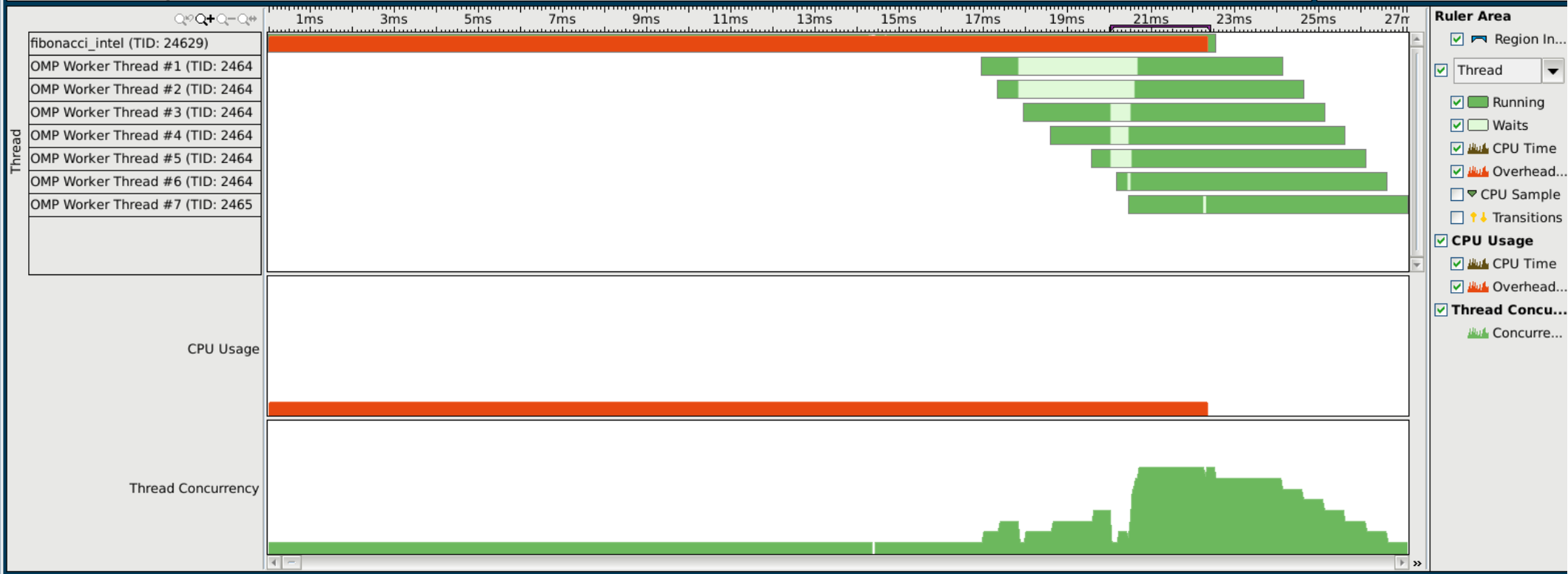
Function / Call Stack	CPU Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Wait Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Over... and S...	Thr.. Ov..	Module	Function (Full)	Source File	Start Address
__kmpc_barrier	20.000ms		20.000ms	0ms	libiomp5.so	__kmpc_barrier	kmp_csupport.c	0x474e0
__kmp_api_omp_set_c		0.092ms			libiomp5.so	__kmp_api_omp_set_dynamic	kmp_ftn_entry.h	0x552f0
__kmp_launch_thread		7.043ms			libiomp5.so	__kmp_launch_thread	kmp_runtime.c	0x65870
__printf		0.030ms			libc-2.19.so	__printf	printf.c	0x542f0
Selected 1 row(s):	20.000ms		20.000ms	0ms				

Data Of Interest (CPU Metrics)

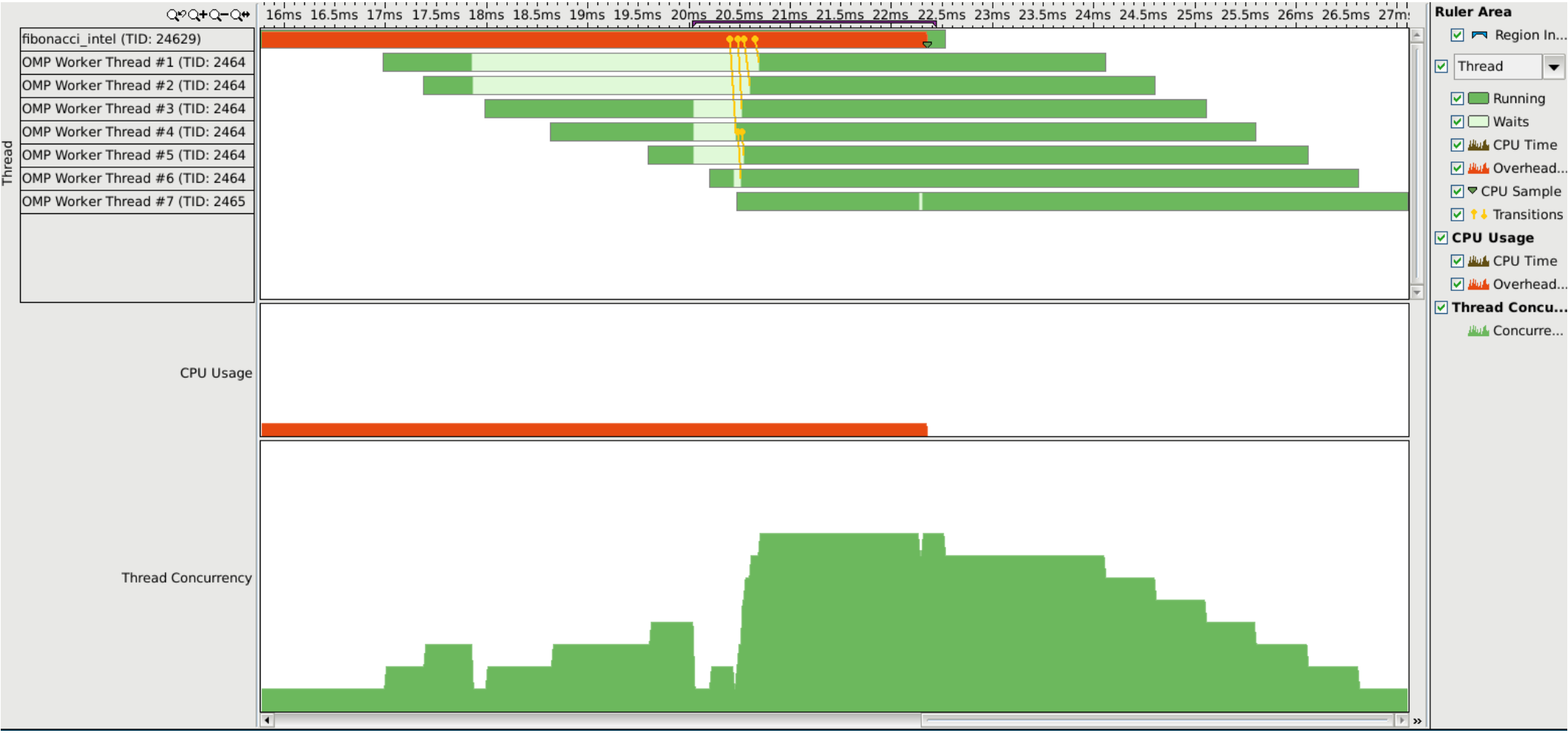
Viewing 1 of 1 selected stack(

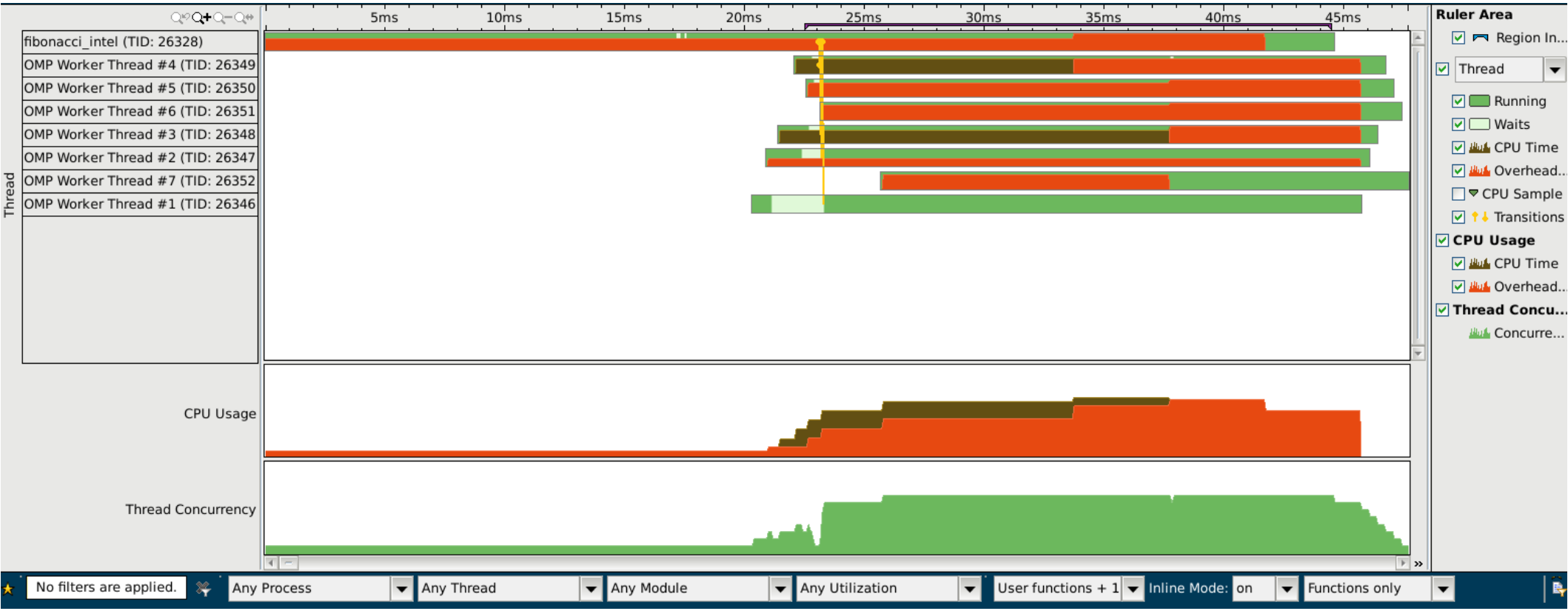
100.0% (0.020s of 0.020s)

libiomp5.so!__kmpc_barrier - kmp...
fibonacci_intel!mainompparall...
libiomp5.so!__kmp_invoke_micro...
libiomp5.so![OpenMP dispatcher]...
libiomp5.so!__kmp_fork_call+0x1...
libiomp5.so![OpenMP fork]+0xd4 ...
fibonacci_intel!main+0xfe - fibona...
libc-2.19.so!__libc_start_main+0xf4...



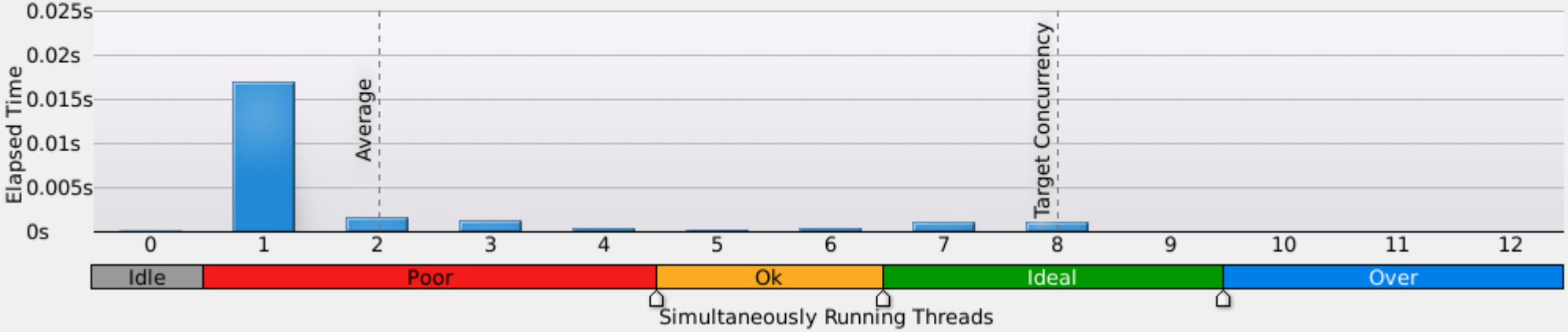
No filters are applied. | Any Process | Any Thread | Any Module | Any Utilization | User functions + 1 | Inline Mode: on | Functions only


16ms 16.5ms 17ms 17.5ms 18ms 18.5ms 19ms 19.5ms 20ms 20.5ms 21ms 21.5ms 22ms 22.5ms 23ms 23.5ms 24ms 24.5ms 25ms 25.5ms 26ms 26.5ms 27ms
Thread
fibonacci_intel (TID: 24629)
OMP Worker Thread #1 (TID: 2464
OMP Worker Thread #2 (TID: 2464
OMP Worker Thread #3 (TID: 2464
OMP Worker Thread #4 (TID: 2464
OMP Worker Thread #5 (TID: 2464
OMP Worker Thread #6 (TID: 2464
OMP Worker Thread #7 (TID: 2465
CPU Usage
Thread Concurrency
Ruler Area
Region In...
Thread
Running
Waits
CPU Time
Overhead...
CPU Sample
Transitions
CPU Usage
CPU Time
Overhead...
Thread Concu...
Concurre...


5ms
10ms
15ms
20ms
25ms
30ms
35ms
40ms
45ms
Thread
fibonacci_intel (TID: 26328)
OMP Worker Thread #4 (TID: 26349
OMP Worker Thread #5 (TID: 26350
OMP Worker Thread #6 (TID: 26351
OMP Worker Thread #3 (TID: 26348
OMP Worker Thread #2 (TID: 26347
OMP Worker Thread #7 (TID: 26352
OMP Worker Thread #1 (TID: 26346
CPU Usage
Thread Concurrency
Ruler Area
Region In...
Thread
Running
Waits
CPU Time
Overhead...
CPU Sample
Transitions
CPU Usage
CPU Time
Overhead...
Thread Concu...
Concurre...
No filters are applied.
Any Process
Any Thread
Any Module
Any Utilization
User functions + 1
Inline Mode:
on
Functions only

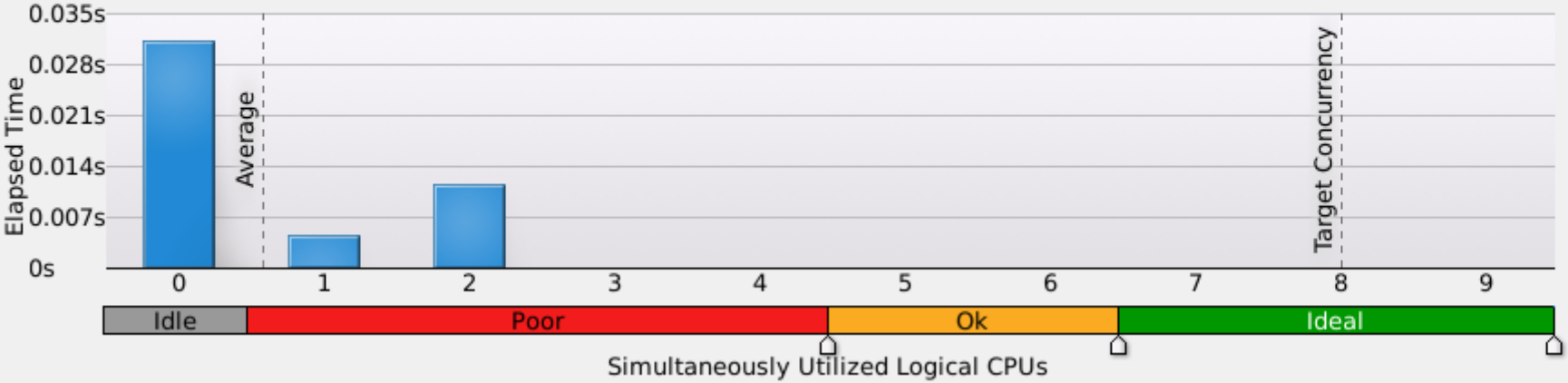
Thread Concurrency Histogram

This histogram represents a breakdown of the Elapsed Time. It visualizes the percentage of the wall time the specific number of threads were running simultaneously. Threads are considered running if they are either actually running on a CPU or are in the runnable state in the OS scheduler. Essentially, Thread Concurrency is a measurement of the number of threads that were not waiting. Thread Concurrency may be higher than CPU usage if threads are in the runnable state and not consuming CPU time.



CPU Usage Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU usage value. CPU usage may be higher than the Thread Concurrency level if a thread is executing code on a CPU while it is logically waiting. Try to keep your Target Concurrency value as close to the CPU number as possible.



Concurrency Hotspots by CPU Usage viewpoint (change) — Intel VTune Amplifier XE 2013

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Tasks and Frames | fibonacci_... | fibonacci_in...

Source | Assembly | Assembly grouping: Address

S. Li.	Source	CPU Time: Total by Utilization (Idle, Poor, Ok, Ideal, Over)	CPU Time: Self by Utilization (Idle, Poor, Ok, Ideal, Over)	Wait Time: Total by (Idle, Poor, Ok, Id)
1				
2	#include <stdio.h>			
3	#include <omp.h>			
4	int fib(int n)			
5	{			
6	int i, j;			
7	if (n<2)			
8	return n;			
9	else			
10	{			
11	#pragma omp task			
12	i=fib(n-1);	12.011ms	0ms	
13				
14	#pragma omp task			
15	j=fib(n-2);	12.011ms	0ms	
16				
17	#pragma omp taskw	12.011ms	0ms	
18	return i+j;			
19	}			
20	}			
21				
22	int main()			
23	{			
24	int n = 20;			
25				
26	omp_set_dynamic(0);			
27	omp_set_num_threads(8)			
28				
29	#pragma omp parallel s			
30	{			
31	#pragma omp single			
32	printf ("fib(%d) = %			
33	}			
34	}			
35				

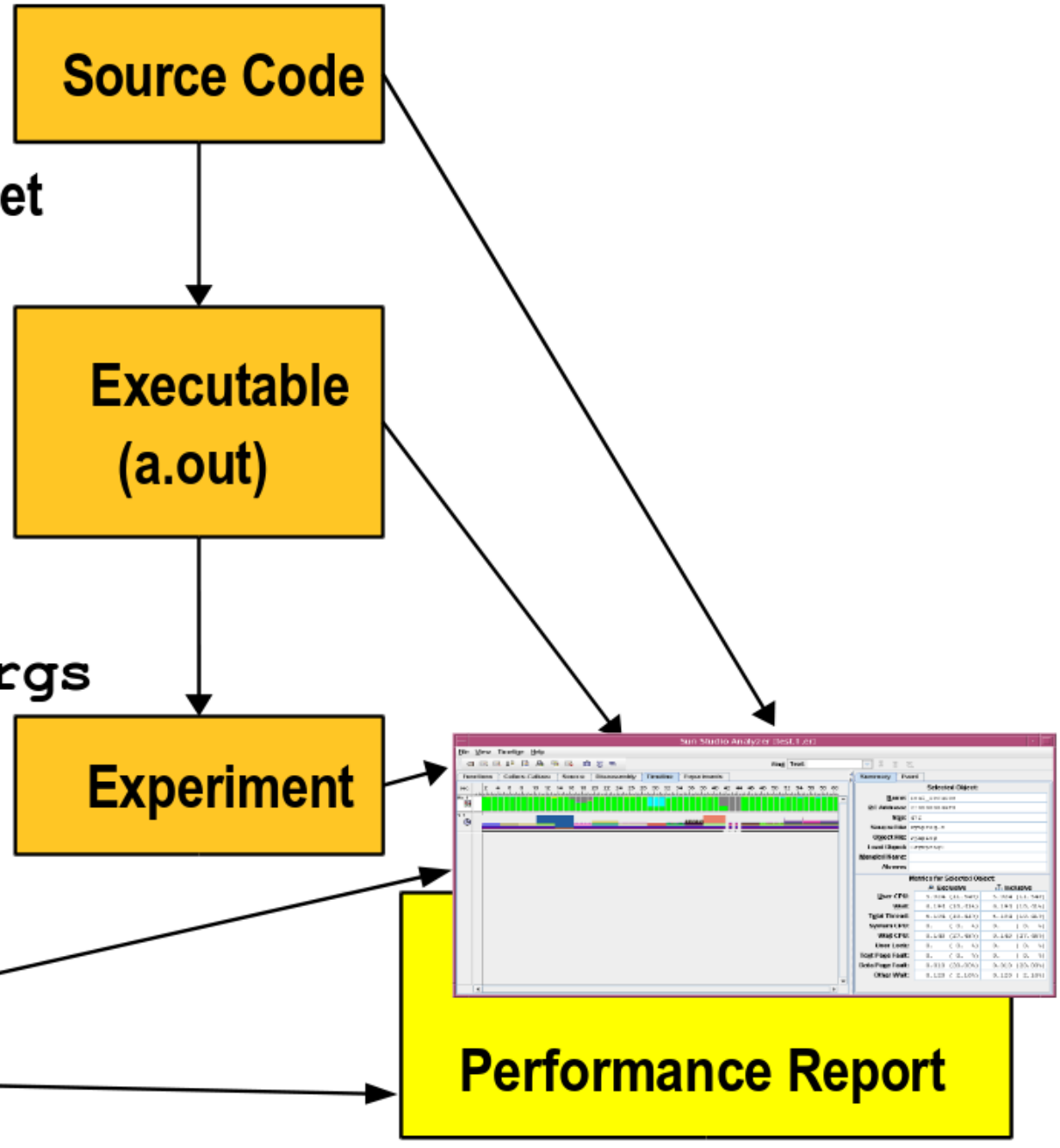
Data Of Interest (CPU Metrics)

Viewing 1 of 1 selected stack(s)

100.0% (0.012s of 0.012s)

- libiomp5.so!__kmpc_omp_taskwait+0x12b - kmp_tasking.c:1173
- fibonacci_intel!fib+0x1ac - fibonacci_intel.c:17
- fibonacci_intel!fib+0x2c0 - fibonacci_intel.c:15
- libiomp5.so!__kmpc_barrier+0x6a - kmp_csupport.c:647
- fibonacci_intel!mainompparallel@29+0xa6 - fibonacci_intel.c:31
- libiomp5.so!__kmp_invoke_microtask+0x92 - [Unknown]:[Unknown]
- libiomp5.so![OpenMP dispatcher]+0x86 - kmp_runtime.c:6917
- libiomp5.so!__kmp_fork_call+0x12ec - kmp_runtime.c:2205
- libiomp5.so![OpenMP fork]+0xd4 - kmp_csupport.c:312
- fibonacci_intel!main+0xfe - fibonacci_intel.c:29
- libc-2.19.so!__libc_start_main+0xf4 - libc-start.c:287

User Model



Step 1. Compile the target

```
make a.out
```

Step 2. Collect data

```
collect a.out args
```

Step 3. Examine the data

`analyzer` (GUI)

`er_print` (text)

SolarisStudio

```
export PATH=~/Downloads/SolarisStudio12.3-linux-x86-bin/solarisstudio12.3/bin:$PATH

suncc -xopenmp=parallel -g -o ./fibonacci_sun ./fibonacci_sun.c

collect ./fibonacci_sun

analyzer ./test.2.er
```

SolarisStudio supports also GNU compilers!

Oracle Solaris Studio Performance Analyzer [test.3.er [fibonacci_sun]]

File View Help

View Mode User Find Text:

Functions | Callers-Callees | Call Tree | Source | Disassembly | OpenMP Parallel Region | Summary | Timeline Details

Source File: fibonacci_sun.c
Object File: fibonacci_sun
Load Object: <fibonacci_sun>

User CPU (sec.)	User CPU (sec.)	OMP Work (sec.)	OMP Wait (sec.)	Source
				1.
				2. #include <stdio.h>
				3. #include <omp.h>
				4. int fib(int n)
0.	0.	0.	0.	5. {
				<Function: fib>
				6. int i, j;
0.	0.	0.	0.	7. if (n<2)
				8. return n;
				9. else
				10. {
				Source OpenMP region below has tag R1
				Shared variables in R1: i
				Firstprivate variables in R1: n
0.020	0.020	0.	0.	11. #pragma omp task shared(i) firstprivate(n)
0.	0.020	0.	0.	12. i=fib(n-1);
				13.
				Source OpenMP region below has tag R2
				Shared variables in R2: j
				Firstprivate variables in R2: n
0.	0.	0.	0.	14. #pragma omp task shared(j) firstprivate(n)
0.	0.020	0.	0.	15. j=fib(n-2);
				16.
0.	0.	0.	0.	17. #pragma omp taskwait
0.	0.	0.	0.	18. return i+j;
				19. }
0.	0.	0.	0.	20. }
				21.
				22. int main()
0.	0.	0.	0.	23. {
				<Function: main>
0.	0.	0.	0.	24. int n = 20;
				25.
0.	0.	0.	0.	26. omp_set_dynamic(0);
0.	0.	0.	0.	27. omp_set_num_threads(8);
				28.
				Source OpenMP region below has tag R3
				Shared variables in R3: n
0.	0.	0.	0.	29. #pragma omp parallel shared(n)

Selected Object:

Name:	line 5 in "fibonacci_sun.c"
PC Address:	2:0x000009D0
Size:	0
Source File:	P2015A/lectures/08/code/fibonacci
Object File:	fibonacci_sun
Load Object:	<fibonacci_sun>
Mangled Name:	
Aliases:	

Metrics for Selected Object:

	Exclusive	Inclus
User CPU:	0. (0. %)	0. (
OpenMP Work:	0. (0. %)	0. (
OpenMP Wait:	0. (0. %)	0. (
OpenMP Overhead:	0. (0. %)	0. (

Oracle Solaris Studio Performance Analyzer [test.5.er [fibonacci_sun]]

File View Help

View Mode Expert Find Text:

Functions | Callers-Callees | Call Tree | Source | Disassembly | Lines | Timeline | Open...

Summary | Timeline Details

Time(msec) 135 140 145

1.3

1.4

1.2

1.1

Scale(msec) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Details for Selected Event

Experiment Name:

Event Type:

Leaf Function:

Timestamp (sec.):

LWP:

Thread:

CPU:

References

- Profiling OpenMP applications with Intel Vtune Amplifier XE. https://software.intel.com/sites/default/files/managed/d8/e0/profiling-openmp-applications-with-intel-vtune-amplifier-xe.pdf

- HPC Profiling with the Sun Studio Performance Tools by Itzkowitz et al, http://tools.zih.tu-dresden.de/2009/downloads/itzkowitz-SunStudio-toolsws2009.pdf

- Test program: http://docs.oracle.com/cd/E19205-01/820-7883/girtd/index.html