CS341: Computer Architecture Lab

Assignment 4
Report

Sudhir Kumar (170050053)

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

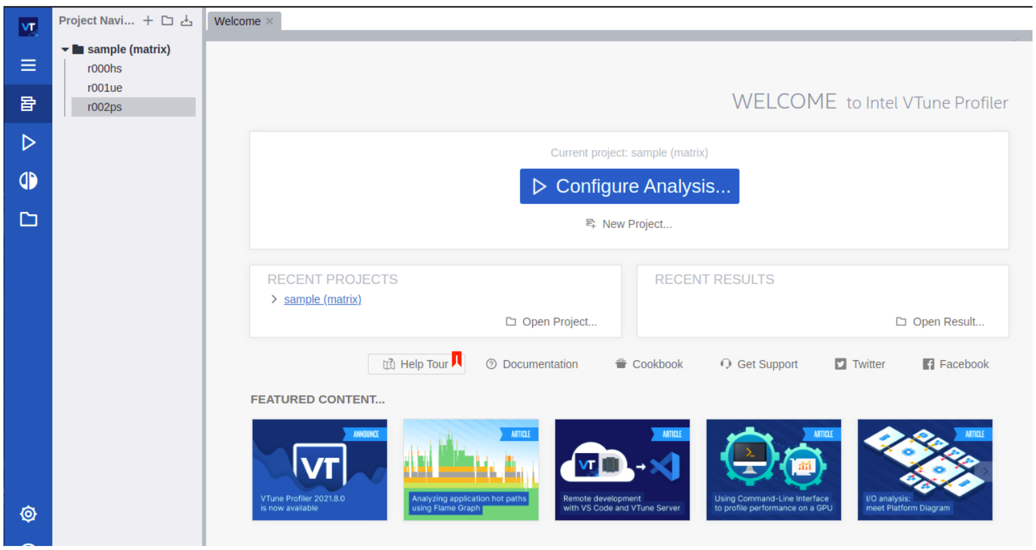
2021-2022

Contents

1. Part 0: Getting Things Ready

1.1 Intel VTune Profiler Installation

followed the guidance according to given link and installed Intel VTune Profiler on ubuntu 20.04.

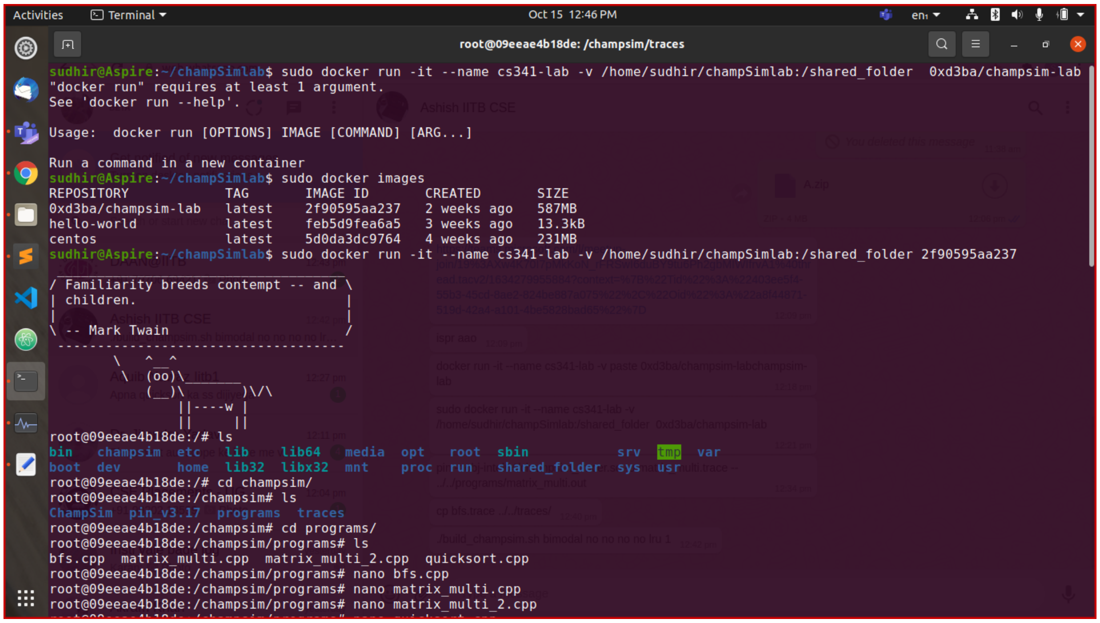


1.2 Challenges during installation

It was not much difficult to install. I just followed all the instructions given at the site provided in problem statements and relative websites. After I downloaded the package followed the instructions of installer. It took me almost 1 hour, 30 minutes to understand, download and install the application. But again it took almost one and half hours to check if I have downloaded correctly and how to open the application. So total of 3 hours for installation and to get it running.

1.3 Docker installation

Installed using guide from internet and pulled 0xd3ba/champsim-lab

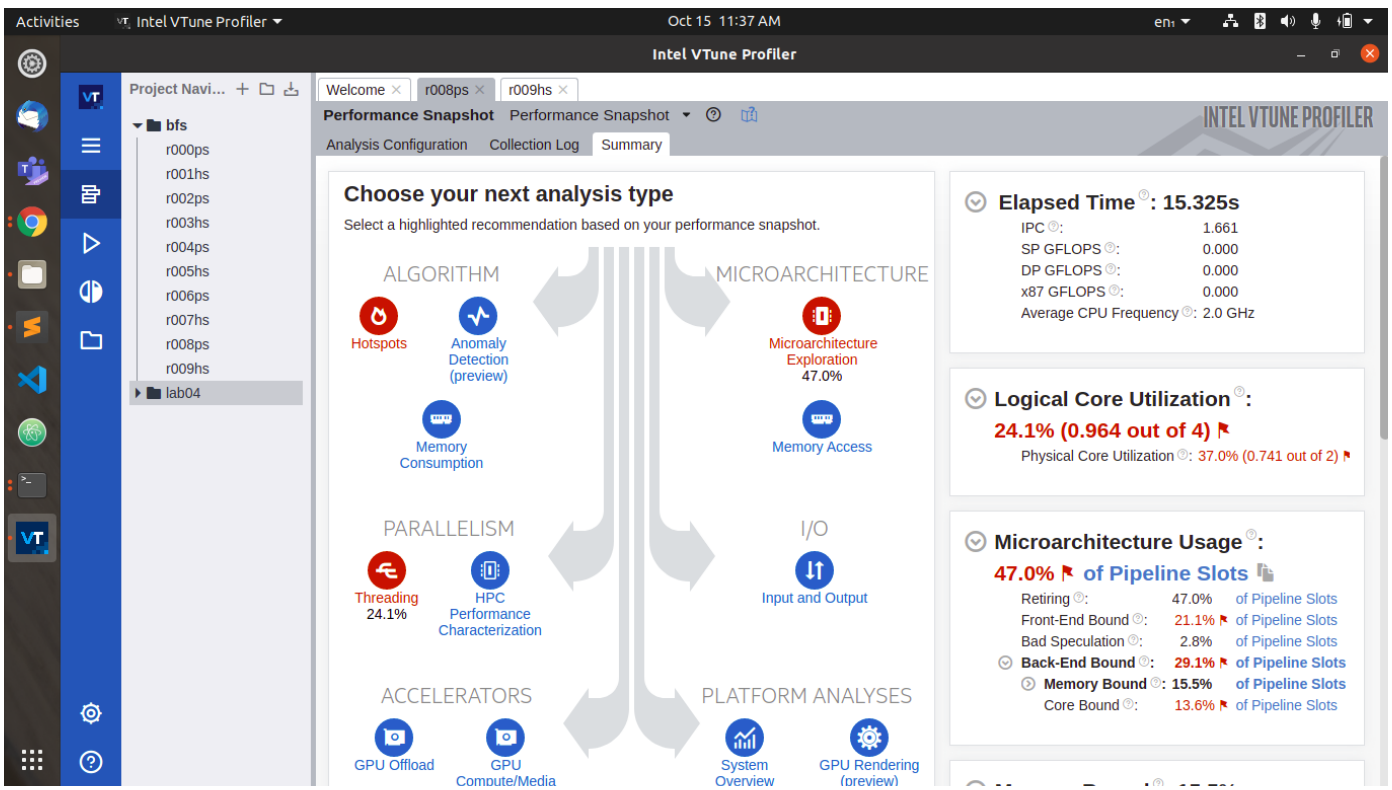

Activities
Terminal
Oct 15 12:46 PM
root@09eeae4b18de: /champsim/traces
sudhir@Aspire:~/champSimlab$ sudo docker run -it --name cs341-lab -v /home/sudhir/champSimlab:/shared_folder 0xd3ba/champsim-lab
"docker run" requires at least 1 argument.
See 'docker run --help'.
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
Run a command in a new container
sudhir@Aspire:~/champSimlab$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
0xd3ba/champsim-lab latest 2f90595aa237 2 weeks ago 587MB
hello-world latest feb5d9fea6a5 3 weeks ago 13.3kB
centos latest 5d0da3dc9764 4 weeks ago 231MB
sudhir@Aspire:~/champSimlab$ sudo docker run -it --name cs341-lab -v /home/sudhir/champSimlab:/shared_folder 2f90595aa237
/ Familiarity breeds contempt -- and \
| children. |
\ -- Mark Twain /
root@09eeae4b18de:/# ls
bin champsim etc lib lib64 media opt root sbin srv tmp var
boot dev home lib32 libx32 mnt proc run shared_folder sys usr
root@09eeae4b18de:/# cd champsim/
root@09eeae4b18de:/champsim# ls
ChampSim pin_v3.17 programs traces
root@09eeae4b18de:/champsim# cd programs/
root@09eeae4b18de:/champsim/programs# ls
bfs.cpp matrix_multi.cpp matrix_multi_2.cpp quicksort.cpp
root@09eeae4b18de:/champsim/programs# nano bfs.cpp
root@09eeae4b18de:/champsim/programs# nano matrix_multi.cpp
root@09eeae4b18de:/champsim/programs# nano matrix_multi_2.cpp

2. Part 1: Profiling with VTune

located provided programs and run with -g -O2 flags to create executable after suitable modification in codes to get execution time as per guided.

2.1 Performance Snapshot

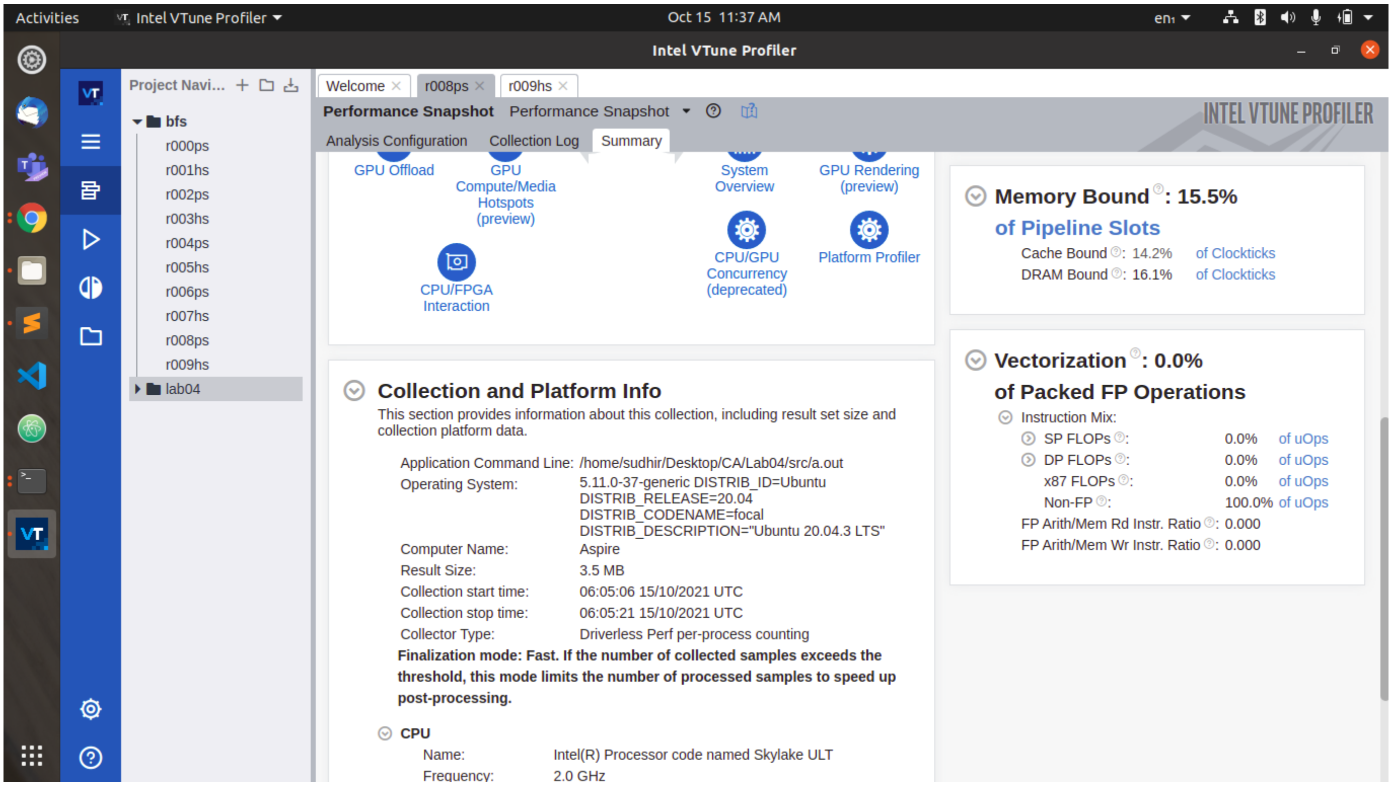
2.1.1 bfs.cpp



IPC = 1.661

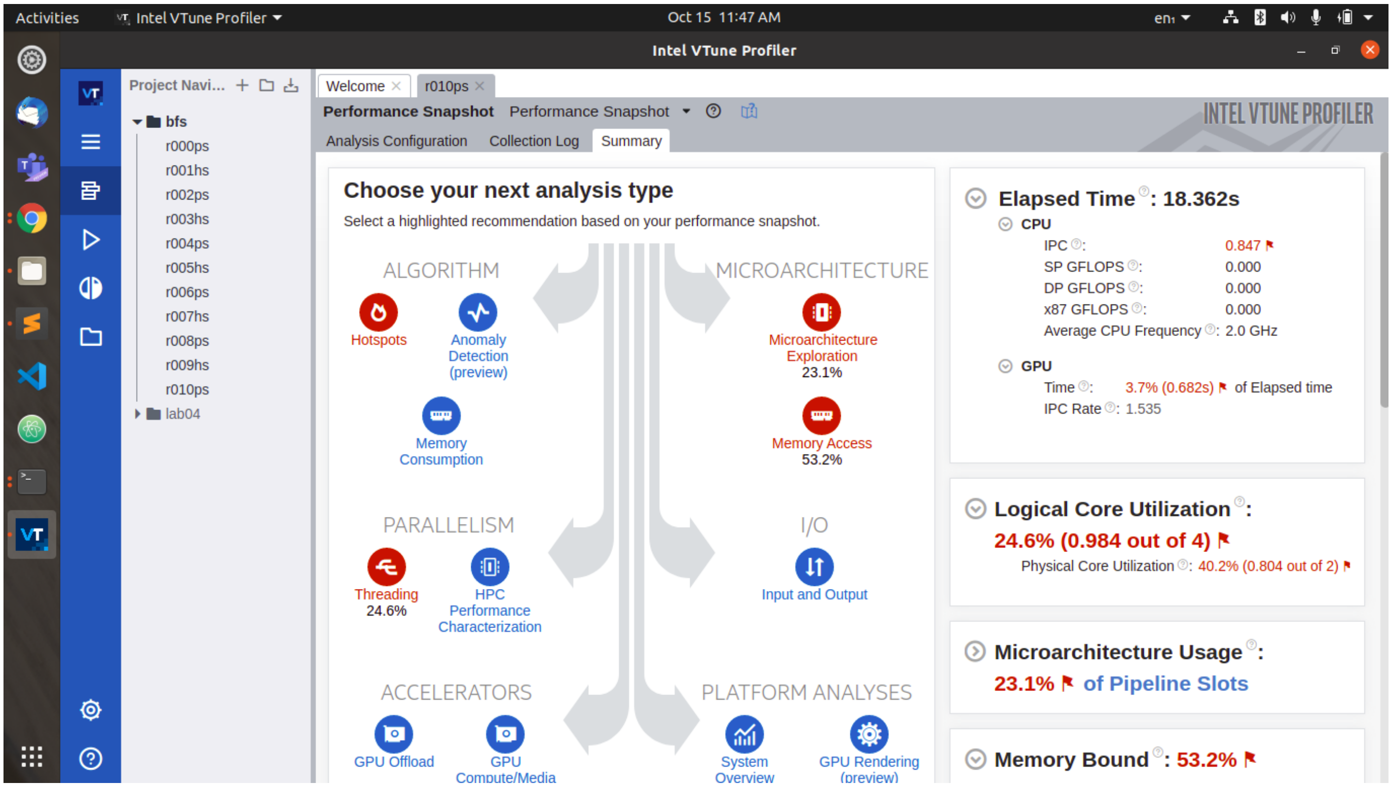
Logical core utilization = 24.1% (0.964 out of 4)

Physical core utilization = 37.0% (0.741 out of 2)



Memory Bound = 15.5 %

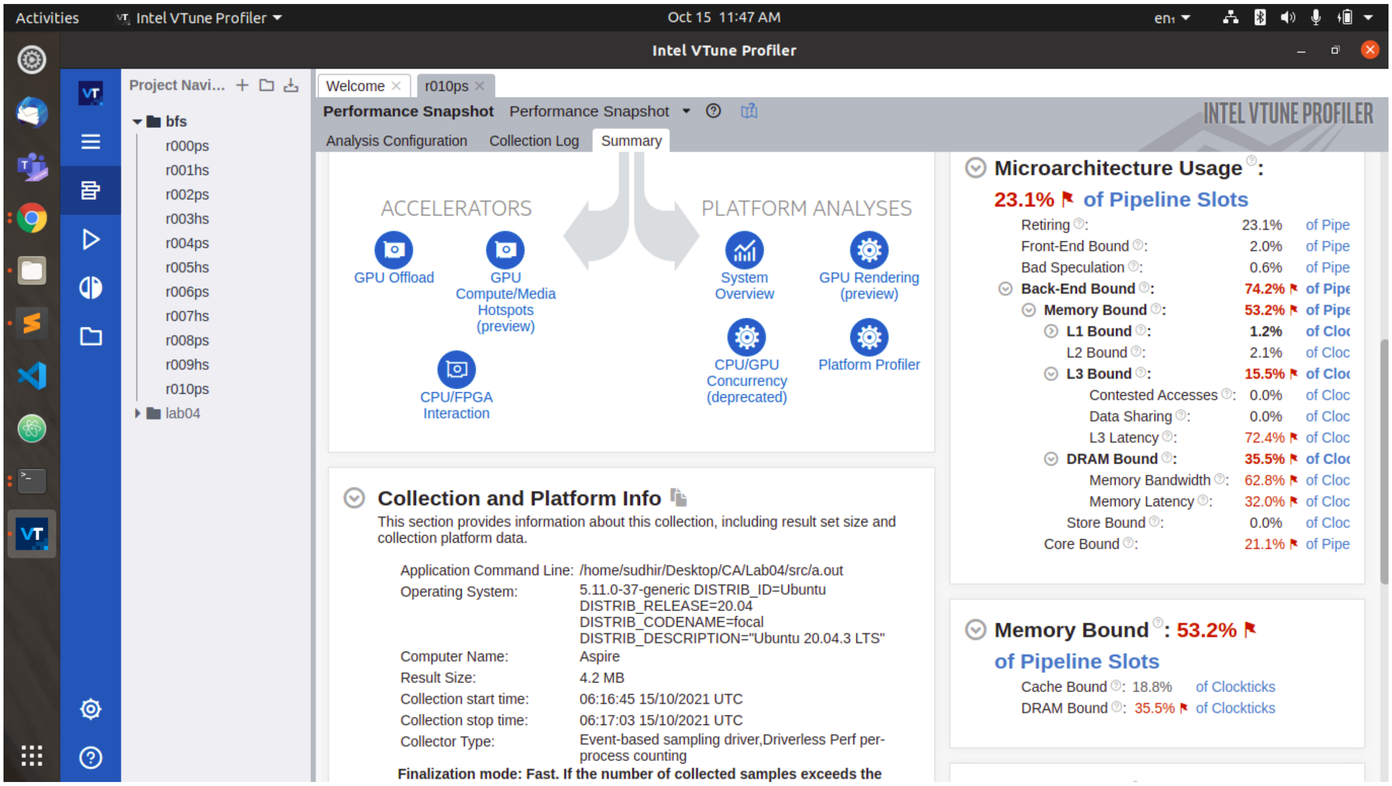
2.1.2 matrix_multi.cpp



IPC = 0.847

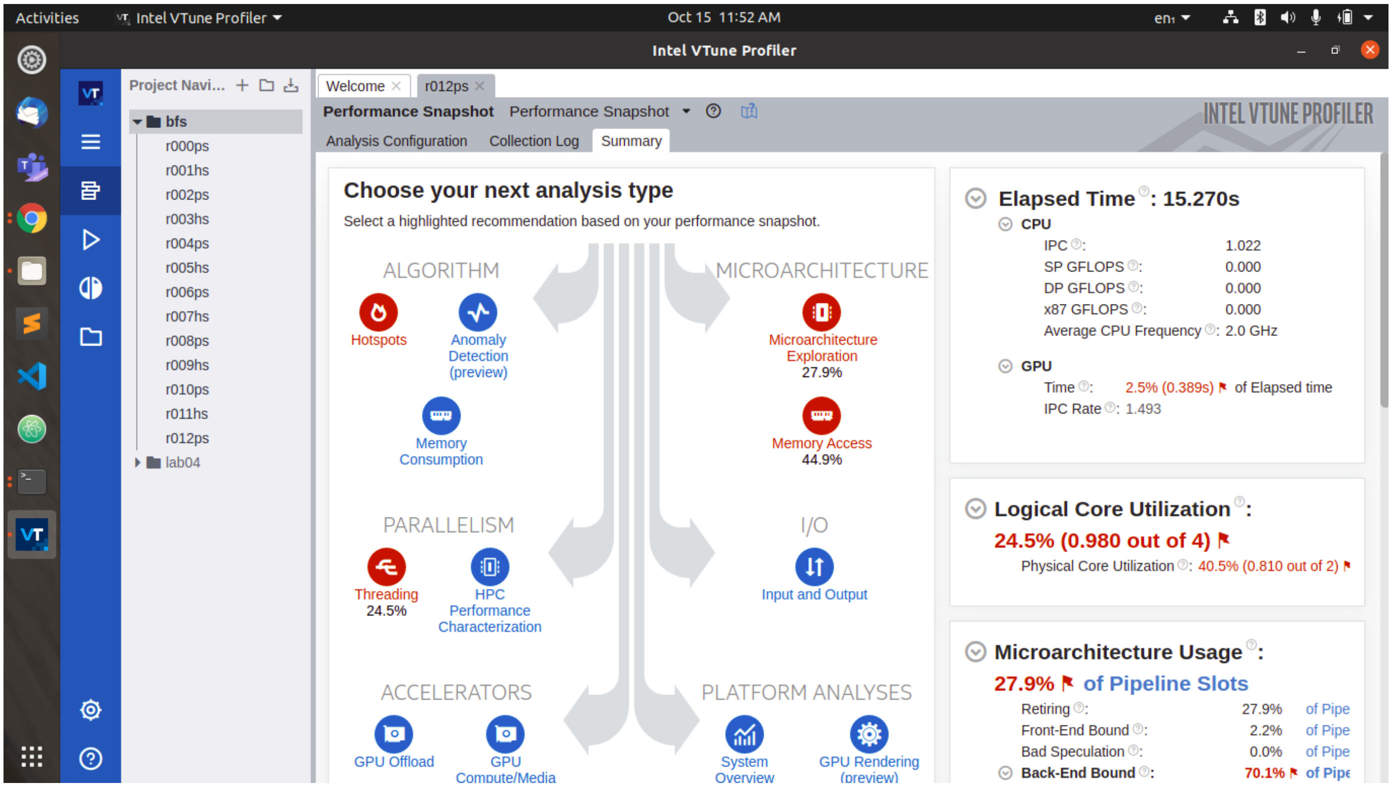
Logical core utilization = 24.6% (0.984 out of 4)

Physical core utilization = 40.2% (0.804 out of 2)



Memory Bound $= 53.2$ %

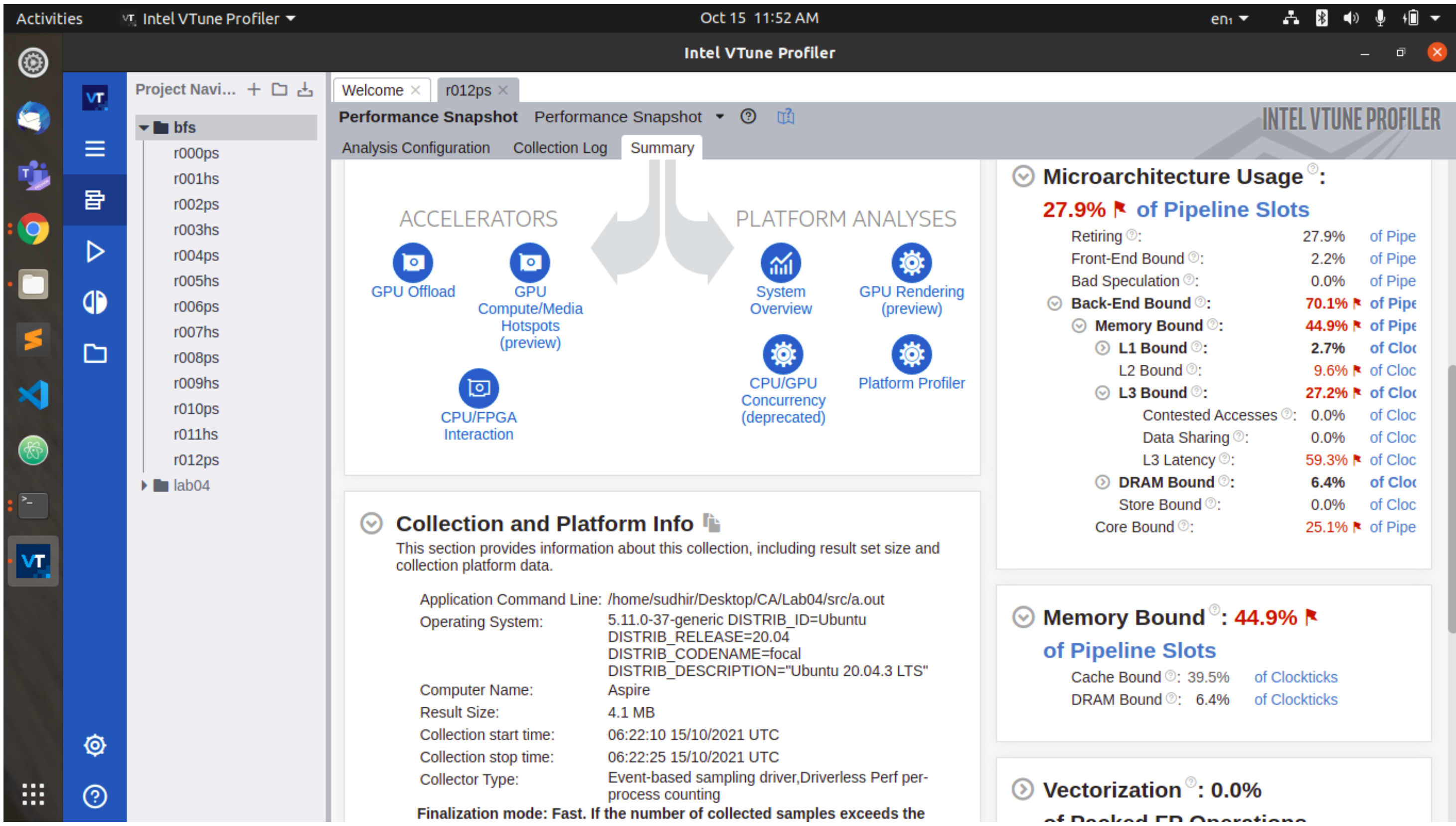
2.1.3 matrix_multi_2.cpp



IPC = 1.022

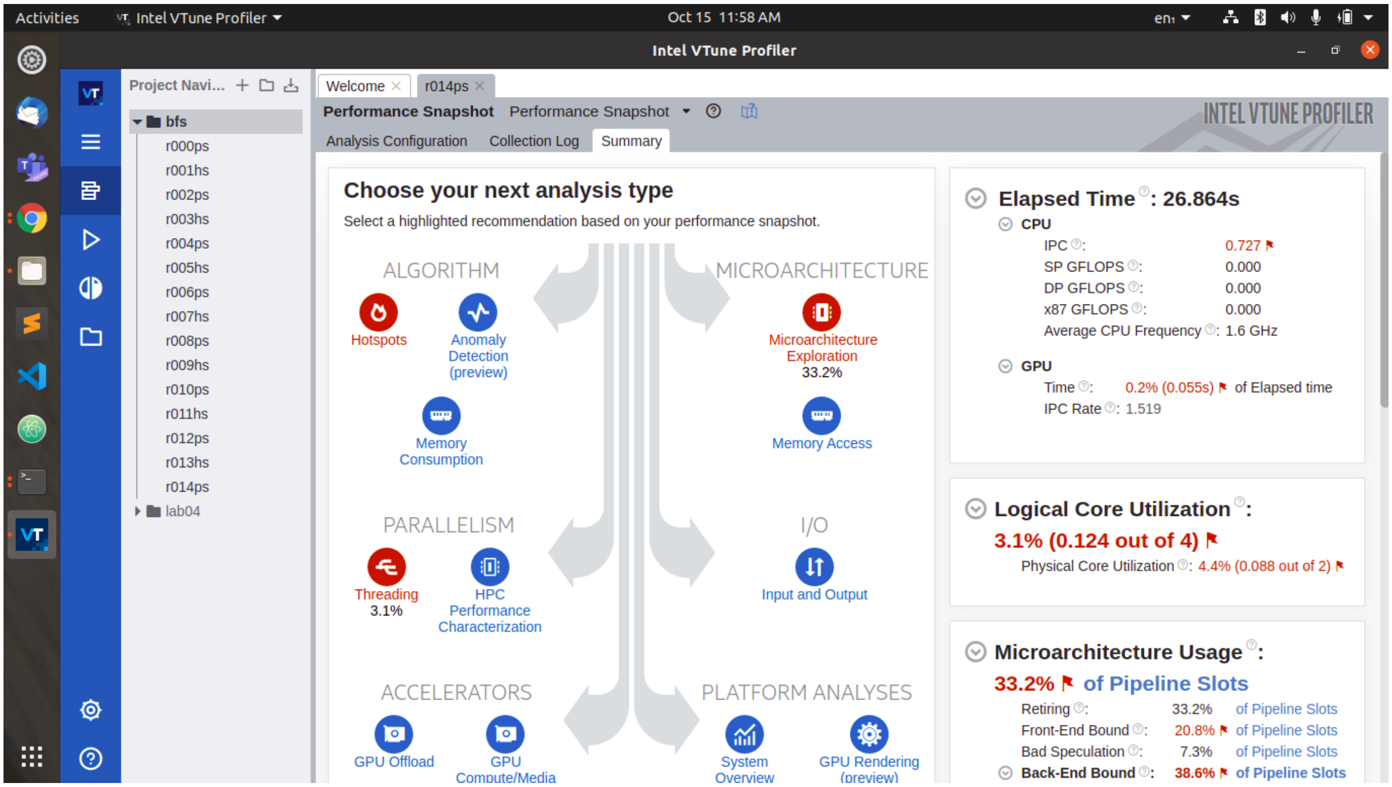
Logical core utilization = 24.5% (0.980 out of 4)

Physical core utilization = 40.5% (0.810 out of 2)

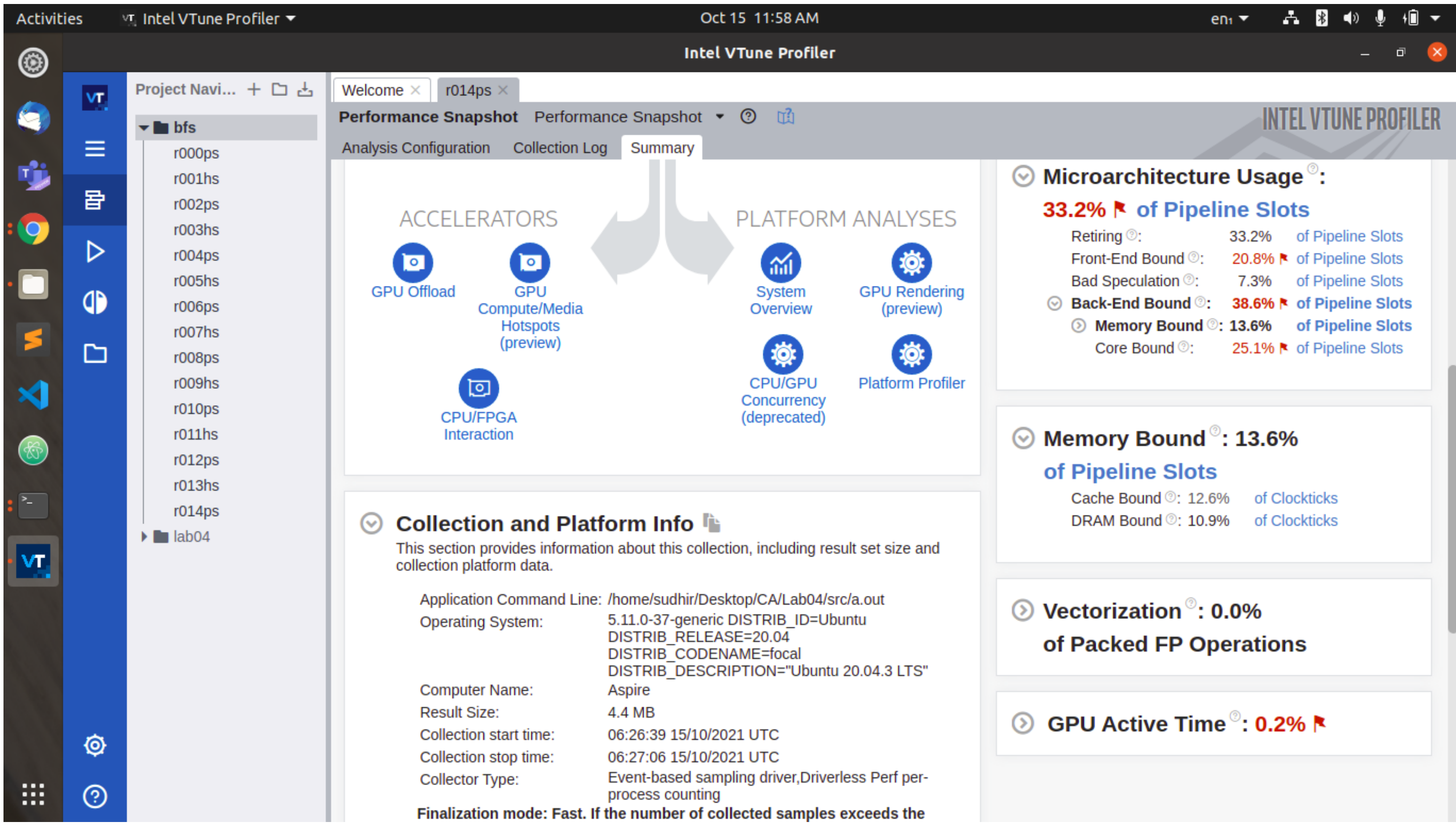


Memory Bound = 44.9 %

2.1.4 quicksort.cpp



IPC = 0.727
Logical core utilization = 3.1% (0.124 out of 4)
Physical core utilization = 4.4% (0.088 out of 2)



Memory Bound = 13.6 %

2.2 Hotspots

2.2.1 bfs.cpp

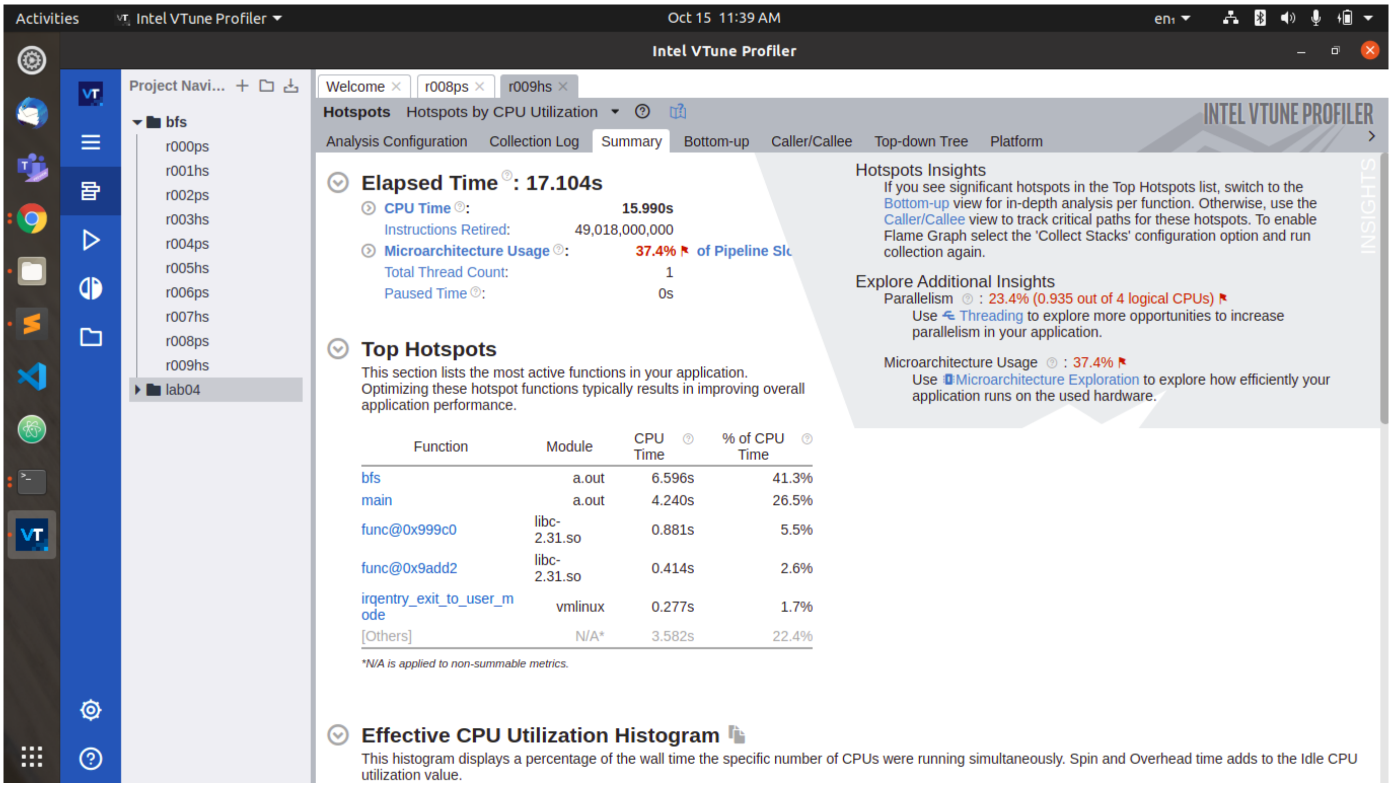


Table 2.1: Top Hotspots

Function	Module	CPU Time	% of CPU Time
bfs	a.out	6.596s	41.3%
main	a.out	4.240s	26.5%
func@0x999c0	libc-2.31.so	0.881s	5.5%
func@0x9add2	libc-2.31.so	0.414s	2.6%
irqentry_exit_to_user_mode	vmlinux	0.277s	1.7%
[Others]	N/A*	3.582s	22.4%

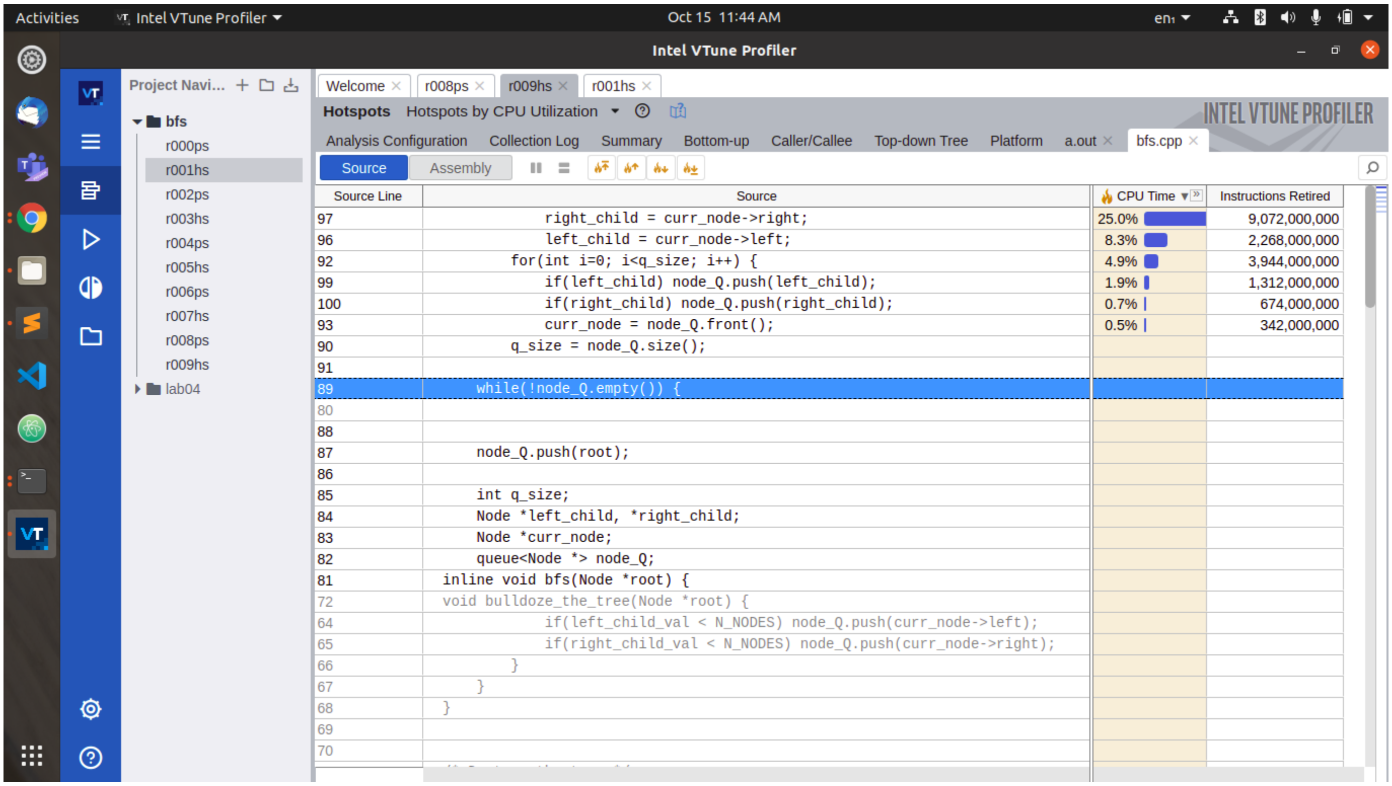


Table 2.2: Statements and % CPU Time

Source	% CPU Time
right_child = curr_node->right;	25.0%
left_child = curr_node->left;	8.3%
for(int i=0; i<q_size; i++) {	4.9%
if(left_child) node_Q.push(left_child);	1.9%
if(right_child) node_Q.push(right_child);	0.7%
curr_node = node_Q.front();	0.5%
bfs(root);	26.5%

2.2.2 matrix_multi.cpp

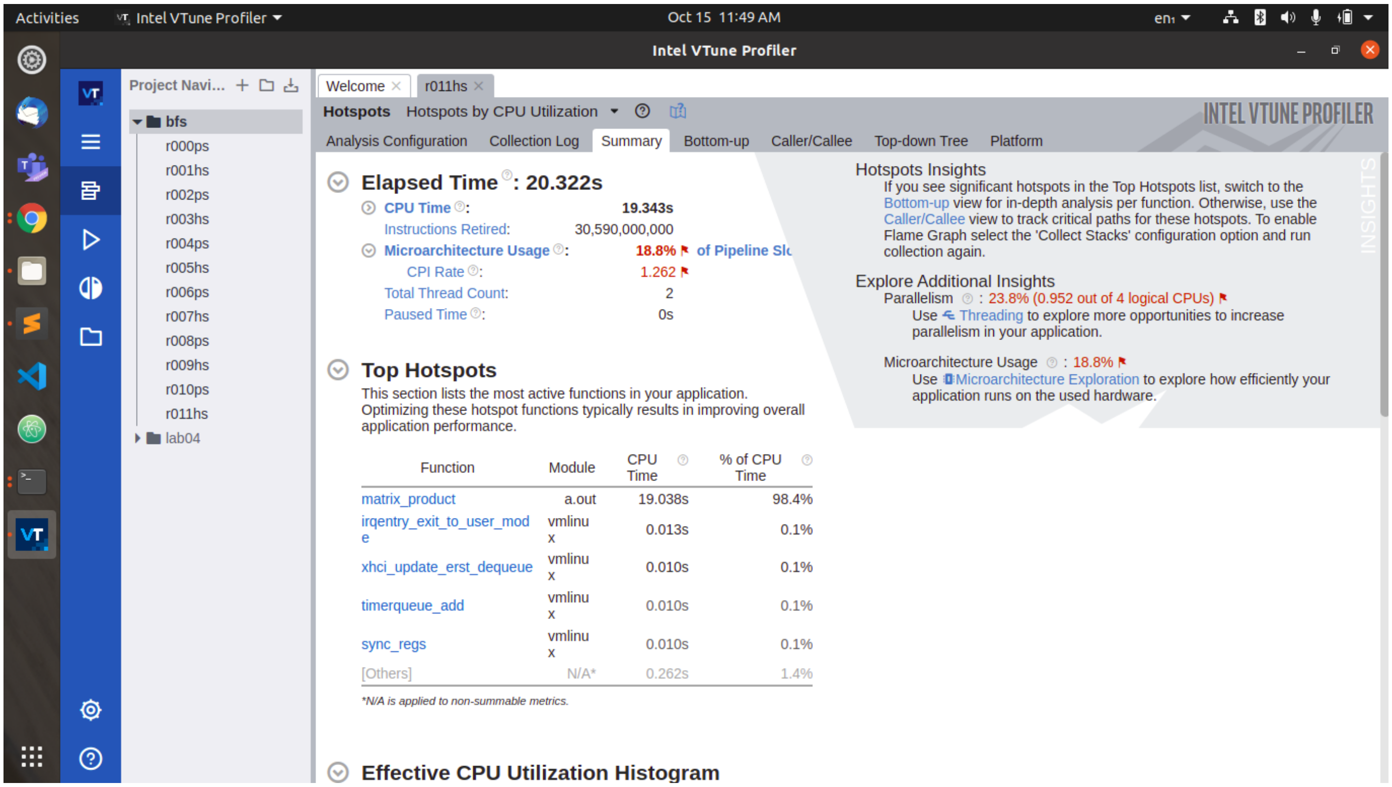


Table 2.3: Top Hotspots

Function	**Module**	**CPU Time**	**% of CPU Time**
matrix_product	a.out	19.038s	98.4%
irqentry_exit_to_user_mode	vmlinux	0.013s	0.1%
xhci_update_erst_dequeue	vmlinux	0.010s	0.1%
timerqueue_add	vmlinux	0.010s	0.1%
sync_regs	vmlinux	0.010s	0.1%
[Others]	N/A*	0.262s	1.4%

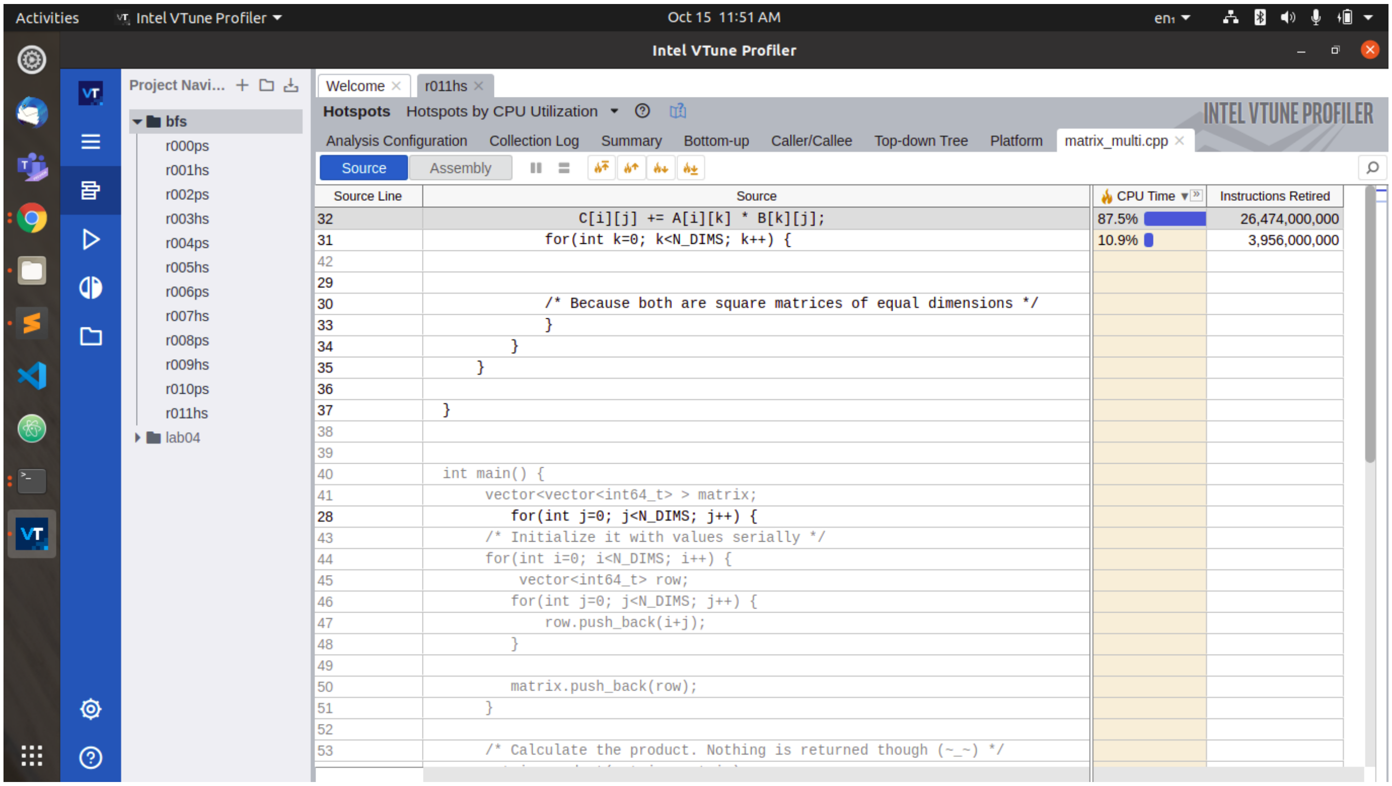


Table 2.4: Statements and % CPU Time

Source	% CPU Time
C[i][j] += A[i][k] * B[k][j];	87.5%
for(int k=0; k<N_DIMS; k++) {	10.9%

2.2.3 matrix_multi_2.cpp

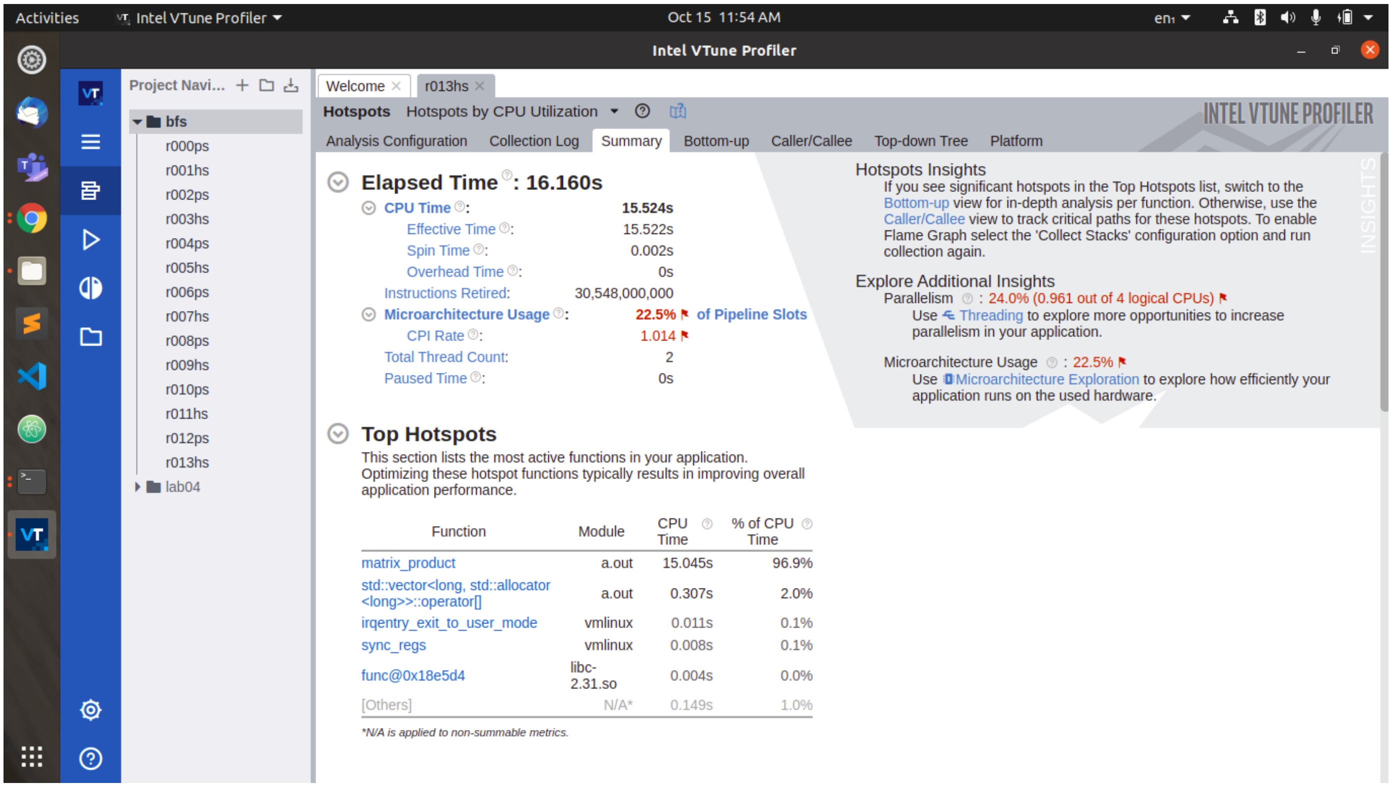


Table 2.5: Top Hotspots

Function	Module	CPU Time	% of CPU Time
matrix_product	a.out	15.045s 96.9%	
std::vector<long, std::allocator<long»::operator[]	a.out	0.307s	2.0%
irqentry_exit_to_user_mode	vmlinux	0.011s	0.1%
sync_regs	vmlinux	0.008s	0.1%
func@0x18e5d4	libc-2.31.so	0.004s	0.0%
[Others]	N/A*	0.149s	1.0%

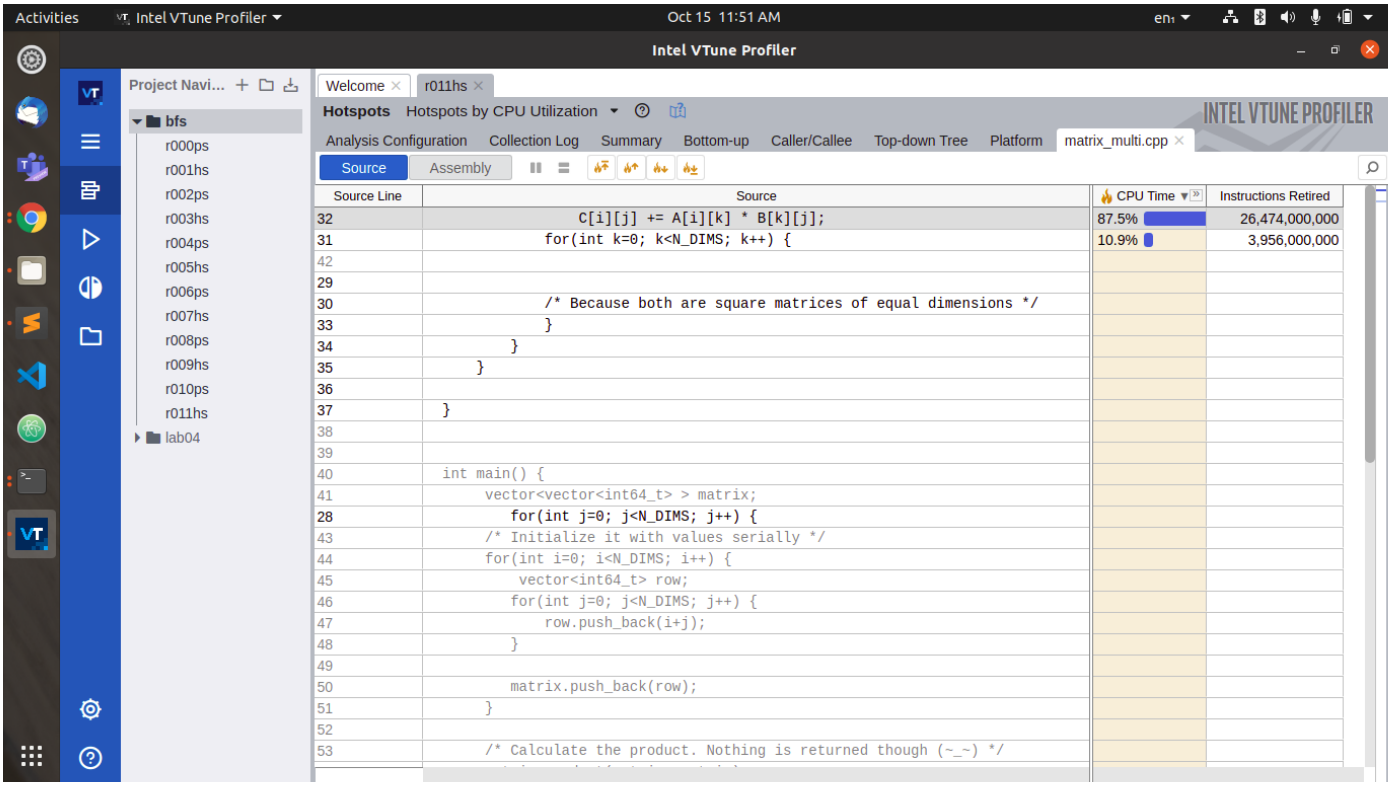


Table 2.6: Statements and % CPU Time

Source	**% CPU Time**
[i][j] += A[i][k] * B[k][j];	82.5%
for(int k=0; k<N_DIMS; k++) {	14.4%
return *(this->_M_impl._M_start + __n);	2.0%

2.2.4 quicksort.cpp

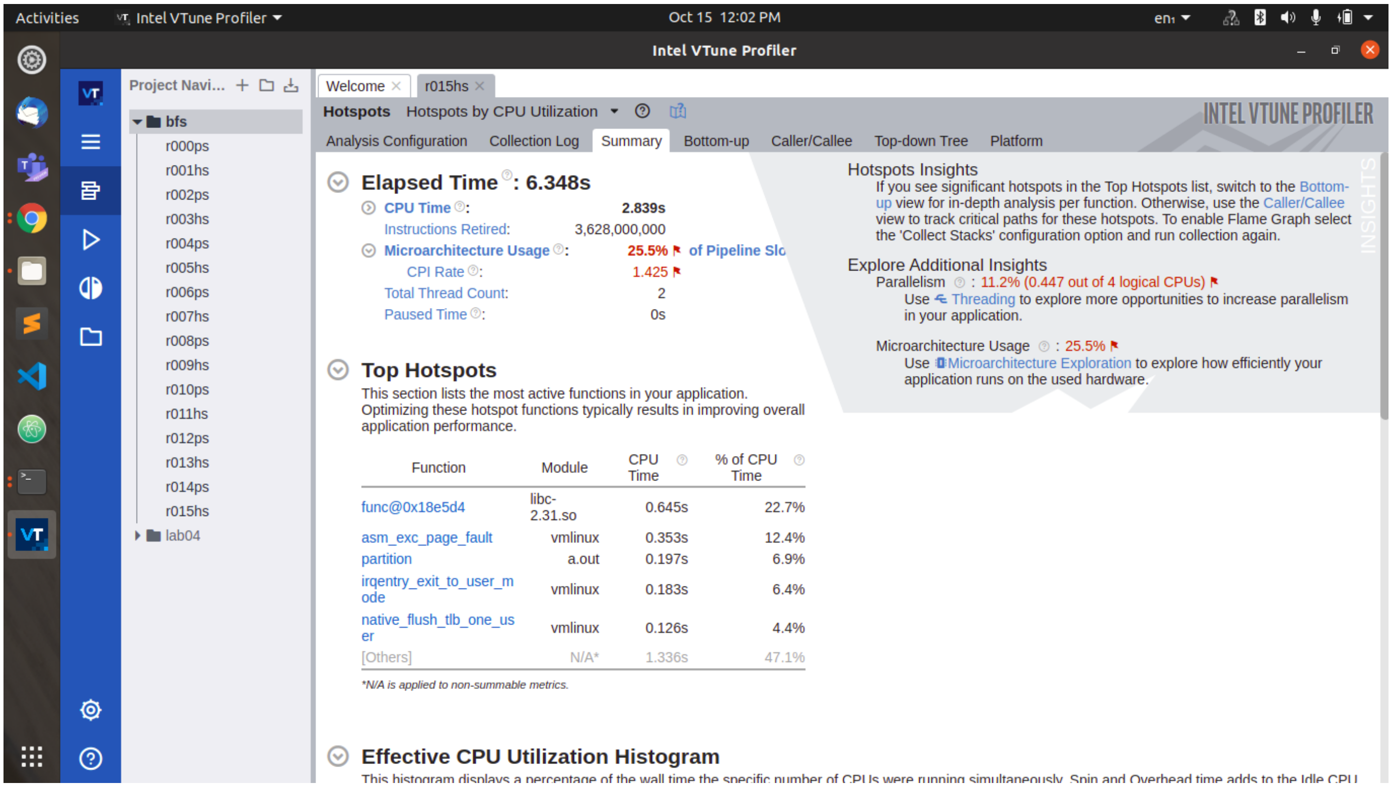


Table 2.7: Top Hotspots

Function	Module	CPU Time	% of CPU Time
func@0x18e5d4	libc-2.31.so	0.645s	22.7%
asm_exc_page_fault	vmlinux	0.353s	12.4%
partition	a.out	0.197s	6.9%
irqentry_exit_to_user_mode	vmlinux	0.183s	6.4%
native_flush_tlb_one_user	vmlinux	0.126s	4.4%
[Others]	N/A*	1.336s	47.1%

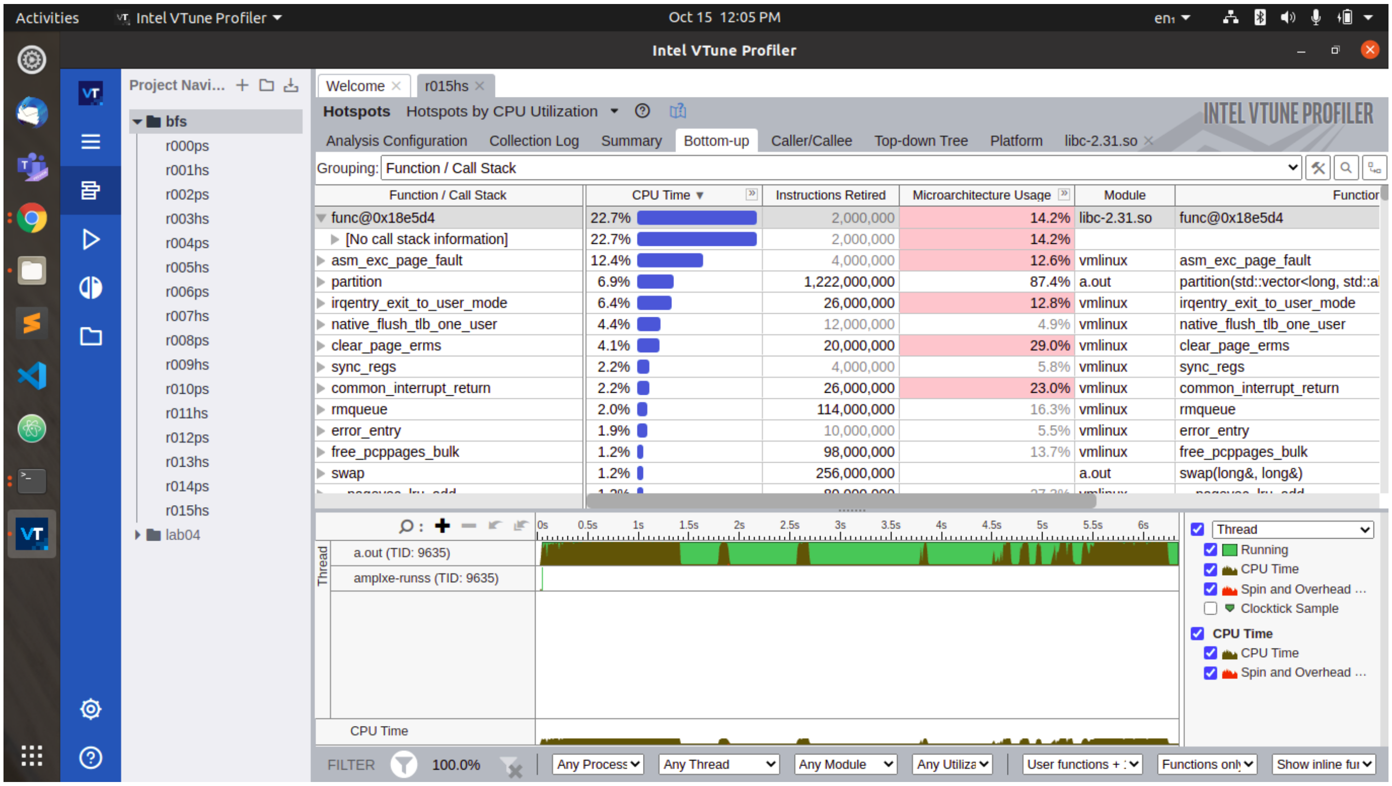


Table 2.8: Statements and % CPU Time

Source	% CPU Time
for(long i=lo; i<hi; i++) {	2.9%
slow_ptr++;	2.3%
if(nums[i] < pivot) {	1.7%

3. Part 2: Simulating with ChampSim

3.1 Preparing the traces

Prepared the traces for each program
for bfs define N_NODES 700000 /* The number of nodes in the tree */
for matrix_multi.cpp and matrix_multi_2.cpp kept define N_DIMS 1500
and for quiksort set define N_ELEM 15000
Moved them to the /champsim/traces/ directory inside the container.

3.2 baseline

prepared four traces (one for each program) for baseline and kept in baseline directory

configuration can be found at beginning of any of trace file
which are
LLC sets: 2048
LLC ways: 16

for BFS
IPC: 1.1329
MPKI: 5.6621

for matrix_multi.cpp
IPC: 1.13602
MPKI: 5.6752

for matrix_multi_2.cpp
IPC: 1.13483
MPKI: 5.6752

for quicksort.cpp
IPC: 1.13238
MPKI: 5.6882

3.3 direct-mapped/: Effect of using Direct-Mapped Cache at all levels

configuration

LLC sets: 32768
LLC ways: 1

for BFS
IPC: 1.09044
MPKI: 5.6621

for matrix_multi.cpp
IPC: 1.09046
MPKI: 5.6752

for matrix_multi_2.cpp
IPC: 1.0834
MPKI: 5.6752

for quicksort.cpp
IPC: 1.05551
MPKI: 5.6882

IPC decreased for for all

In bfs and matrix multiplication, a memory location is not accessed very frequently
in quicksort memory locations are frequently reused this causes more time to be spent in fetching memory hence lower IPC

because of index clash, old data is evicted which leads to more misses

3.4 fully-associative/: Effect of using Fully-Associative Cache at all levels

configuration
LLC sets: 1

LLC ways: 32768

for BFS
IPC: 1.14423
MPKI: 5.6621

for matrix_multi.cpp
IPC: 1.14384
MPKI: 5.6752

for matrix_multi_2.cpp
IPC: 1.1422
MPKI: 5.6752

for quicksort.cpp
IPC: 1.14146
MPKI: 5.6882

The IPC and MPKI are almost same as initial

3.5 reduced-size/: Effect of halving the size of the caches at all levels

configuration
LLC sets: 1024
LLC ways: 16

for BFS
IPC: 1.4236
MPKI: 5.6621

for matrix_multi.cpp
IPC: 1.42255
MPKI: 5.6752

for matrix_multi_2.cpp
IPC: 1.4217
MPKI: 5.6752

for quicksort.cpp
IPC: 1.42177

MPKI: 5.6882

more IPC
MPKI almost same

3.6 doubled-size/: Effect of doubling the size of the caches at all levels

configuration
LLC sets: 4096
LLC ways: 16

for BFS
IPC: 0.886353
MPKI: 5.6621

for matrix_multi.cpp
IPC: 0.885161
MPKI: 5.6752

for matrix_multi_2.cpp
IPC: 0.884669
MPKI: 5.6752

for quicksort.cpp
IPC: 0.884505
MPKI: 5.6882

3.7 doubled-mshr/: Effect of doubling the number of the MSHRs at all levels

configuration
LLC sets: 2048
LLC ways: 16

for BFS
IPC: 1.1329
MPKI: 5.6621

for matrix_multi.cpp
IPC: 1.13602
MPKI: 5.6752

for matrix_multi_2.cpp
IPC: 1.13483
MPKI: 5.6752

for quicksort.cpp
IPC: 1.13238
MPKI: 5.6882

3.8 reduced-mshr/:Effect of halving the number of MSHRs at all levels

configuration
LLC sets: 2048
LLC ways: 16

for BFS
IPC: 1.1329
MPKI: 5.6621

for matrix_multi.cpp
IPC: 1.13602
MPKI: 5.6752

for matrix_multi_2.cpp
IPC: 1.13483
MPKI: 5.6752

for quicksort.cpp
IPC: 1.13238
MPKI: 5.6882