Lecture 19: Multicomputers I&II (Apr 12 and 14, 2005) Yap

Apr 12 and 14, 2005

1 ADMIN

- Hw5 (STM) due on today.

2 Review

- April 12:

  Q: Should page tables be stored per process, or per segment or globally?

  A: Depends on which page table we are talking about! For the file system, it is global. For processes without segmentation, we need tables per process to track pages that are loaded in core. For processes with segmentation, it is per segment.

- April 14:

  Q: Suppose Intel's next Pentium X CPU requires 1 trillion gates and each gate occupies an area that is 0.1 micron square. What does the speed of light imply about the maximum clock speed for Pentium X?

  Assume that the CPU must be able to send a signal to any part of the chip in one clock cycle, and other similar simplifications.

  A: Let us do the calculations. Speed of light in a wire is 20cm/nsec. The diameter of the CPU is at least ...

3 Introduction

- In 2000, we saw the arrival of the 1GHz CPU.

  Today, we have reached 4GHz machines.

- Moore's law (complexity of chips double every 18 months) is at work:

  Predictive as well as prescriptive!

- This trend cannot continue indefinitely because of fundamental physical limits.

  But beware of any predictions about where this limit is! History shows that you are likely to be wrong.

- Speed of light:

  30 cm/nsec (in vacuum)

  20 cm/nsec (in copper or optic fibre)

  10 GHz clock means the signal can travel 2cm in a cycle.

- Heat dissipation is another problem

- Let us measure computer speed by "operations per second" (OPS).

  Of course, clock speed is closely correlated with OPS.

  There is one way to obtain more OPS: introduce PARALLELISM!

  ONE WAY: have more than one CPU in a computer, or perhaps use many less-than-full-fledged CPU but independent computing units.

- Many forms of parallel architecture are possible!

  (1) "Superscalar Machines" just means the CPU can execute more than instruction in one clock cycle. Thus, the instruction fetching unit can fetch many instructions at a time, and the decoding unit can decide which instructions can be done in parallel or otherwise pipelined, etc.

  Practically all modern processors are superscalar!

  (2) Vector machines

  – these instructions that can can take a large number of operands for a given operation.

  (3) Multi-CPU systems

  – number of CPU's can go up to a million!

  – MPP (massively parallel processor systems)

  (4) Distributed networks

  (5) Grid computers

  REMARK: "Super computers" just means computers that are capable of achieving a large amount of parallelism.

- DO WE REALLY NEED ALL THIS SPEED?

  APPLICATIONS: weather simulation, oil and mineral exploration, earthquake prediction, computational biology or drug design, molecular simulation, etc.

- In scientific computing, one simple benchmark is ”flops”.

  Current supercomputers can reach teraflops (trillion flops).

  NOTE: Flops is one benchmark of computer speed, and stands for ”FLoating-point Operations Per Second”. Obviously it is biased for numerical/scientific computations.

4 Shared Memory Computers

- Let us begin with a fairly simple classification of multi-CPU computers:

  1. Shared Memory Computers
     –tightly coupled
     –small communication latency
     –no message passing
  2. Local Memory (only) Computers
     –less tightly coupled
     –medium communication latency
     –message passing
  3. Distributed System
     –loosely coupled
     –large communication latency
     –message passing

  NOTE: the range of communication latency is 3 orders of magnitude, which Tanenbaum describes as ranging from 1 day to 3 years.

- Shared Memory Computers (a.k.a. multiprocessor computers)

  – which memory is shared? RAM!

  – shared disk? may be.

  – classification: uniform memory access (UMA) and the non-uniform memory access (NUMA)

  – HOW TO ACHIEVE A GLOBAL SHARED MEMORY?

  (bus, crossbar, multistage switching)

  1. Bus Architecture
     – contention in bus access is the bottleneck
     – local cache is a solution
     – cache lines can be read-only (multiple copies) or read/write (single copy)
     – can further add local memory
     – limitations: ¡ 32 CPU's ...

2. Crossbar Switch
3. Multistage Switch
   - Built out of a 2x2 **switches** (upper and lower inputs, upper and lower outputs).
   - Connect 2^n CPUs to 2^n Memory Units (MUs) in n stages
   - ASSUME the CPU's are numbered 0 to $2^n - 1$ in binary (from top to bottom, in columns). SIMILARLY for the MU's.
     –Routing is from left to right
   - Every stage is the same: there are $N = 2^n$ inputs and $N = 2^n$ outputs.
     –These inputs go into $n/2$ switches (arranged from top to bottom, in a column)
   - There are $n + 1$ stage of routing. All but the last stage has this form:
     –Working from top to bottom: let each of the 2^n input choose the topmost switch with a free upper input.
     –When there is no more free upper input, choose the topmost switch with a free lower input.
     (ALTERNATIVELY: input address s will connect to output address $CSHIFTL(s)$ (circular shift left).
   - The last stage is trivial: just connect the ith input to the ith output.
   - REMARK: the inputs for the first stage are the CPU's, and the output for the last stage are the MU's.
     –The outputs for the ith stage ($i = 1, \ldots, n$) are just inputs for the ith column switches
   - EXAMPLE: $n = 3$ for $N = 8$ processors (DRAW FIGURE)
     $(000) \to (000)$,
     $(001) \to (010)$,
     $(010) \to (100)$,
     $(011) \to (110)$,
     $(100) \to (001)$,
     $(101) \to (011)$,
     $(110) \to (101)$,
     $(111) \to (111)$
   - ROUTING: Suppose the source address is $(s_1 s_2 s_3) = (101)$ and the destinating address is $(d_1 d_2 d_3) = (100)$.
     Initial LOCATION is $(s_1 s_2 s_3) = (101)$.
     After first shuffle, LOCATION is $(s_2 s_3 s_1) = (011)$.
     After first switch, LOCATION is $(s_2 s_3 d_1) = (011)$.
     After second shuffle, LOCATION is $(s_3 d_1 s_2) = (110)$.
     After second switch, LOCATION is $(s_3 d_1 d_2) = (110)$.
     After third shuffle, LOCATION is $(d_1 d_2 s_3) = (101)$.
     After third switch, LOCATION is $(d_1 d_2 d_3) = (100)$.

That is it!!!

- Incidentally, this also constitute a proof that our routine algorithm is correct.
  That is, if you begin with at LOCATION $(s_1 s_2 \cdots s_n)$ and switch using $(d_1 d_2 \cdots d_n)$, you will end up in LOCATION $(d_1 d_2 \cdots d_n)$.
- Omega, banyan, shuffle-exchange, etc are all basically equivalent (can simulate each other in constant time).
- HISTORY: this was used in IBM RP3, BBN Butterfly, NYU's Ultracomputer
- Only need $(N/2)\lg N$ switches for $N = 2^n$ computers
- Unfortunately, this network can block.
  – Proof: there are $N!$ permutations
  – But with $(N/2)\lg N$, we can get only $2^{(N/2)\lg N} = N^{N/2}$ permutations.
  – E.g., $N = 8$, we get only $8^4 = 4096$ permutations out of the $8! = 40,320$ possible.
  – $\ln(N!) = N \ln N - N + \frac{1}{2}\ln(2\pi N) + \theta(1/N)$.
  – $\ln N^{N/2} = (N/2)\ln N$.
  – So there are permutations that cannot be represented!

- Multiprocessor Synchronizations:

  – Recall the TSL (Test and Set Lock) Operation:

  – TSL RX, LOCK

  will first read memory contents of LOCK into register RX and then writes 1 into LOCK

  – Convention: LOCK is shared variable, LOCK=1 means "in use", LOCK=0 means "free"

  – But TSL breaks down in multiprocessor environment.

  – SOLUTION: the TSL operation is now:

  (a) Lock Bus

  (b) Read LOCK into RX

  (c) Write 1 into LOCK

  (d) Unlock Bus

  – Refinements: the cache containing LOCK may be shuffled among the contending CPU's.

  Prefix the above steps (a)-(d) with an attempt to read LOCK. Keep trying until this succeeds.

  Upon success, carry out steps (a)-(d) above.

- Scheduling in Multiprocessor Systems:

1. To Switch or to Wait?

   – in uniprocessor, if we need mutex, just wait for it (either by polling or by joining a queue for the mutex)

   – in multiprocessor, we have a choice: switch processor or wait (as in the uniprocessor case).

   – TRADEOFFS in this choice: waiting wastes CPU cycles, switching is expensive (and may need new cache lines, etc).

   – Solution: track the recent wait (spin) times: if this is small, then waiting may be preferred. Otherwise, switching is preferred.

2. Which Process is Next?

   – We must pick (CPU,Process) pair.

   – Processes may be related

   – Example: in a "parallel make" of many targets, the different processes invoked are related.