Chap. 2 part 1

CIS*3090 Fall 2016
Provocative question

(p30) How much do we need to know about the HW to write good par. prog.?

- Chap. gives HW background knowledge that will allow us to understand:
  - Features of modern processor architectures
  - Be aware of what our par. progs. need to take into account in order to achieve speedup

- My theme: Do we want to get good results by accident or by intention?
Authors’ strategy

First, go down to HW architectural details using 6 representative case studies

- Then, go back “up” to create abstractions that preserve most important characteristics while hiding details → yields 2 **high-level models** of processor architecture

Armed with good model, we can reason about parallel SW, especially par. algos
Learn Flynn’s Taxonomy

- SISD = traditional computer
- SIMD, 2 uses:
  - Instruction level: refers to single inst. that can operate on multiple variables/array elements
  - Program level = SPMD (p49), how MPI operates on SHARCNET cluster
- MIMD (or MPMD) = multiple cores/hosts running individual threads/programs
- MISD $\rightarrow$ rare, HW redundancy (Space Shuttle computer)
6 architectures

Grouped as chip multiprocessors, SMPs, heterogeneous designs, clusters, and supercomputers

- What’s the difference between...
  - “chip MP” (e.g. Intel Core Duo) vs. SMP?
  - homogeneous vs. heterogeneous designs?

Picking out some highlights...
MESI cache coherency protocol (p32)

Great online simulation

Purpose?
- How does it work?
- What’s “snooping”?

Simulator shows how bus congestion can result and how SMP isn’t scalable
- For SMP, how can bus congestion be reduced?
Figure 2.3
HW vs SW multithreading (p35)

SW: OS (or user-level scheduler) loads the single set of registers from thread’s saved context in RAM, upon interrupts and every time it decides to reschedule.

HW aka “hyper-threading” (Intel):
- OS loads *two* sets of registers from two runnable threads; processor switches back and forth between them when executing thread pauses (even just for mem access)
Is hyper-threading a gimmick?

- Far faster context switching
  - Effectively running 2 threads simultaneously (though with single core)
    - OS sees 2 “virtual processors”
  - Cheaper than 2 full-fledged cores and produces less load on memory bus
    - Still overlaps computation with memory bus transactions so increases true concurrency
  - Technique used in Intel Nehalem (our MacPro Core i7) → looks like 16 cores to OS & SW
Crossbar switch (Sun Fire E25K) (Fig 2.5 p36-37)

- Directly connects any pair of 18 origins & 18 destinations by opening/closing switches
- Since doesn’t require routing on a common bus, and multiple connections possible at once, it’s the fastest (& most expensive in HW) kind of interconnect
- Not scalable, due to \( n^2 \) HW resources (”18x18 is pushing the limit”)

Fall 2016
CIS*3090 Parallel Programming
Figure 2.5 Crossbar switch connecting four nodes. Notice the output and input channels; crossing wires do not connect unless a connection is shown. Each pair of nodes is directly connected by setting one of the open circles.
Shocking feature of Cell BE (p38)

◆ SPEs have 256K RAM, for program, stack, heap (if any), data

- Solution to memory bus congestion: DON’T share mem, but have local mems and explicitly move data under program control
- So, don’t use recursion, cut down on library use, get most mileage out of code by using SIMD (vector) instructions
Figure 2.6 Architecture of the Cell processor. The architecture is designed to move data: The high speed I/O controllers have a capacity of 76.8 GB/s; each of the two channels to RAM runs at 12.8 GB/s; the capacity of the EIB is theoretically capable of 204.8 GB/s.
Cell BE pro/con

Strength: several levels of parallelism
- Lowest: SIMD insts
- Multiple SPEs (8 or 16)
- Multiple PPE cores/threads

PPE/SPE communication very complex
- Grad student created solution based on Pilot process/channel approach = CellPilot
- Good for heterogeneous cluster
SHARCNET’s specialty clusters

- Nodes furnished with heterogeneous processors:
  - GPUs, FPGAs, Intel Xeon Phi (new)
  - Gave up on Cell BE

What’s an FPGA?
- What use is it with a general-purpose CPU?

All of these fall into category of G-P CPU + accelerator
BlueGene’s 3 networks (Fig 2.8)

- **How many total processors?**
- **Torus network** is for general-purpose message-passing among processors
- **Collective network** optimized for reduction operation
- **Barrier network** for efficient synchronization
**Figure 2.8** BlueGene/L communication networks; (a) 3D torus for standard interprocessor data transfer; (b) collective network for fast evaluation of reductions.
Summary of HW insights

The “ideal” of shared memory + large no. of processors does not exist!

- Too hard to keep memories coherent without blowing up mem. reference time

Striking different balances/tradeoffs

- Intel “concept chip” Single-Chip Cloud Computer (SCC) with 48 cores, 64GB RAM (shareable), but no cache coherency
- Renesas IMAPCAR2 with 128 cores to execute SIMD insts, or reconfigure as 32 independent PEs without cache coherency → AutoPilot