COP 6726: Database Systems Implementation

Spring 2018

Weekly Assignment 3

02-06-2018

* Virtual Memory System
* Up to Intel 386 processor there was no memory protection and each program would fight for the same memory
* That meant every application had to have a robust system to yield memory… including OS otherwise those memory locations would be locked away.
* OS figured out a new mode called protected mode … where interrupts would be disabled while the protected mode was on.
* 386 supported 4 privilege levels, Linus used Rings 0 and 2 in Linux operating system.
* Linux came around launch of 386 chip so leveraged a lot of the new feature from the first version.
* Introduction of Virtual Memory System provided isolation of memory between applications
* It provided an index that maps virtual memory locations to physical address locations
* Now kernel is the only one which had access to shared memory locations.
* In case one of the processes accessing the shared location tries to change it, kernel allocates a duplicate copy of the memory to make sure that nothing gets changed for another process accessing that shared memory.
* All modern processors support 48 bit VMem. That’s a 256GB limit instead of the 4 GB limit in 32bit Vmem
* Right now there are 4TB RAM solutions out there is industry

02-02-2018

* C++ templates can be problematic, be careful if using those.
* Use git blame plugin and you can see who exactly wrote which line of code.
* Mostly DBs are not in Petabytes, in fact they are in the order of TBs.
* It is kind of quircky that many consumer drives can hold that much data.
  + L1: 16KB instructions + 16KB data
  + L2: 512KB per processor
  + L3: 20-40 MB common data
* L1 has only 64 entries.
* GPGPU are idiot proof as you cannot write code if it didn’t match the architecture.
* If you have hashes which can fit in L2 cache, you would see much performance improvement.
* With Pentium, came new things all together.
  + Pipepline Execution
  + Out of order execution
  + Multiple Execution Units
* Multiple Execution Units:
  + They added 2-3 ALU(Arithmetic and Logical Units) and
  + 2-3 FPU(Floating Point Units)
  + These enabled many instructions to be processed at any time for the CPU
* Out of Order Execution
  + Intel found that even the best most good compiler also would leave the execution units not running, so they just made very smart systems that could see incoming code and run instructions out of order to keep the CPU engaged as much as possible.
* Pipeline Execution
  + They made 20 – 30 stages that meant that at least it felt as the code was moving along faster.
  + Pipeline flush is bad, try to make the code as simple as possible, that ways processor is not working extra hard to see what you did.