Lecture 21:
Heterogeneous Parallelism and Hardware Specialization

Parallel Computer Architecture and Programming
CMU 15-418/15-618, Spring 2019
Let’s begin this lecture by reminding you…

That we observed in assignment 1 that a well-optimized parallel implementation of a compute-bound application was about 44 times faster than single-threaded C code compiled with gcc -O3, running on the same processor
You need to buy a new computer...
You need to buy a computer system

**Processor A**
- 4 cores
- Each core has sequential performance $P$

**Processor B**
- 16 cores
- Each core has sequential performance $P/2$

All other components of the system are equal.

Which do you pick?
Amdahl’s law revisited

\[ \text{speedup}(f, n) = \frac{1}{(1 - f) + \frac{f}{n}} \]

\[ f = \text{fraction of program that is parallelizable} \]

\[ n = \text{parallel processors} \]

Assumptions:
Parallelizable work distributes perfectly onto \( n \) processors of equal capability
Rewrite Amdahl’s law in terms of resource limits

\[
\text{speedup}(f, n, r) = \frac{1}{\frac{1-f}{\text{perf}(r)} + \frac{f}{\text{perf}(r) \cdot \frac{n}{r}}}
\]

Relative to processor with 1 unit of resources, \( n=1 \).
Assume \( \text{perf}(1) = 1 \)

\( f = \) fraction of program that is parallelizable
\( n = \) total processing resources (e.g., transistors on a chip)
\( r = \) resources dedicated to each processing core,
each of the \( \frac{n}{r} \) cores has sequential performance \( \text{perf}(r) \)

Two examples where \( n=16 \)
\( r_A = 4 \)
\( r_B = 1 \)

[Hill and Marty 08]
**Speedup (relative to n=1)**

X-axis = $r$  (chip with many small cores to left, fewer “fatter” cores to right)
Each line corresponds to a different workload
Each graph plots performance as resource allocation changes, but total chip resources resources resources kept the same (constant $n$ per graph)

$\text{perf}(r)$ modeled as $\sqrt{r}$

[Figure credit: Hill and Marty 08]
Asymmetric set of processing cores

Example: $n=16$
- One core: $r = 4$
- Other 12 cores: $r = 1$

$$\text{speedup}(f, n, r) = \frac{1}{\frac{1-f}{\text{perf}(r)} + \frac{f}{\text{perf}(r)+(n-r)}}$$

(of heterogeneous processor with $n$ resources, relative to uniprocessor with one unit worth of resources, $n=1$)

[HiIl and Marty 08]
### Speedup (relative to n=1)

#### Symmetric Architectures
- **X-axis** gives \( r \) for all cores (many small cores to left, few “fat” cores to right)

#### Asymmetric Architectures
- **X-axis** gives \( r \) for the single “fat” core (assume rest of cores are \( r = 1 \))

[Source: Hill and Marty 08]
Heterogeneous processing

Observation: most “real world” applications have complex workload characteristics *

They have components that can be widely parallelized. And components that are difficult to parallelize.

They have components that are amenable to wide SIMD execution. And components that are not. (divergent control flow)

They have components with predictable data access And components with unpredictable access, but those accesses might cache well.

Idea: the most efficient processor is a heterogeneous mixture of resources (“use the most efficient tool for the job”)

* You will likely make a similar observation during your projects
Example: Intel “Skylake" (2015)
(6th Generation Core i7 architecture)

4 CPU cores + graphics cores + media accelerators
Example: Intel “Skylake” (2015)
(6th Generation Core i7 architecture)

- CPU cores and graphics cores share same memory system
- Also share LLC (L3 cache)
  - Enables, low-latency, high-bandwidth communication between CPU and integrated GPU
- Graphics cores cache coherent with CPU
More heterogeneity: add discrete GPU

Keep discrete (power hungry) GPU turned off unless needed for graphics-intensive applications
Use integrated, low power graphics for basic graphics/window manager/UI
15in Macbook Pro 2011 (two GPUs)

- AMD Radeon HD GPU
- Quad-core Intel Core i7 CPU (contains integrated GPU)

From ifixit.com teardown
Mobile heterogeneous processors

NVIDIA Tegra X1
- Four ARM Cortex A57 CPU cores for applications
- Four low performance (low power) ARM A53 CPU cores
- One Maxwell SMM (256 “CUDA” cores)

Apple A9 (2015)
- > 2 B transistors
- Dual Core 64 bit CPU
- GPU PowerVR GT6700 (6 “core” GPU)

A9 image credit Chipworks, obtained from http://www.anandtech.com/show/9686/the-apple-iphone-6s-and-iphone-6s-plus-review/3
Smartphone Processor

Apple A12, 2018
- 6.9 billion transistors

Processors
- 2 high-power CPUs
  - 7-wide issue
- 4 low-power CPUs
  - 3-wide issue
- 4-core GPU
- Neural engine
  - for deep neural network evaluation

Neural Engine
- Fixed sequence of arithmetic operations
- 8-bit FP
- 8-wide parallelism
- $5 \times 10^{12}$ ops/second

Specialized hardware
- Video encode/decode
- GPS
- Encryption/Decryption
- ...
Supercomputers use heterogeneous processing

Los Alamos National Laboratory: Roadrunner

Fastest US supercomputer in 2008, first to break Petaflop barrier: 1.7 PFLOPS
Unique at the time due to use of two types of processing elements
(IBM’s Cell processor served as “accelerator” to achieve desired compute density)
- 6,480 AMD Opteron dual-core CPUs (12,960 cores)
- 12,970 IBM Cell Processors (1 CPU + 8 accelerator cores per Cell = 116,640 cores)
- 2.4 MWatt (about 2,400 average US homes)
GPU-accelerated supercomputing

- **Oak Ridge Summit**
  - World’s most powerful computer
- **Each Node**
  - 2 IBM 22-core POWER9 processors
  - 6 nVidia Graphics Processing Units
  - 608 GB DRAM
  - 1600 GB Flash
- **Overall**
  - 13MW water cooled
  - $325 M for two machines

* Source: NPR
Intel Xeon Phi (Knights Landing)

- 72 “simple” x86 cores (1.1 Ghz, derived from Intel Atom)
- 16-wide vector instructions (AVX-512), four threads per core
- Targeted as an accelerator for supercomputing applications
### Heterogeneous architectures for supercomputing

Source: Top500.org Fall 2018 rankings

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>System</th>
<th>Cores</th>
<th>Rmax [TFlop/s]</th>
<th>Rpeak [TFlop/s]</th>
<th>Power [kW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DOE/SC/Oak Ridge National Laboratory, United States</td>
<td>Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz NVIDIA Volta GPU, Dual-rail Mellanox EDR Infiniband IBM</td>
<td>2,397,824</td>
<td>143,500.0</td>
<td>200,794.9</td>
<td>9,783</td>
</tr>
<tr>
<td>2</td>
<td>DOE/NNSA/LLNL, United States</td>
<td>Sierra - IBM Power System S922LC, IBM POWER9 22C 3.1GHz NVIDIA Volta GPU, Dual-rail Mellanox EDR Infiniband IBM/NVIDIA/Mellanox</td>
<td>1,572,480</td>
<td>94,640.0</td>
<td>125,712.0</td>
<td>7,438</td>
</tr>
<tr>
<td>3</td>
<td>National Supercomputing Center in Wuxi, China</td>
<td>Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz Sunway NRPC</td>
<td>10,649,600</td>
<td>93,014.6</td>
<td>125,435.9</td>
<td>15,371</td>
</tr>
<tr>
<td>4</td>
<td>National Super Computer Center in Guangzhou, China</td>
<td>Tianhe-2A - THU-VBR-EFP Cluster, Intel Xeon E5-2692v2 12C 2.26GHz, 1T express-z, Matrix-2000 NUDT</td>
<td>4,981,760</td>
<td>61,444.5</td>
<td>100,678.7</td>
<td>18,482</td>
</tr>
<tr>
<td>5</td>
<td>Swiss National Supercomputing Centre (CSCS), Switzerland</td>
<td>Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.4GHz, Aries interconnect, NVIDIA Tesla P100 Cray Inc.</td>
<td>387,872</td>
<td>21,230.0</td>
<td>27,154.3</td>
<td>2,384</td>
</tr>
<tr>
<td>6</td>
<td>DOE/NNSA/LANL/SNL, United States</td>
<td>Trinity - Cray XC40, Xeon E5-2693v3 14C 2.3GHz, Intel Xeon Phi 7250 80C 3.5GHz, Aries interconnect Cray Inc.</td>
<td>979,072</td>
<td>20,158.7</td>
<td>41,461.2</td>
<td>7,578</td>
</tr>
<tr>
<td>7</td>
<td>National Institute of Advanced Industrial Science and Technology (AIST), Japan</td>
<td>Al Bridging Cloud Infrastructure (ABC) - PRIMERGY CX2570 M5, Xeon Gold 6148 20C 2.46GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR Fujitsu</td>
<td>391,680</td>
<td>19,880.0</td>
<td>32,576.6</td>
<td>1,649</td>
</tr>
</tbody>
</table>
**Green500: most energy efficient supercomputers**

Efficiency metric: MFLOPS per Watt

<table>
<thead>
<tr>
<th>Rank</th>
<th>TOP500 Rank</th>
<th>System</th>
<th>Cores</th>
<th>Rmax (TFlop/s)</th>
<th>Power (kW)</th>
<th>Power Efficiency (GFlops/watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>375</td>
<td>Shoubu system B, ZettaScaler-2.2, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2, PEZY Computing / Exascale Inc. Advanced Center for Computing and Communication, RIKEN Japan</td>
<td>953,280</td>
<td>1,063.3</td>
<td>60</td>
<td>17.604</td>
</tr>
<tr>
<td>2</td>
<td>374</td>
<td>DGX SaturnV Volta, NVIDIA DGX-1 Volta36, Xeon E5-2698v4 20C 2.2GHz, Infiniband EDR, NVIDIA Tesla V100, NVIDIA Corporation United States</td>
<td>22,440</td>
<td>1,070.0</td>
<td>97</td>
<td>15.113</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIAVolta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States</td>
<td>2,397,824</td>
<td>143,500.0</td>
<td>9,783</td>
<td>14.668</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>Al Bridging Cloud Infrastructure (ABCI) - PRIMERGY BX2570 ML, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR, Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan</td>
<td>391,680</td>
<td>19,880.0</td>
<td>1,649</td>
<td>14.423</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>TSUBAME3.0 - SGI ICE XA, IP139-SXM2, Xeon E5-2680v4 14C 2.4GHz, Intel Omni-Path, NVIDIA Tesla P100 SXM2, HPE GSIC Center, Tokyo Institute of Technology Japan</td>
<td>135,828</td>
<td>8,125.0</td>
<td>792</td>
<td>13.704</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Sierra - IBM Power System S922L C, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States</td>
<td>1,572,480</td>
<td>94,640.0</td>
<td>7,438</td>
<td>12.723</td>
</tr>
</tbody>
</table>

Source: Green500 Fall 2018 rankings
Research: ARM + GPU Supercomputer

- Observation: the heavy lifting in supercomputing applications is the data-parallel part of workload
  - Less need for “beefy” sequential performance cores
- Idea: build supercomputer out of power-efficient building blocks
  - ARM CPUs (for control/scheduling) + GPU cores (primary compute engine)
- Project underway at Barcelona Supercomputing Center
  - [www.montblanc-project.eu](http://www.montblanc-project.eu)  CUDA GPU  Tegra ARM CPU
Energy-constrained computing

- Supercomputers are energy constrained
  - Due to shear scale
  - Overall cost to operate (power for machine and for cooling)
- Datacenters are energy constrained
  - Reduce cost of cooling
  - Reduce physical space requirements
- Mobile devices are energy constrained
  - Limited battery life
  - Heat dissipation
Limits on chip power consumption

- General mobile processing rule: the longer a task runs the less power it can use
  - Processor's power consumption is limited by heat generated (efficiency is required for more than just maximizing battery life)

- Electrical limit: max power that can be supplied to chip

- Die temp: (junction temp -- Tj): chip becomes unreliable above this temp
  (chip can run at high power for short period of time until chip heats to Tj)

- Case temp: mobile device gets too hot for user to comfortably hold
  (chip is at suitable operating temp, but heat is dissipating into case)

- Battery life: chip and case are cool, but want to reduce power consumption to sustain long battery life for given task

Slide credit: adopted from original slide from M. Shebanow: HPG 2013 keynote

- iPhone 6 battery: 7 watt-hours
- 9.7in iPad Pro battery: 28 watt-hours
- 15in Macbook Pro: 99 watt-hours
Efficiency benefits of compute specialization

- Rules of thumb: compared to high-quality C code on CPU...

- Throughput-maximized processor architectures: e.g., GPU cores
  - Approximately 10x improvement in perf/watt
  - Assuming code maps well to wide data-parallel execution and is compute bound

- Fixed-function ASIC (“application-specific integrated circuit”)
  - Can approach 100-1000x or greater improvement in perf/watt
  - Assuming code is compute bound and is not floating-point math

[Source: Chung et al. 2010, Dally 08]
[Figure credit Eric Chung]
Hardware specialization increases efficiency

**Area-normalized FFT Performance (40nm)**

- Core i7
- LX760
- GTX285
- GTX480
- ASIC

ASIC delivers same performance as one CPU core with ~ 1/1000th the chip area.

**FFT Energy Efficiency (40nm)**

ASIC delivers same performance as one CPU core with only ~ 1/100th the power.

---

[Chung et al. MICRO 2010]
Benefits of increasing efficiency

- **Run faster for a fixed period of time**
  - Run at higher clock, use more cores (reduce latency of critical task)
  - Do more at once

- **Run at a fixed level of performance for longer**
  - e.g., video playback
  - Achieve “always-on” functionality that was previously impossible

---

**iPhone:**
Siri activated by button press or holding phone up to ear

**Moto X:**
Always listening for “ok, google now”
Device contains ASIC for detecting this audio pattern.

**Google Glass:** ~40 min recording per charge
(nowhere near “always on”)
Example: iPad Air (2013)

- Apple A7 Processor
- Dual-core 64-bit ARM CPU
- Imagination PowerVR GPU
- Video Encode/Decode
- Image Processor
- 4MB L3 Cache
- Core
- Core
- 4MB L3
- Touchscreen controllers
- Flash memory
- DRAM
- Motion co-processor (accelerometer, gyro, compass, etc.)

Image Credit: ifixit.com
Original iPhone touchscreen controller

Separate digital signal processor to interpret raw signal from capacitive touch sensor (do not burden main CPU)
Modern computing: efficiency often matters more than in the past, not less

Fourth, there’s battery life.

To achieve long battery life when playing video, mobile devices must decode the video in hardware; decoding it in software uses too much power. Many of the chips used in modern mobile devices contain a decoder called H.264 – an industry standard that is used in every Blu-ray DVD player and has been adopted by Apple, Google (YouTube), Vimeo, Netflix and many other companies.

Although Flash has recently added support for H.264, the video on almost all Flash websites currently requires an older generation decoder that is not implemented in mobile chips and must be run in software. The difference is striking: on an iPhone, for example, H.264 videos play for up to 10 hours, while videos decoded in software play for less than 5 hours before the battery is fully drained.

When websites re-encode their videos using H.264, they can offer them without using Flash at all. They play perfectly in browsers like Apple’s Safari and Google’s Chrome without any plugins whatsoever, and look great on iPhones, iPods and iPads.

Steve Jobs’ “Thoughts on Flash”, 2010
http://www.apple.com/hotnews/thoughts-on-flash/

(Justification for why Apple won’t support Adobe Flash)
Example: image processing on a Nikon D7000

Process 16 MPixel RAW data from sensor to obtain JPG image:
On camera: ~ 1/6 sec per image
Adobe Lightroom a quad-core Macbook Pro laptop: 1-2 sec per image

This is a older camera: much, much faster image processing performance on a modern smart phone (burst mode)
Qualcomm Hexagon Digital Signal Processor

- Originally used for audio/LTE support on Qualcomm SoC’s
- Multi-threaded, VLIW DSP
- Third major programmable unit on Qualcomm SoCs
  - Multi-core CPU
  - Multi-core GPU (Adreno)
  - Hexagon DSP
Up next? application programmable image signal processors

- All modern systems have fixed-function support for common image processing tasks: image/video encode/decode, sensor to image conversion, etc.
- Computational photography: use of advanced algorithms to make better photographs and videos
  - Large space of (rapidly evolving techniques)

High Dynamic Range (HDR) and low light enhancement

Automatic panoramas

Remove camera shake
Anton supercomputer

- Supercomputer highly specialized for molecular dynamics
  - Simulates time evolution of proteins
- ASIC for computing particle-particle interactions (512 of them in machine)
- Throughput-oriented subsystem for efficient fast-fourier transforms
- Custom, low-latency communication network designed for communication patterns of N-body simulations

[Developed by DE Shaw Research]
GPU’s are heterogeneous multi-core processors

Compute resources your CUDA programs used in assignment 2

Graphics-specific, fixed-function compute resources
Example graphics tasks performed in fixed-function HW

**Rasterization:**
Determining what pixels a triangle overlaps

**Texture mapping:**
Warping/filtering images to apply detail to surfaces

**Geometric tessellation:**
Computing fine-scale geometry from coarse geometry
FPGAs (Field Programmable Gate Arrays)

- Middle ground between an ASIC and a processor
- FPGA chip provides array of logic blocks, connected by interconnect
- Programmer-defined logic implemented directly by FPGA

Image credit: Bai et al. 2014
Project Catapult [Putnam et al. ISCA 2014]

- Microsoft Research investigation of use of FPGAs to accelerate datacenter workloads
- Demonstrated offload of part of Bing Search’s document ranking logic

1U server (Dual socket CPU + FPGA connected via PCIe bus)
Summary: choosing the right tool for the job

Energy-optimized CPU

Throughput-oriented processor (GPU)

Programmable DSP

FPGA/Future reconfigurable logic

ASIC

Video encode/decode,
Audio playback,
Camera RAW processing,
nearal nets (future?)

~10X more efficient

Easiest to program

~100X???
(jury still out)

Difficult to program
(making it easier is active area of research)

~100-1000X more efficient

Not programmable +
costs 10-100′s millions
of dollars to design /
verify / create

Credit Pat Hanrahan for this taxonomy
Challenges of heterogeneous designs
Challenges of heterogeneity

- So far in this course:
  - Homogeneous system: every processor can be used for every task
  - To get best speedup vs. sequential execution, “keep all processors busy all the time”

- Heterogeneous system: use preferred processor for each task
  - Challenge for system designer: what is the right mixture of resources to meet performance, cost, and energy goals?
    - Too few throughput-oriented resources (lower peak performance/efficiency for parallel workloads -- should have used resources for more throughput cores)
    - Too few sequential processing resources (get bitten by Amdahl’s Law)
    - How much chip area should be dedicated to a specific function, like video?

  (these resources are taken away from general-purpose processing)

- Implication: increased pressure to understand workloads accurately at chip design time
Pitfalls of heterogeneous designs

Say 10% of the workload is rasterization
Let’s say you under-provision the fixed-function rasterization unit on GPU:
Chose to dedicate 1% of chip area used for rasterizer, really needed 20% more throughput: 1.2% of chip area

Problem: rasterization is bottleneck, so the expensive programmable processors (99% of chip) are idle waiting on rasterization. So the other 99% of the chip runs at 80% efficiency!
Tendency is to be conservative, and over-provision fixed-function components (diminishing their advantage)
Challenges of heterogeneity

- Heterogeneous system: preferred processor for each task
  - Challenge for hardware designer: what is the right mixture of resources?
    - Too few throughput oriented resources (lower peak throughput for parallel workloads)
    - Too few sequential processing resources (limited by sequential part of workload)
    - How much chip area should be dedicated to a specific function, like video? (these resources are taken away from general-purpose processing)
    - Work balance must be anticipated at chip design time
  - System cannot adapt to changes in usage over time, new algorithms, etc.
  - Challenge to software developer: how to map programs onto a heterogeneous collection of resources?
    - Challenge: “Pick the right tool for the job”: design algorithms that decompose well into components that each map well to different processing components of the machine
    - The scheduling problem is more complex on a heterogeneous system
    - Available mixture of resources can dictate choice of algorithm
    - Software portability & maintenance nightmare (we’ll revisit this next class)
Reducing energy consumption idea 1: use specialized processing

Reducing energy consumption idea 2: move less data
Data movement has high energy cost

- Rule of thumb in mobile system design: always seek to reduce amount of data transferred from memory
  - Earlier in class we discussed minimizing communication to reduce stalls (poor performance). Now, we wish to reduce communication to reduce energy consumption

- “Ballpark” numbers [Sources: Bill Dally (NVIDIA), Tom Olson (ARM)]
  - Integer op: ~1 pJ *
  - Floating point op: ~20 pJ *
  - Reading 64 bits from small local SRAM (1mm away on chip): ~26 pJ
  - Reading 64 bits from low power mobile DRAM (LPDDR): ~1200 pJ

- Implications
  - Reading 10 GB/sec from memory: ~1.6 watts
  - Entire power budget for mobile GPU: ~1 watt (remember phone is also running CPU, display, radios, etc.)
  - iPhone 6 battery: ~7 watt-hours (compare: Macbook Pro laptop: 99 watt-hour battery)
  - Exploiting locality matters!!!

* Cost to just perform the logical operation, not counting overhead of instruction decode, load data from registers, etc.
Three trends in energy-optimized computing

- **Compute less!**
  - Computing costs energy: parallel algorithms that do more work than sequential counterparts may not be desirable even if they run faster

- **Specialize compute units:**
  - Heterogeneous processors: CPU-like cores + throughput-optimized cores (GPU-like cores)
  - Fixed-function units: audio processing, “movement sensor processing” video decode/encode, image processing/computer vision?
  - Specialized instructions: expanding set of AVX vector instructions, new instructions for accelerating AES encryption (AES-NI)
  - Programmable soft logic: FPGAs

- **Reduce bandwidth requirements**
  - Exploit locality (restructure algorithms to reuse on-chip data as much as possible)
  - Aggressive use of compression: perform extra computation to compress application data before transferring to memory (likely to see fixed-function HW to reduce overhead of general data compression/decompression)
Summary

- Heterogeneous parallel processing: use a mixture of computing resources that each fit with mixture of needs of target applications
  - Latency-optimized sequential cores, throughput-optimized parallel cores, domain-specialized fixed-function processors
  - Examples exist throughout modern computing: mobile processors, servers, supercomputers

- Traditional rule of thumb in “good system design” is to design simple, general-purpose components
  - This is not the case with emerging processing systems (optimized for perf/watt)
  - Today: want collection of components that meet perf requirement AND minimize energy use

- Challenge of using these resources effectively is pushed up to the programmer
  - Current CS research challenge: how to write efficient, portable programs for emerging heterogeneous architectures?