Contents

Parallel Architectures

- Wide range of parallel architectures
- Same ideas emerge again and again
  - Moore's law means that the components change that we build parallel computers from
  - So the trade-offs between components change
- Different parallel architectures exploit parallelism at different levels

Flynn's Taxonomy

- Prof. Mike Flynn's famous taxonomy of parallel computers
  - Single instruction stream, single data stream
    - VLIW, found in Cell BE
  - Single instruction stream, multiple data streams
    - Vector SIMD, found in Cell BE
  - Multiple instruction streams, single data stream
    - Unusual, sometimes found in safety critical systems, found in Texas Instruments Hercules processors
  - Multiple instruction streams, multiple data streams
    - Multiple parallel core, found in Cell BE
- It is quite difficult to fit many parallel architectures into Flynn's taxonomy
  - It's not entirely clear where some architectures fit
    - Instruction level parallel
    - Fine-grain speculative multithreading
- Most important distinction in between SIMD and MIMD

Instruction Level Parallelism

- Attempt to execute multiple machine instructions from the same instruction stream in parallel
- Two major approaches
  - Get the compiler to find instruction level parallelism
    - VLIW, in-order superscalar
  - Get the hardware to find independent instructions in the running program
    - Out-of-order superscalar

Superscalar is a processor that executes one or more instructions at once

Vector Computers

- Most machine instructions operate on a single value at a time
- Vector computers operate on an array of data with a single instruction
- Many parallel computing problems operate on large arrays of data

- Vector computers were some of the earliest and most successful parallel computers
- Vector instructions are a form of SIMD
  - Very popular on modern computers
  - Accelerate multimedia, games, scientific graphics apps
- Vector sizes are smallish
  - E.g. 16 bytes, four floats - SSE, NEON
  - Intel AVX uses 32 bytes
  - Intel AVX-512 uses 64 bytes
- Vector SIMD used in Cell BE, where all SPE instructions are SIMD

Multithreaded Computers

- Pipelined computers spend a lot of their time stalled
  - Cache misses, branch mispredictions, long-latency instructions
  - Multi-threaded computers attempt to cover the cost of stalls by running multiple threads on the same processor core
  - Require very fast switching between threads
  - Separate register set for each thread
- Three major approaches
  - Switch to a different thread every cycle
  - Switch thread on a major stall
  - Use OoO superscalar hardware to mix instructions from different threads together
- Big problem in software - programming must divide code into threads

Multi-Core Computers

- Power wall, ILP wall and memory wall pushed architecture designers to put multiple cores on a single chip
- Two major approaches
  - Shared memory multi-core
  - Distributed memory multi-core
- Shared memory is usually easier to program, but
  - Hardware does not scale well
  - Shared memory needs cache consistency
- Big problem is dividing software into threads

Shared Memory Multiprocessors

- Shared memory multiprocessors are parallel computers where there are multiple processors sharing a single memory
- Main approaches
  - Symmetric multiprocessors
  - Non uniform memory access
    - Usually a distributed shared memory
- Big problem in software

Distributed Memory Multicomputers

- A system of communicating processors each with their own memory
- Distributed memory scales really well
  - The hardware is simple
- Huge variety of configurations
  - Ring, star, tree, hypercube, mesh
- Programming is arguably even more difficult than for shared memory machines
  - Always need to move data explicitly to where it is needed

Clusters

- A group of loosely coupled computers that work closely and can be viewed as a single machine
- Multiple stand-alone machines connected by a network
- Clusters are a form of distributed memory multicomputers
  - But may support a virtual shared memory using software
- Most big supercomputers these days are clusters
  - It's usually cheaper to build large machines out of thousands of cheap PCs

Graphics Processing Units

- GPUs have very large amounts of parallelism on a single chip
  - Vector processing units
  - Multithreaded processing units
  - Multiple cores
- Originally aimed at graphics rendering
  - Very large memory bandwidth but high memory latency
  - Lots of low-precision floating point units
- Implement graphics pipeline directly
  - Generality was once very limited, but it getting better with each new generation of GPU
- Increasingly programmable
- Trend towards "general purpose computation" (GPGPU)
  - But mapping general purpose applications to GPU architecture is challenging
  - Only certain types of apps run well on GPU
- Special languages are popular on GPGPU

Field Programmable Gate Arrays

- Gate array - there is an array of gates that can be used to implement hardware circuits
- Field programmable - the array can be configured at any time "in the field" (programmed)
- FPGAs can exploit parallelism at many levels
  - Special purpose processing units
  - Fine grain parallelism - circuit is clocked
  - Coarse grain parallelism - FPGA can implement any parallel architecture that fits on the chip
- Seem like magic
  - GPUs like growing plants and eating them
  - FPGAs like feeding them to animals than eating the animal (not very efficient)

Low Power Parallel Processors

- Several big changes in computing
  - Giant computer room with single users: 1950s
  - Mainframe computers with hundreds of users: 1960s
  - Minicomputer with dozens of users: 1970s
  - Microcomputer with single user: 1980s
  - Networks and microcomputer servers: 1990s
  - Continuous network connection: 2000s
  - Mobile internet devices: 2010s
- Power and energy are extremely important for mobile computing
  - Trend towards mobile devices, mobile phones, wearable computing IoT
  - Battery life depends on energy consumption
  - Devices that consume very little energy can really add up
- Parallelism is alternative to faster clock
  - Dynamic power of $capacitance \times frequency \times voltage^2$
  - Significant increase in clock normally requires increase in voltage
- Low power multiple processors
  - Often trade software complexity for hardware complexity
    * Many VLIW processors
    * Instruction-level parallelism under software control
    * Complicated compilers
  - On-chip scratchpad memory instead of caches
    * Caches need logic to find cache line
    * Cache line tags, comparators, replacement logic, etc.
    * Scratchpad memory is fully under software control
    * Data transfers using so-called direct memory access (DMA)
    * More complicated compilers, software
  - Cores do not share on-chip memory
    * Programming must manage each core's scratchpad memory
    * Explicit movement of data between cores