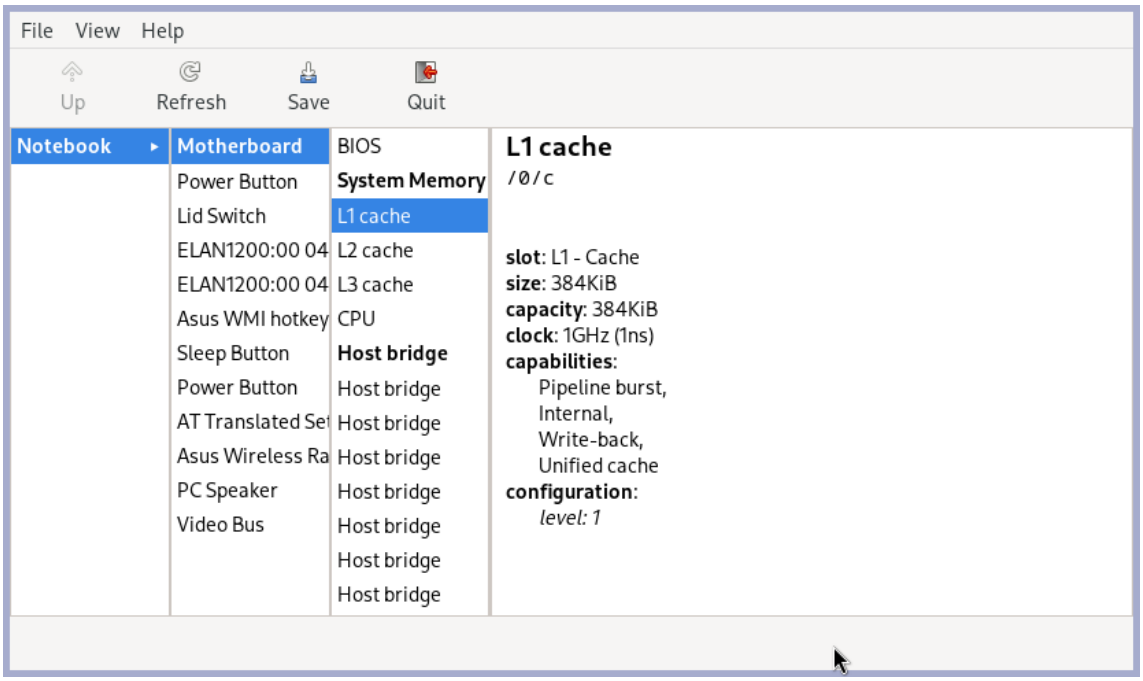
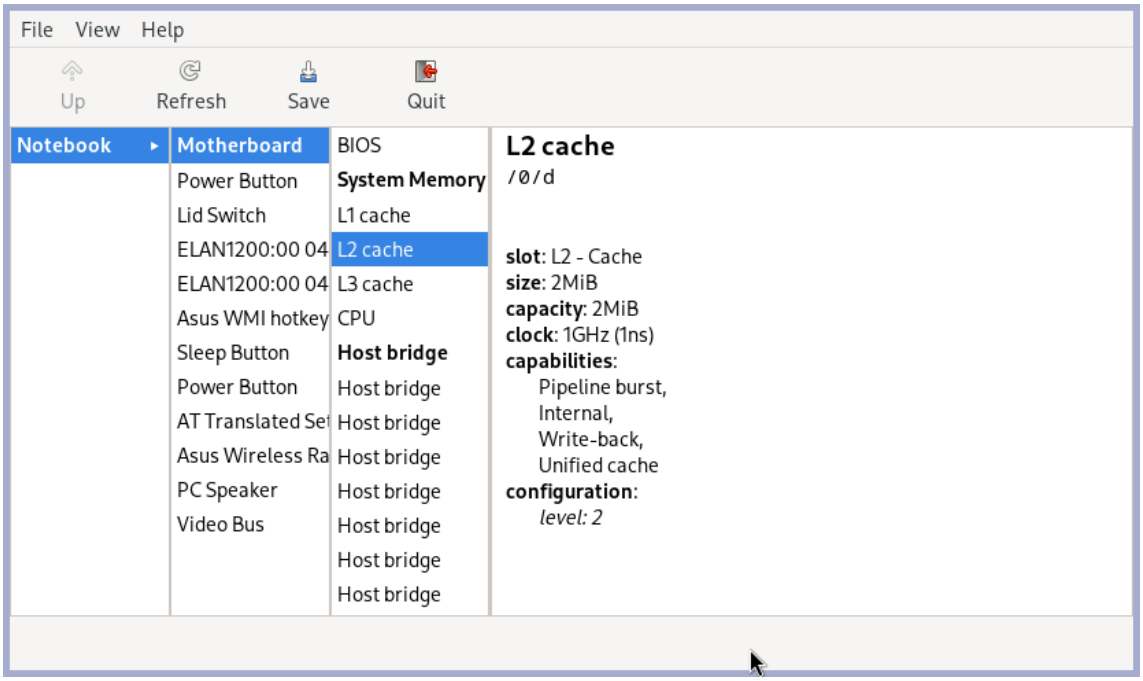
EE2003: Assignment 3

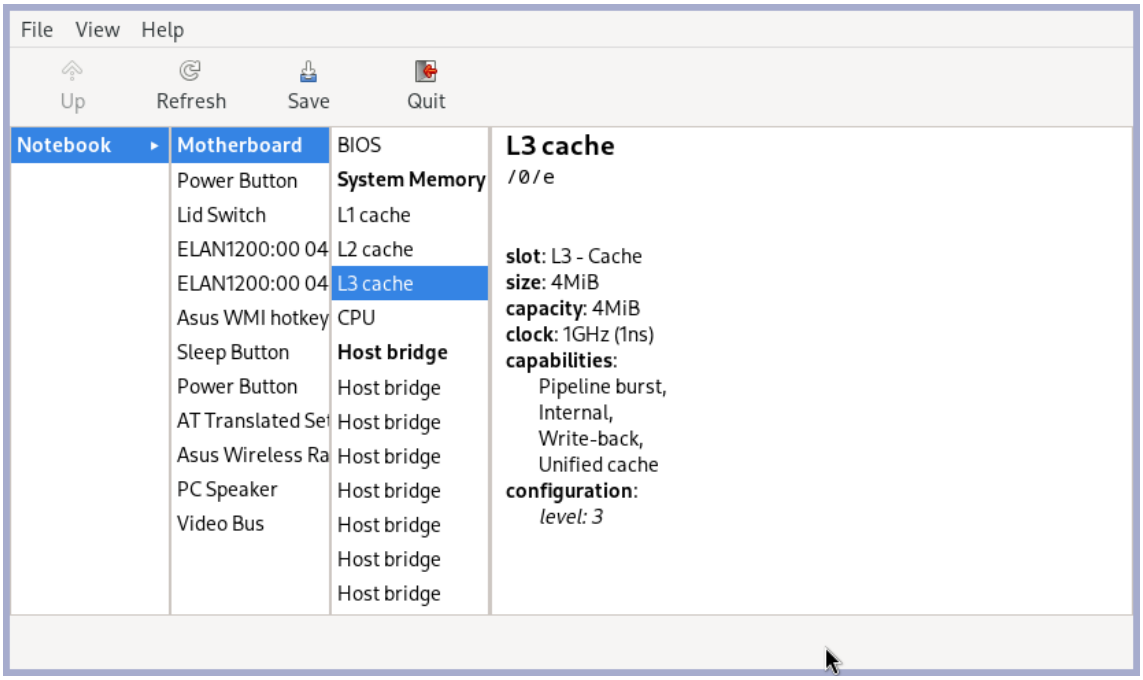
Submitted by Niranjan A. Kartha

My cache configuration

The LSHW command on Linux can be used to see cache configuration:







Thus, my CPU cache is 384 KB, 2 MB and 4 MB for L1, L2, and L3.

Question 8

Plot the TLB Miss Rate by varying the Physical Page size keeping the Virtual Memory size constant (at least 4 different sizes of columns). Also vary the number of levels of tables (at least 2 values including default). Analyze the result.

Procedure used

I have used a bash script (`sim.sh`) that edits the `champsim_config.json` file appropriately for each value required to be simulated, and runs the simulations in parallel.

I then used another script (`process.sh`) to delete all the invalid results (deadlocks, ChampSim errors), and then append the results to a `csv` file (`output.csv`).

Then, I used a Python script (`plot.py`) to plot the graphs and calculate hit rates.

I have attached all the relevant files along with this PDF.

Calculations

The total virtual memory size is given by

```
(pte_page_size / bytes_per_pte)^(num_levels) * page_size
```

`bytes_per_pte` refers to the size of a single page table entry in the page table, and this is a hardcoded constant in ChampSim, given in `inc/vmem.h` :

```
inline constexpr std::size_t PTE_BYTES = 8;
```

I have changed the default `num_levels` to `4` instead of `5` in order to make the calculations easier. The default `page_size` and `pte_page_size` are left at `4096`.

In order to keep the virtual memory size constant, I vary the `pte_page_size` appropriately according to the `page_size` and `num_levels`:

```
pte_page_size = 2^(ceil((48 - log2(page_size))/num_levels) + 3)
```

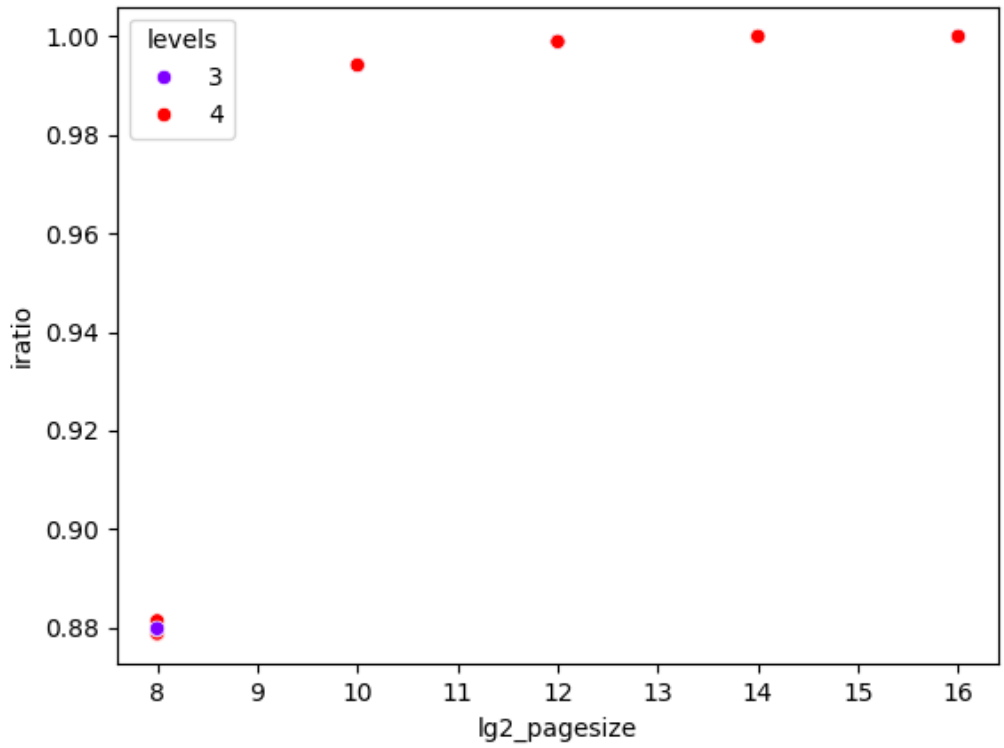
The extra factor of `3` comes from `log2(bytes_per_pte)`.

Results

Page size

Effect on ITLB

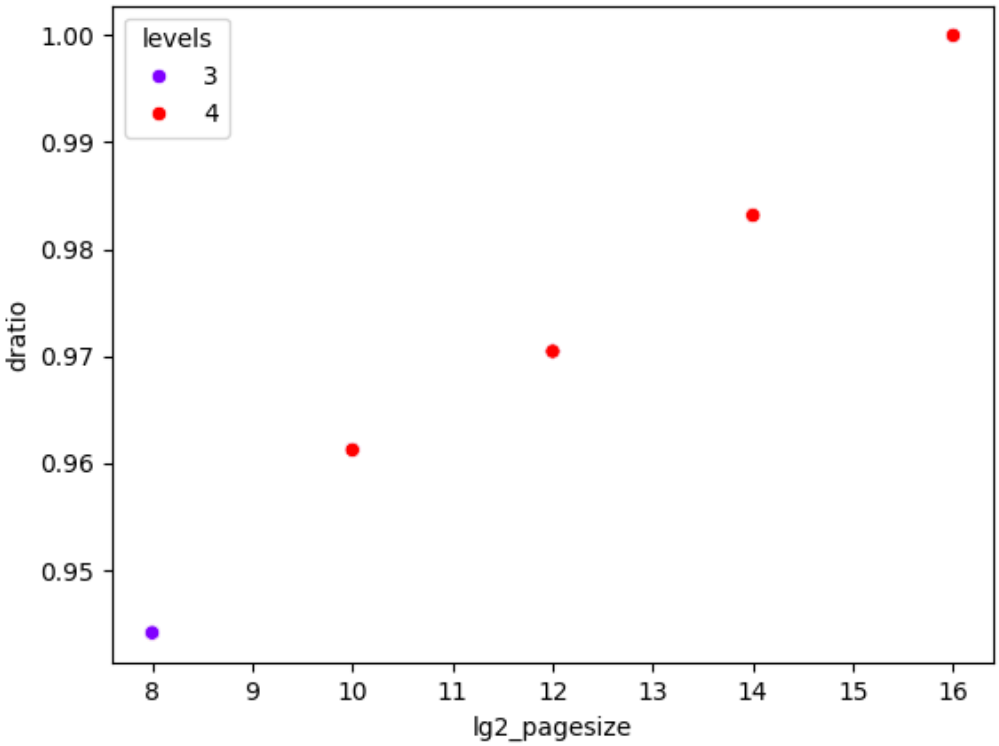
The instruction TLB hit ratio saturates at 1 very quickly as we increase the page size:



This is because the number of instructions is small, and they fit in the same page as we increase the page size.

Effect on DTLB

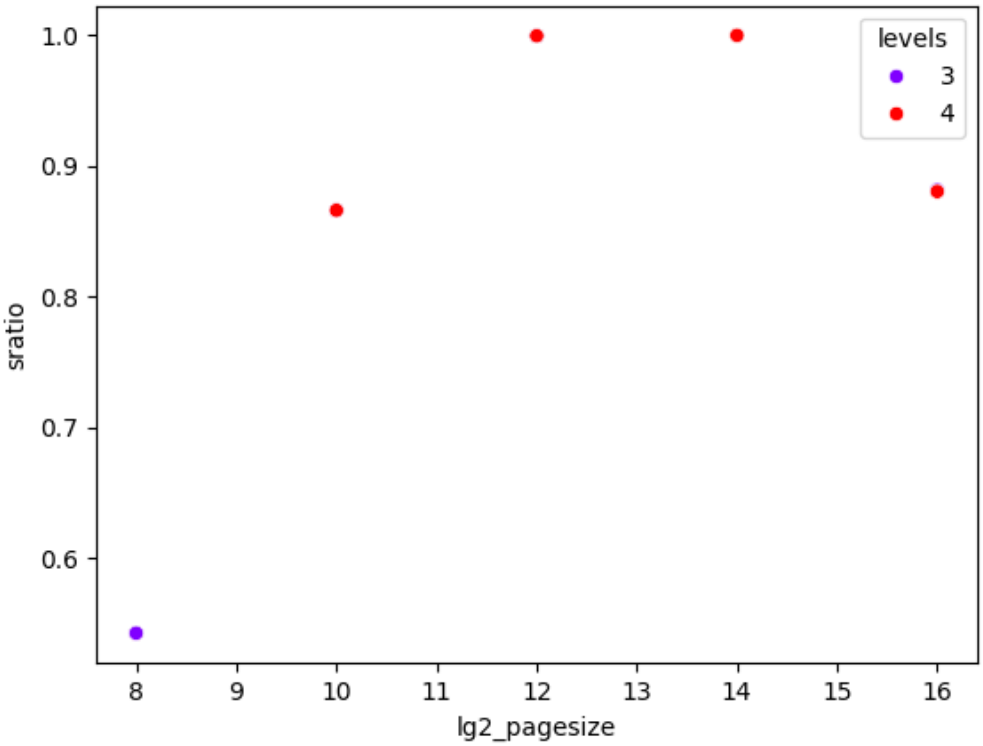
The data TLB hit ratio increases as we increase the page size:



bzip2 works with large amounts of data located nearby each other since it performs a large amount of sorting and memory lookups, and so, as we increase the page size, more of it is present in memory at the same time. This causes the hit ratio to increase as above when we increase the page size.

Effect on STLB

The second-level TLB hit ratio changes as below:



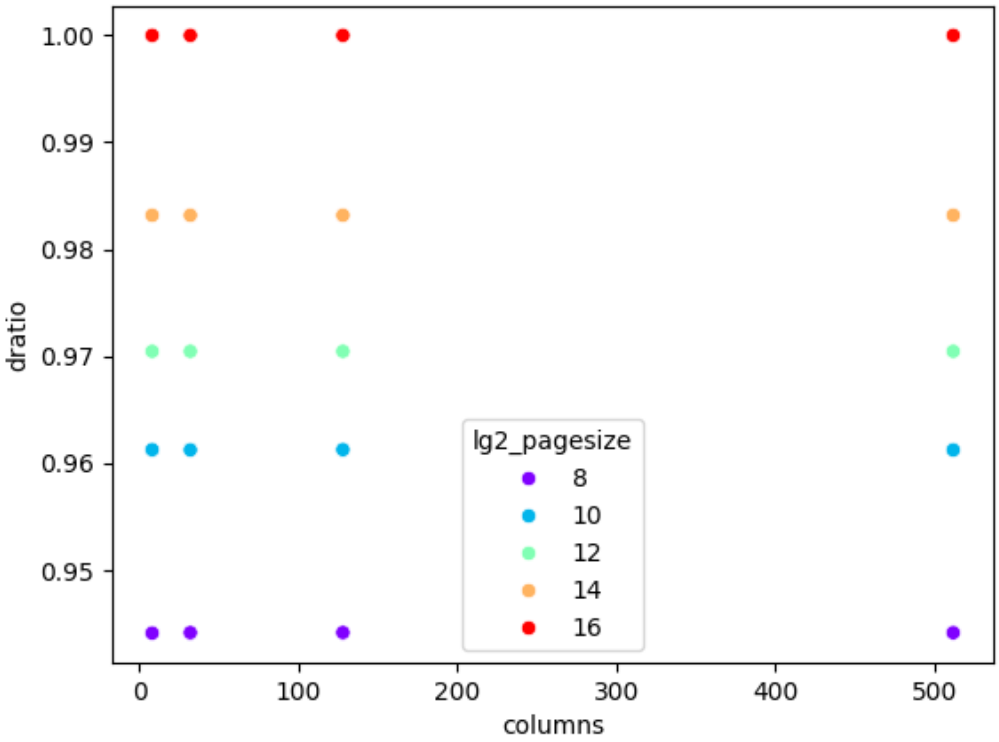
The hit ratio somewhat saturates around `2^14` pages then goes down -- this is because beyond this point, the primary TLB is able to deal with all the repeated memory requests, and the misses come from mostly new page loads, and so the STLB also misses.

Columns

Effect on TLB hit ratio

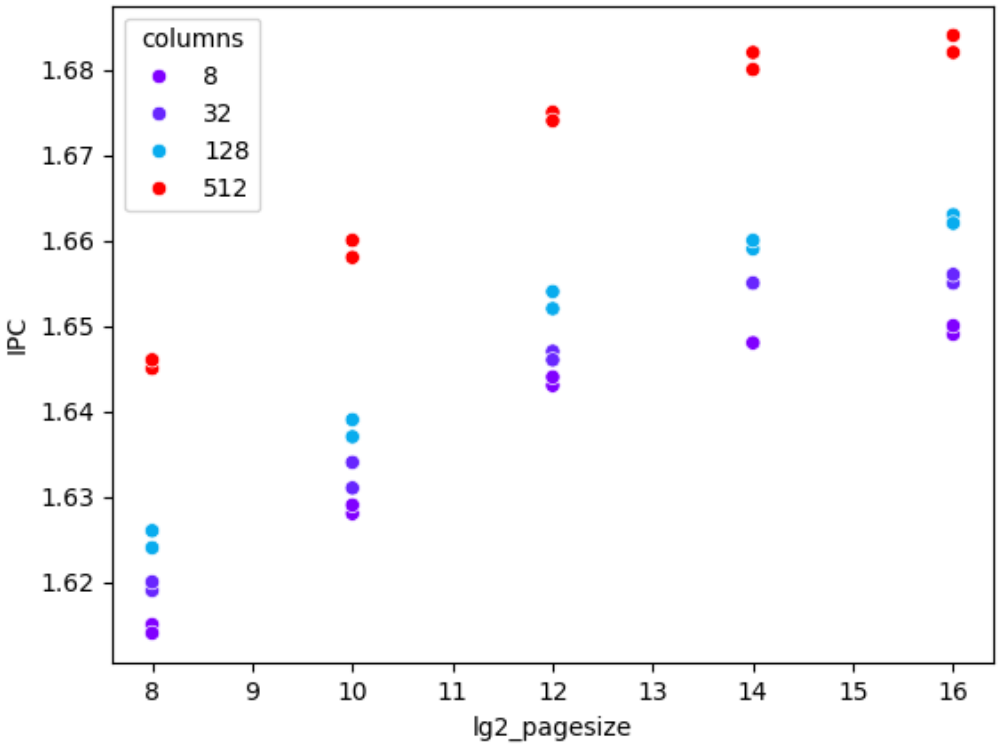
There is no effect on the number of columns on the TLB hit ratio, as the number of columns is purely related to the physical memory, and is abstracted away as we load

virtual pages.



Effect on IPC

As we increase the column size in physical memory, more data is fetched at once, and so, the processing speed is faster:

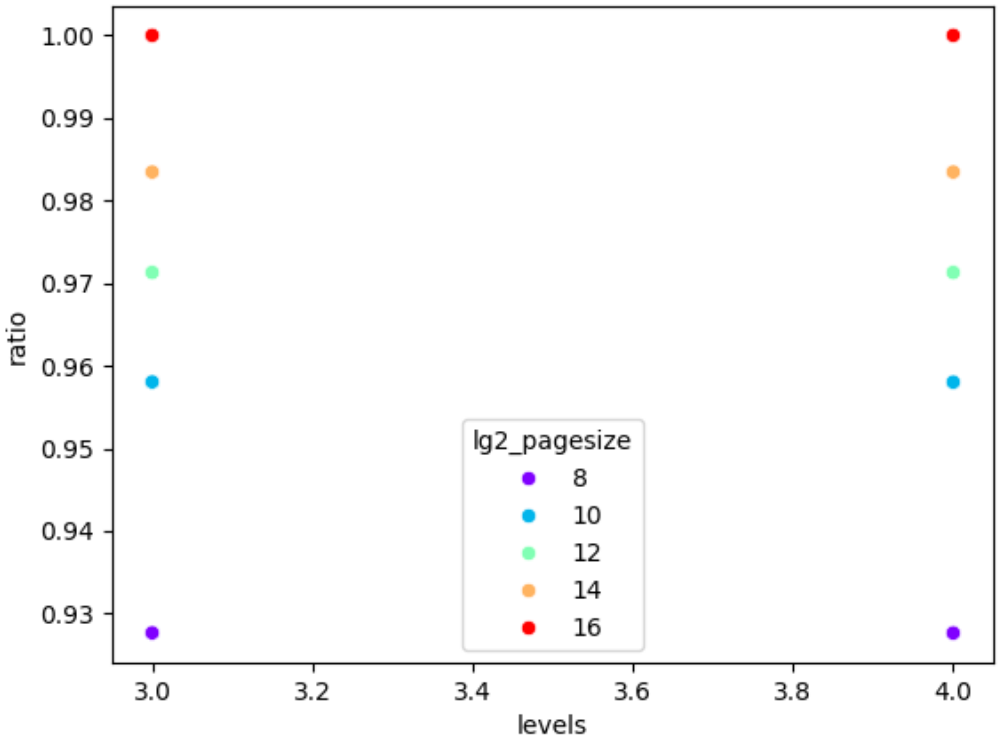


Number of levels

Effect on TLB

The number of levels has no effect on the TLB hit ratios. This is because page-walking is done before the TLB is looked up, and so regardless of how the memory pages are

organized, the TLB is queried the same way in the end.



Effect on IPC

As we decrease the number of levels in the page table, we get a better performance. This is because fewer memory accesses need to be done in order to translate an address in the case of a TLB miss. As we increase the number of levels in the page table, we

save up on memory, but sacrifice on performance.