Assignment No. B3

Title: Implement a stack sampling using threads using VTune Amplifier.

Aim: Write a program to develop a stack sampling using threads using VTune Amplifier

Objectives:

- To understand sampling and VTune Amplifier.

Theory:

Intel VTune Amplifier is a commercial application for software performance analysis. Although basic features work on both Intel and AMD hardware, advanced hardware-based sampling requires an Intel-manufactured CPU. It is available as part of Intel Parallel Studio or as a stand-alone product.

Code optimization

VTune Amplifier assists in various kinds of code profiling including stack sampling, thread profiling and hardware event sampling. The profiler result consists of details such as time spent in each sub routine which can be drilled down to the instruction level. The times taken by the instructions are indicative of any stalls in the pipeline during instruction execution. The tool can be also used to analyze thread performance.

Features

1. **Software sampling**
   Works on x86 compatible processors and gives both the locations where time is spent and the call stack used.

2. **JIT profiling support**
   Profiles dynamically generated code.

3. **Locks and waits analysis**
   Finds long synchronization waits that occur when cores are underutilized.

4. **Threading timeline**
   Shows thread relationships to identify load balancing and synchronization issues. It can also be used to select a region of time and filter the results. This can remove the clutter of data gathered during uninteresting times like application start-up.

5. **Source view**
   Sampling results are displayed line by line on the source / assembly code.

6. **Hardware event sampling**
   This uses the on chip performance monitoring unit and requires an Intel processor. It can find specific tuning opportunities like cache misses and branch mispredictions.

7. **Performance tuning utility (PTU)**

PTU Was a separate download that gave VTuneAmplifer XE users access to experimental tuning technology. This includes things like Data Access Analysis that identifies memory hotspots and relates them to code hotspots. PTU now is fully integrated into VTuneAmplifer XE.

Performance Profiling with Intel VTune Amplifier XE

System Requirements

Processor requirements

The list of supported processors is constantly being extended. Here is a partial list of processors where the EBS analysis is enabled:

Mobile Processors

- Intel® Atom™ Processor
- Intel® Core™ i7 Mobile Processor Extreme Edition (including 2nd, 3rd, 4th and 5th Generation Intel® Core™ processors)
- Intel® Core™ i7, i5, i3 Mobile Processors (including 2nd, 3rd, 4th, 5th and 6th Generation Intel® Core™ processors)
- Intel® Core™2 Extreme Mobile Processor
- Intel® Core™2 Quad Mobile Processor
- Intel® Core™2 Duo Mobile Processor
- Intel® Pentium® Mobile Processor

Desktop Processors

- Intel® Atom™ Processor
- Intel® Core™ i7 Desktop Processor Extreme Edition (including 2nd, 3rd, 4th and 5th Generation Intel® Core™ processors)
- Intel® Core™ i7, i5, i3 Desktop Processors (including 2nd, 3rd, 4th, 5th and 6th Generation Intel® Core™ processors)
- Intel® Core™2 Quad Desktop Processor
- Intel® Core™2 Extreme Desktop Processor
- Intel® Core™2 Duo Desktop Processor

Server and Workstation Processors

- Intel® Xeon® processors E7 family (including v2 and v3)
- Intel® Xeon® processor E5 family (including v2 and v3)
- Intel® Xeon® processors E3 family (including v2, v3 and v4)
- Intel® Xeon® Processor D family
- Intel® Xeon® Processor 7000 Sequence
- Intel® Xeon® Processor 6000 Sequence
- Intel® Xeon® Processor 5000 Sequence

- Intel® Xeon® Processor 3000 Sequence
- Intel® Xeon Phi™ Coprocessors
- Quad-Core Intel® Xeon® processors 7xxx, 5xxx, and 3xxx series
- Dual-Core Intel® Xeon® processors 7xxx, 5xxx, and 3xxx series

System Memory Requirements

- At least 2 GB of RAM

Disk Space Requirements

- 900 MB free disk space required for all product features and all architectures

Software Requirements

1. **Supported Linux distributions:**

- Red Hat* Enterprise Linux 5, 6 and 7 [1]
- CentOS* versions equivalent to Red Hat* Enterprise Linux* versions listed above
- SUSE* Linux* Enterprise Server (SLES) 11 and 12
- Fedora* 221 and 232 □ Ubuntu* 12.04, 14.04 and 15.1004
- Debian* 7.0 and 8.0

2. **Supported compilers:**

- Intel® C/C++ Compiler 11 and higher
- Intel® Fortran Compiler 11 and higher
- GNU C/C++ Compiler 3.4.6 and higher

3. **Supported programming languages:**

- Fortran
- C
- C++
- Java*
- OpenCL*

Installation steps

1. Unzip the tar file
2. After Unzipping, enter the unzipped folder and copy the path of that folder using RIGHT CLICK + PROPERTIES.
3. In the terminal, enter the following command: cd <path> (path: the path you copied in the previous step)
4. Enter the next command : ./install_GUI.sh
5. Installation window will open; continue the installation. (SIMPLE) 6. Installation done!

Steps to use VTune Amplifier.

1. Go to Computer in Files; Enter opt folder, enter vtune_apmlifier<version> folder.
2. Copy the path of that folder using RIGHT CLICK + PROPERTIES.
3. Open a new terminal; Enter the following command: source <path>/amplxe-vars.sh (path: Paste the path copied in the previous step)
4. Enter the next command: source <path>/amplxe-vars.sh&
5. You have now entered into the Vtune Amplifier Interpreter.
6. Enter 'amplxe-gui'
7. Vtune Amplifier Application Opens!

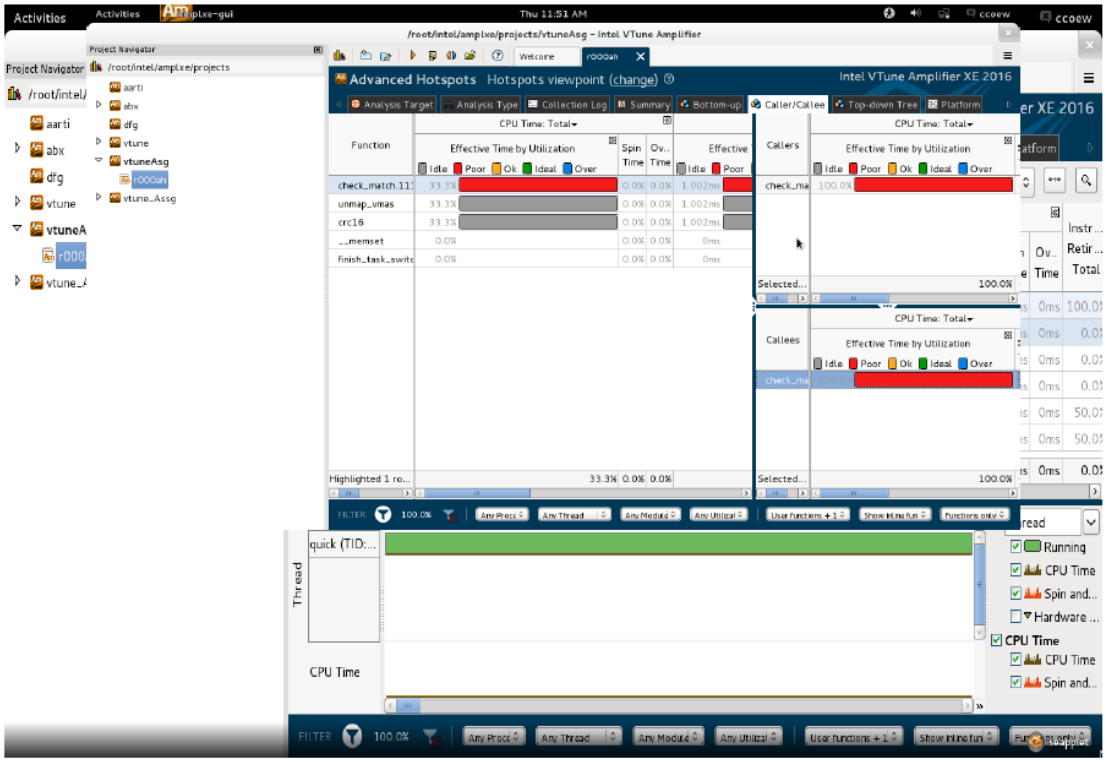
Conclusion: Hence we have studied stack sampling using threads and using VTune.

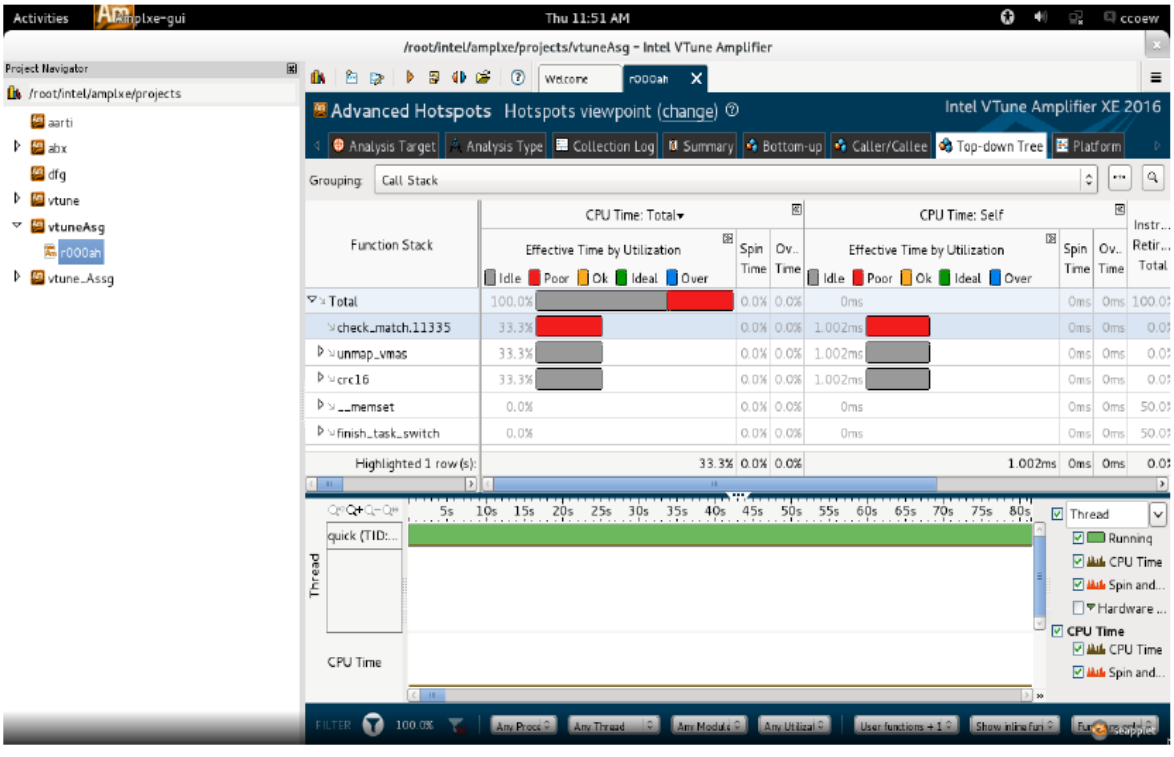
Code:

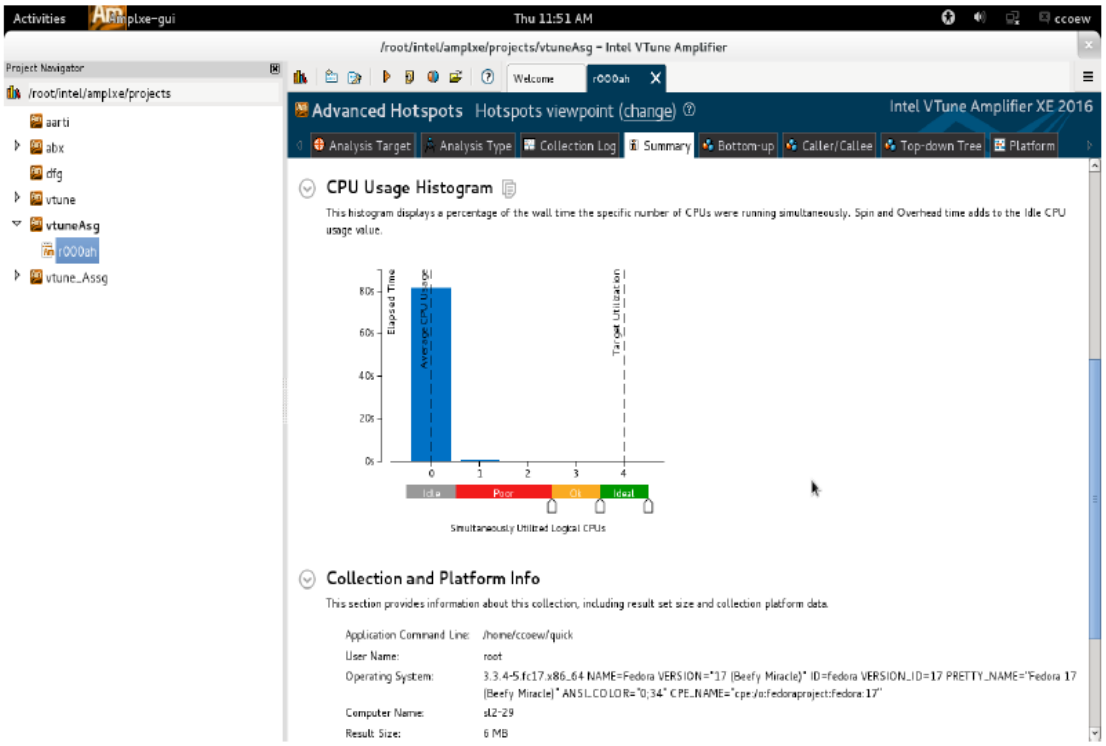
```
#include<iostream>
#include<omp.h>
using namespace std;
int k=0;
class sort
{
        int a[20];
        int n;
public:
        void getdata();
        void Quicksort();
        void Quicksort(int low, int high);
        int partition(int low, int high);
        void putdata();
};
void sort::getdata()
{
        cout<<"Enter the no. of elements in array\t";
        cin>>n;
        cout<<"Enter the elements of array:"<<endl;
        for(int i=0;i<n;i++)
        {
        cin>>a[i];
        }
}
void sort::Quicksort()
{
Quicksort(0,n-1);
}

void sort::Quicksort(int low, int high)
{
if(low<high)
{
        int partn;
        partn=partition(low,high);

cout<<"\n\nThread Number: "<<k<<"  pivot element selected : "<<a[partn];
        #pragma omp parallel sections
        {
        #pragma omp section
                {
                k=k+1;
                Quicksort(low, partn-1);
                }
        #pragma omp section
                {
                k=k+1;
                Quicksort(partn+1, high);
                }
        }//pragma_omp Parallel_end
}

}
int sort::partition(int low ,int high)
{
        int pvt;
        pvt=a[high];
        int i;
        i=low-1;
        int j;
        for(j=low;j<high;j++)
```

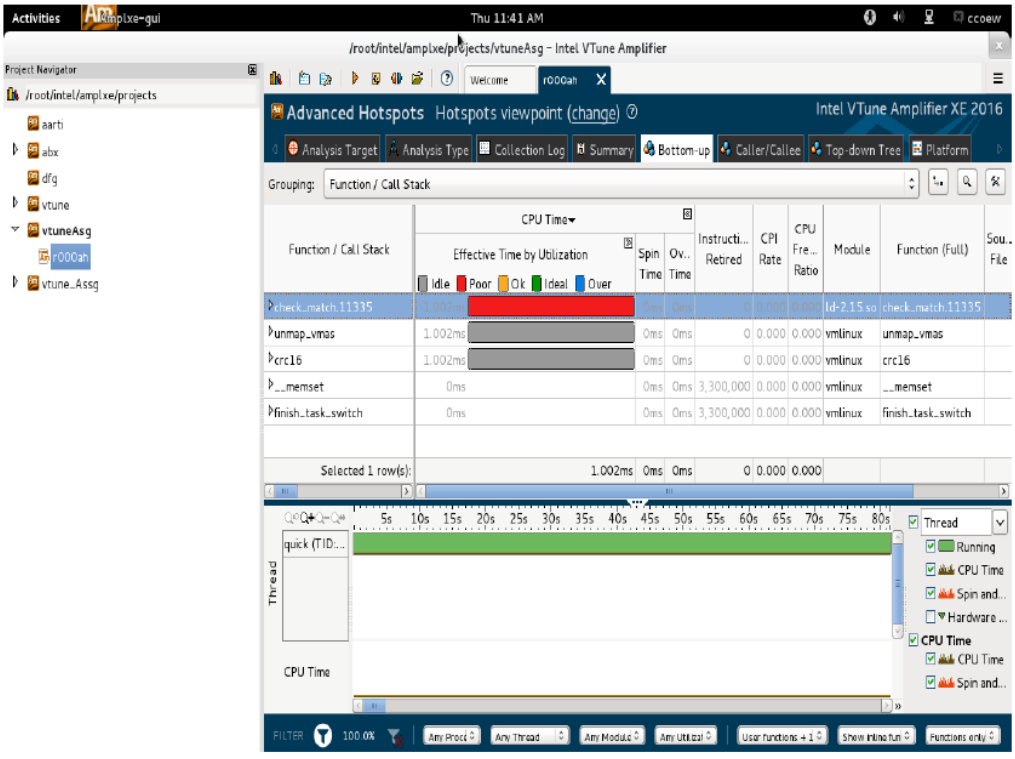
```
        {
                if(a[j]<=pvt)
                {
                        int tem=0;
                        tem=a[j];
                        a[j]=a[i+1];
                        a[i+1]=tem;
                        i=i+1;
                }
        }
        int te;
        te=a[high];
        a[high]=a[i+1];
        a[i+1]=te;
        return i+1;
}
void sort::putdata()
{
        cout<<endl<<"\nThe Array is:"<<endl;
        for(int i=0;i<n;i++)
        cout<<" "<<a[i];
}
int main()
{
        int n;
        sort s1;
        int ch;
do
{
                s1.getdata();
                s1.putdata();

                cout<<"\nUsing Quick Sort";
   double start = omp_get_wtime();
                s1.Quicksort();
   double end = omp_get_wtime();
                cout<<"\nThe Sorted  ";
                s1.putdata();
   cout<<"\nExcecution time : "<<end - start<<" seconds ";

cout<<"Would you like to continue? (1/0 y/n)"<<endl;
cin>>ch;
}while(ch==1);
}
```

```
mpiu@DB21:/mirror/mpiu/CL4 prog/B-3$ g++ quicksort_sample.cpp -fopenmp
mpiu@DB21:/mirror/mpiu/CL4 prog/B-3$ ./a.out
Enter the no. of elements in array  7
Enter the elements of array:
4 5 6 7 8 0 1


The Array is:
4 5 6 7 8 0 1
Using Quick Sort

Thread Number: 0  pivot element selected : 1

Thread Number: 2  pivot element selected : 5

Thread Number: 4  pivot element selected : 7
The Sorted

The Array is:
0 1 4 5 6 7 8
Excecution time : 0.00218175 seconds Would you like to continue? (1/0 y/n)
0
mpiu@DB21:/mirror/mpiu/CL4 prog/B-3$
```




Activities
Thu 11:51 AM
/root/intel/amplxe/projects/vtuneAsg – Intel VTune Amplifier
Project Navigator
/root/intel/amplxe/projects
aarti
abx
dfg
vtune
vtuneAsg
r000ah
vtune_Assg
Advanced Hotspots
Hotspots viewpoint (change)
Intel VTune Amplifier XE 2016
Analysis Target
Analysis Type
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
Grouping:
Call Stack
Function Stack
CPU Time: Total
CPU Time: Self
Effective Time by Utilization
Idle
Poor
Ok
Ideal
Over
Spin Time
Ov.. Time
Instr... Retir... Total
Total
check_match.11335
unmap_vmas
crc16
__memset
finish_task_switch
Highlighted 1 row(s):
Thread
quick (TID:...
CPU Time
Running
Spin and...
Hardware ...


Activities
amplxe-gui
Thu 11:51 AM
ccoew
/root/intel/amplxe/projects/vtuneAsg – Intel VTune Amplifier
Project Navigator
/root/intel/amplxe/projects
aarti
abx
dfg
vtune
vtuneAsg
r000ah
vtune_Assg
Welcome
r000ah
Advanced Hotspots
Hotspots viewpoint (change)
Intel VTune Amplifier XE 2016
Analysis Target
Analysis Type
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
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.
Elapsed Time
Average CPU Usage
Target Utilization
80s
60s
40s
20s
0s
0
1
2
3
4
Idle
Poor
Ok
Ideal
Simultaneously Utilized Logical CPUs
Collection and Platform Info
This section provides information about this collection, including result set size and collection platform data.
Application Command Line: /home/ccoew/quick
User Name: root
Operating System: 3.3.4-5.fc17.x86_64 NAME=Fedora VERSION="17 (Beefy Miracle)" ID=fedora VERSION_ID=17 PRETTY_NAME="Fedora 17 (Beefy Miracle)" ANSI_COLOR="0;34" CPE_NAME="cpe:/o:fedoraproject:fedora:17"
Computer Name: sl2-29
Result Size: 6 MB


Activities
amplxe-gui
Thu 11:41 AM
ccoew
/root/intel/amplxe/projects/vtuneAsg – Intel VTune Amplifier
Project Navigator
/root/intel/amplxe/projects
aarti
abx
dfg
vtune
vtuneAsg
r000ah
vtune_Assg
Welcome
r000ah
Advanced Hotspots
Hotspots viewpoint (change)
Intel VTune Amplifier XE 2016
Analysis Target
Analysis Type
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
Grouping: Function / Call Stack
Function / Call Stack
CPU Time
Effective Time by Utilization
Idle Poor Ok Ideal Over
Spin Time
Ov.. Time
Instructi... Retired
CPI Rate
CPU Fre... Ratio
Module
Function (Full)
Sou.. File
check_match.11335
unmap_vmas
crc16
__memset
finish_task_switch
Selected 1 row(s):
Thread
quick (TID:...
CPU Time
Running
CPU Time
Spin and...
Hardware ...
FILTER
100.0%