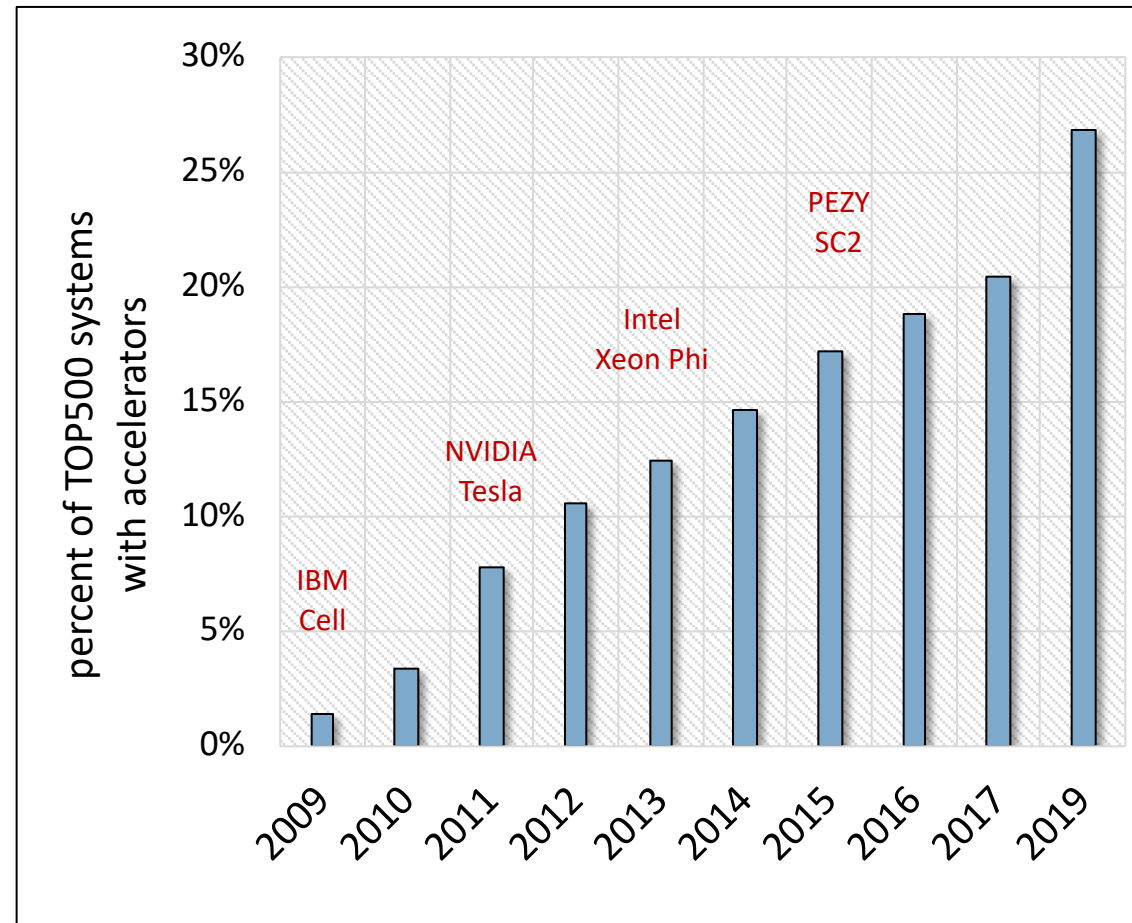


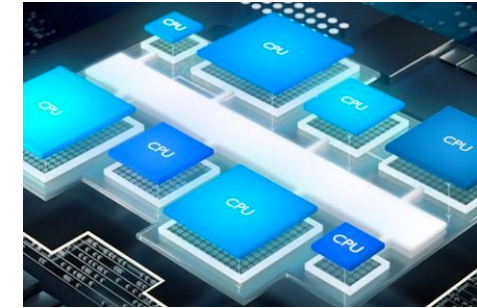
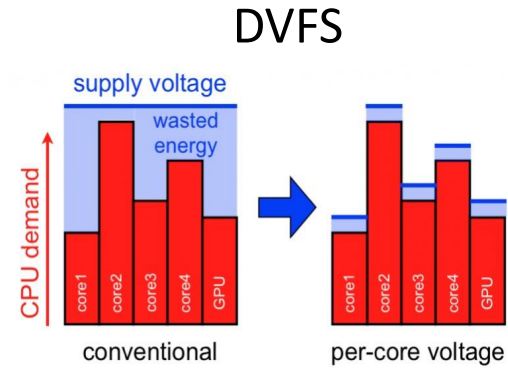
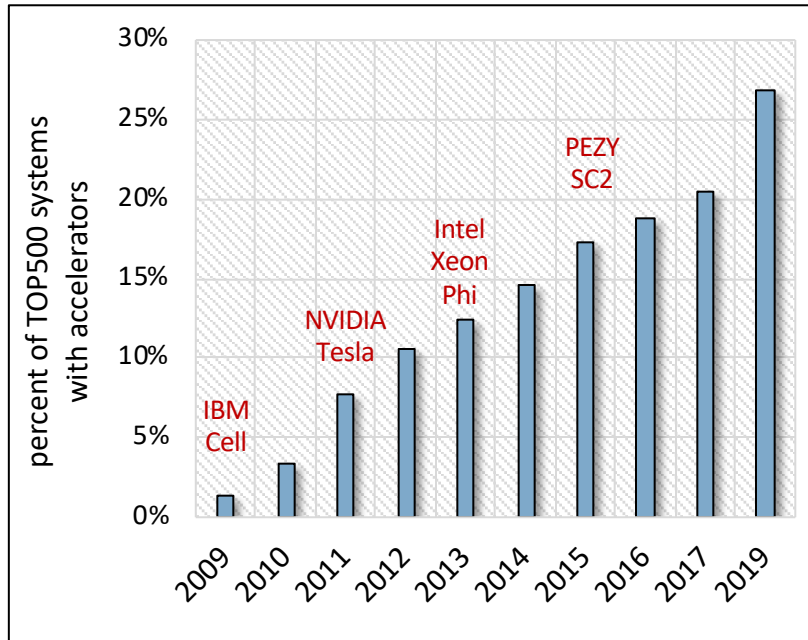
A Gentle Introduction to Heterogeneous Computing for CS1 Students

APAN QASEM
TEXAS STATE UNIVERSITY
SAN MARCOS, TX

The Rise of Heterogenous Computing

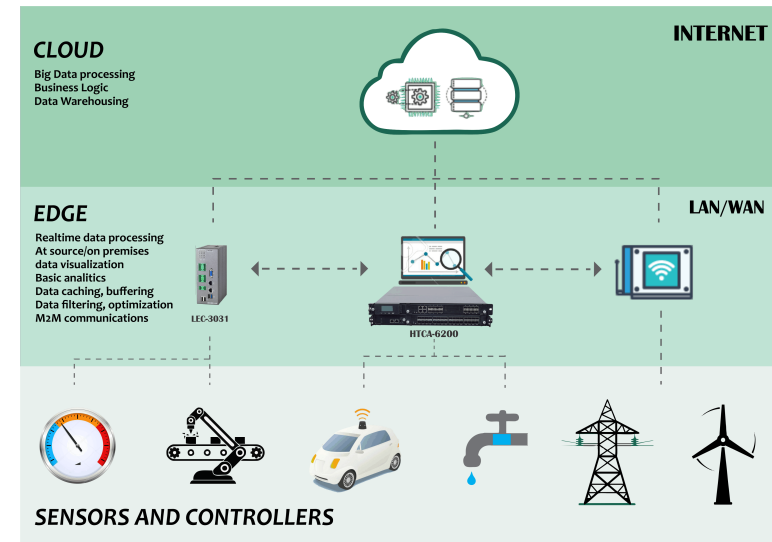


The Rise of Heterogenous Computing



Fat and Thin Cores

Heterogeneity in the Cloud



HC in the Undergraduate Curriculum

The One Vision That Intel, AMD And Nvidia Are All Chasing – Why Heterogeneous Computing Is The Future

The Floodgates are Opening: The Promise of Heterogeneous Computing

**Heterogeneous Computing:
Here to Stay**
Hardware and Software Perspectives



Given current industry trends, it is critically important that CS/CE undergraduates are exposed to heterogeneous computing concepts as a required part of the curriculum

- The ToUCH project was initiated with this goal in mind
- The CS1 module was developed as part of this initiative

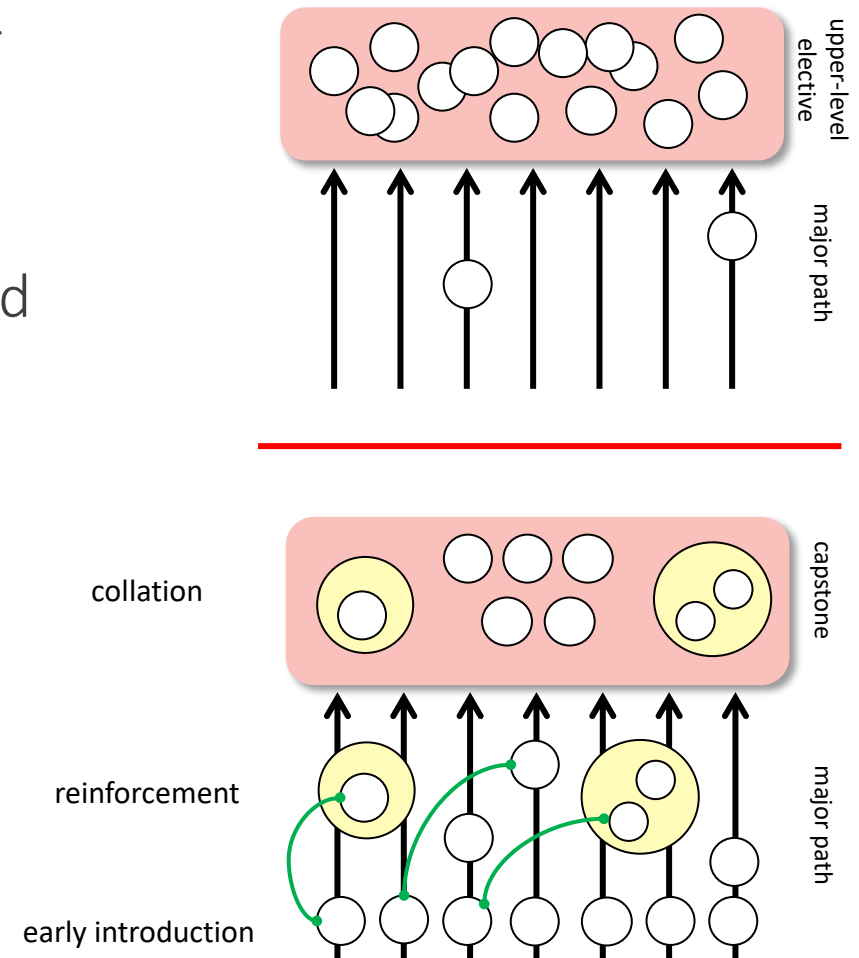


Outline

- Motivation
- Design Principles
- Module Description
 - Lecture slides
 - In-class Demo
 - Lab
- Assessment
- Conclusions and Future Work

The Early-and-Often Approach

- Breaks away from the traditional approach of teaching a topic X as an upper-level elective
- Introduce X early in the curriculum in required courses
- Repeat key concepts in different courses throughout the curriculum
- Tie concepts together in an upper-level capstone course



Design Principles

- modules should be classified and introduced based on level of *abstraction*
 - Cover Amdahl's Law in CS1; memory coherence in a senior-level course
- modules should provide heterogeneous *context* to existing content
 - Introduce soft heterogeneity when discussing threads scheduling in OS
- modules should be *self-contained* for easy adoption across different institutions

An HC Module for CS1

- Content
 - Covers fundamental concepts at a high-level of abstraction
 - No programming, but students are exposed to programming languages and tools
- Context
 - CS1/CS2 students
- Length
 - One hour and twenty minutes lecture + lab
- Pedagogy
 - Emphasis on active learning: in-class demo + lab

Topics and Learning Outcomes

	Topic	Bloom Classification
1	Concurrency and Parallelism	Knowledge
2	Multicore Processors	Knowledge
3	GPGPU	Knowledge
4	System-on-Chip	Knowledge
5	Energy Efficiency	Knowledge
6	Tasks and Workloads	Knowledge
7	Task Mapping and Scheduling	Knowledge
8	Amdahl's Law	Knowledge/Comprehension





- Describe and distinguish between different forms of heterogeneity
- Recognize the importance of energy efficiency on current computing systems
- ...



What's included

Processor on a Board

Inside a Processor

Heterogeneous Computing : Elementary Notions

LECTURE
CS1
FALL 2018

Developed by : name, email
Work supported by NSF grant OAC-1829644

Multi-tasking Heterogeneous Computers

Description

Heterogeneous computers are particularly good at performing a variety of tasks concurrently. But in order for them to do a good job, the software must ensure that tasks are appropriately distributed to the specialized cores.

In this assignment, you will investigate performance (and energy) issues of a heterogeneous computing system. You will be given a set of four programs with different characteristics. Your goal is to determine the best mapping of these programs to the different processing cores via experimentation and analysis.

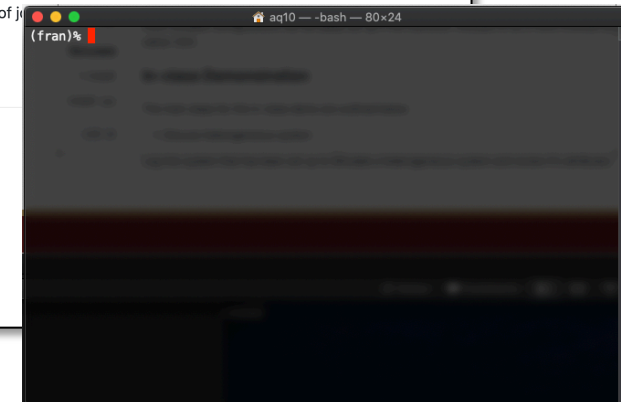
Environment

You will be running experiments on `megatron`, a heterogeneous multicore system. `megatron` has four processing cores and each core has been configured to do a specific type of job. You will perform certain tasks really well.

Tools

Familiarize yourself with the following tools.

- `perf`
- `taskset`
- `likwid`
- `cpufrequtils`
- `cpupower`



Soft Heterogeneity

Objective

Demonstrate the performance and energy impact of soft heterogeneity. Demo

Software and Tools

The following Linux tools are used in this demo.

- `cpufrequtils`
- `cpupower`
- `perf`
- `energy`
- `taskset`

The demo also includes a simple C++/OpenMP code that performance matrix-vec

Environment Set-up

These steps should be carried prior to class time.

Simulate a multicore heterogeneous platform.

1. Install necessary packages and their dependencies

Install `cpufrequtils`

```
sudo apt install cpufrequtils
```

```
(knuth)% pin matvec [0,1]
Program matvec pinned to cores [0,1]
(knuth)% profile ./matvec 10000 2
```

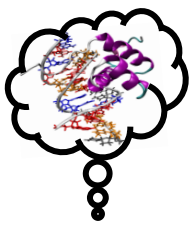
Metric	Sum	Core 0	Core 1	Core 2	Core 3
Runtime [ms]	5721.50	2860.75	2860.75	0.50	0.80
Power [W]	7.01	--	--	--	--
Energy [J]	40.14	--	--	--	--

```
(knuth)% pin matvec [2,3]
Program matvec pinned to cores [2,3]
(knuth)% profile ./matvec 10000 2
```

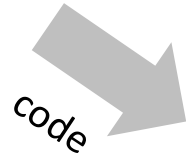
Metric	Sum	Core 0	Core 1	Core 2	Core 3
Runtime [ms]	2875.80	0.20	0.50	1437.90	1437.90
Power [W]	18.05	--	--	--	--
Energy [J]	51.90	--	--	--	--

```
(knuth)%
```

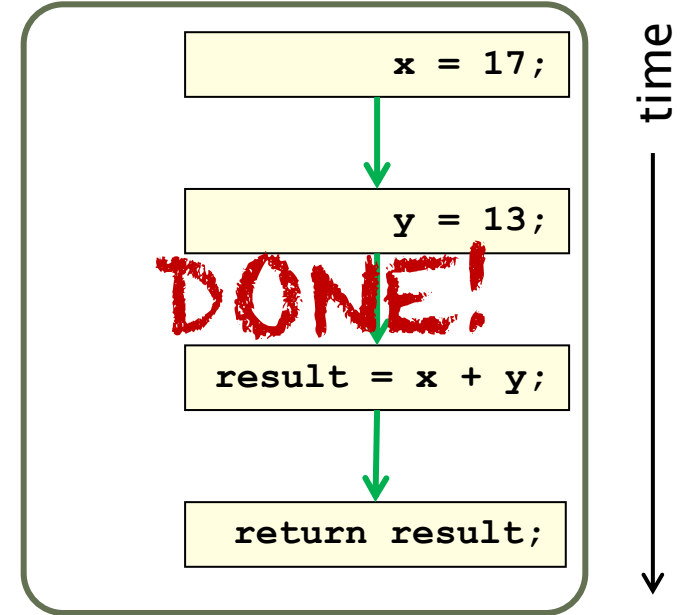
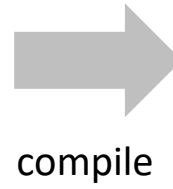
<https://github.com/TeachingUndergradsCHC>



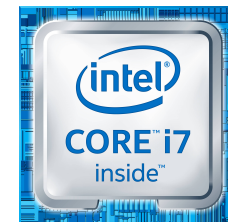
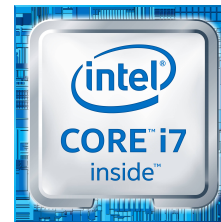
Parallel Program Execution

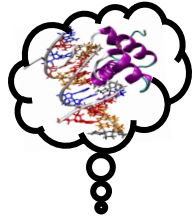


```
int main() {  
    int x, y, result;  
    x = 17;  
    y = 13;  
    result = x + y;  
    return result;  
}
```



*The two assignment statements
x = 17; and y = 13; will execute in parallel*





Heterogeneous

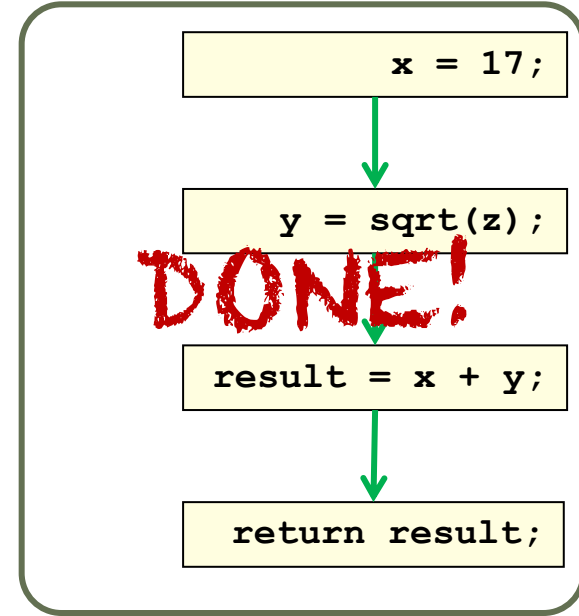
Program Execution



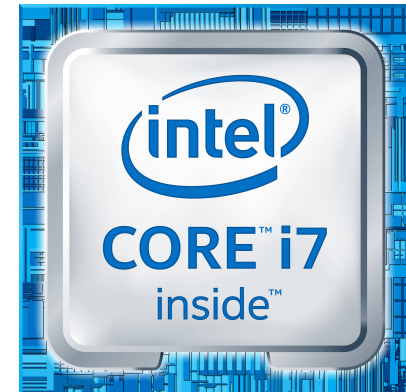
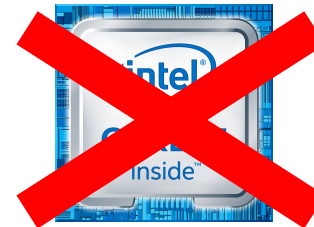
code

```
int dostuff(int z) {
  int x, y, result;
  x = 17;
  y = sqrt(z);
  result = x + y;
  return result;
}
```

compile



inefficient mapping



“Big” processor is twice as powerful

Assign heavier computation to the “bigger” processor

Demo: Soft Heterogeneity

- Objective
 - demonstrate performance and energy tradeoff on heterogeneous multicore systems
 - serve as a preview of the lab to follow
- Preparation
 - Set up a multicore system to simulate Big-Little behavior with DVFS tools
 - Install performance and power profiling tools on system
- Main steps
 - Familiarize students with a heterogeneous environment
 - Build and execute *simple* code on Little and Big cores
 - Discuss performance and energy differences

In-class Demonstration

The main steps for the in-class demo are outlined below

Discuss heterogeneous system.

Log into system that has been set up to simulate a heterogeneous system and review it's attributes.

```
cat /proc/cpuinfo
cpuf
```

```
(knuth)% pin matvec [0,1]
Program matvec pinned to cores [0,1]
(knuth)% profile ./matvec 10000 2
```

Metric	Sum	Core 0	Core 1	Core 2	Core 3
Runtime [ms]	5721.50	2860.75	2860.75	0.50	0.80
Power [W]	7.01	--	--	--	--
Energy [J]	40.14	--	--	--	--

```
(knuth)% pin matvec [2,3]
Program matvec pinned to cores [2,3]
(knuth)% profile ./matvec 10000 2
```

Metric	Sum	Core 0	Core 1	Core 2	Core 3
Runtime [ms]	2875.80	0.20	0.50	1437.90	1437.90
Power [W]	18.05	--	--	--	--
Energy [J]	51.90	--	--	--	--

```
(knuth)%
```

Lab: Multi-tasking Heterogeneous Computers

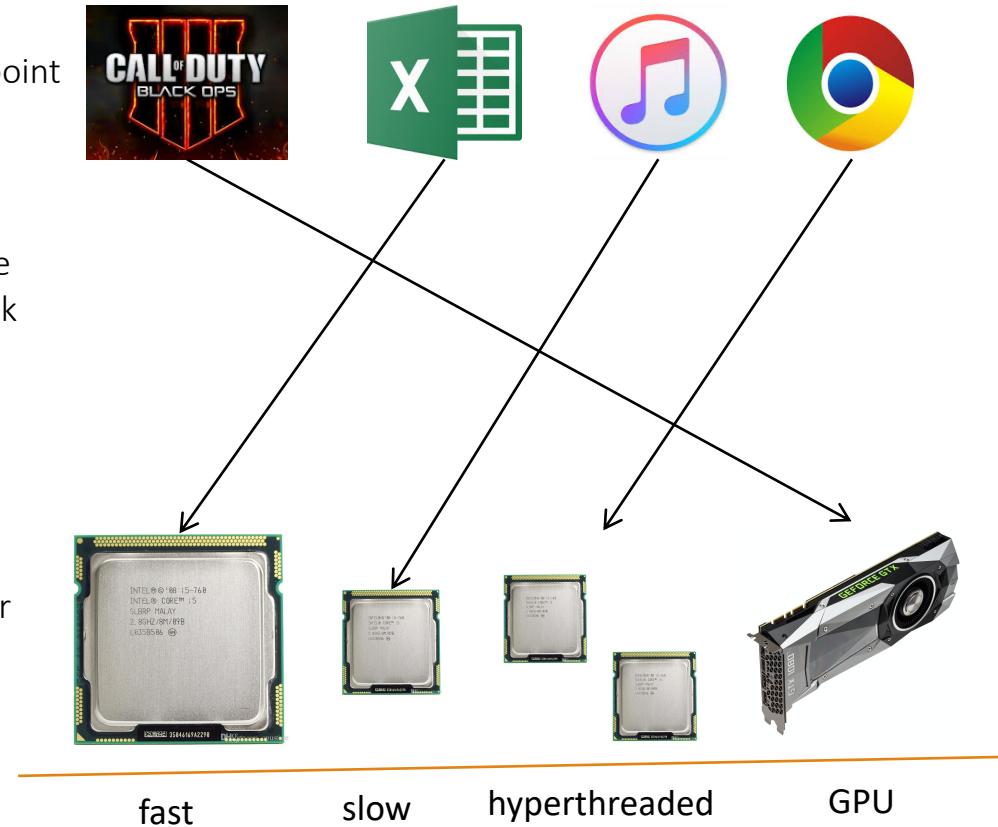
Synthetic workload captures application characteristics such as graphics, floating-point math, multimedia streaming and multithreading

Students conduct experiments and analyze the performance and energy impact of task mapping

Tools used: `pin` and `profile`
(under the hood: `perf`, `likwid`, `taskset`)

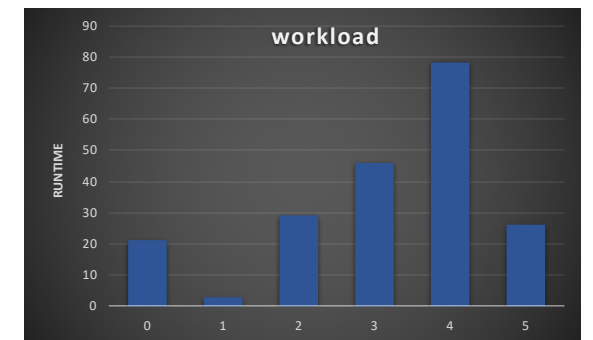
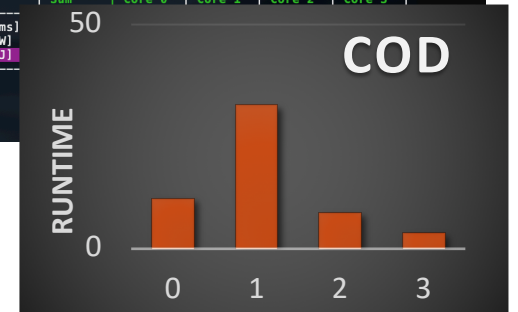
Simulated multicore heterogeneous server with consist of fast and slow cores; hyperthreaded cores and a GPU

Tools needed `cpufrequtils` and `pin`



```
(knuth)% pin matvec [0,1]
Program matvec pinned to cores [0,1]
(knuth)% profile ./matvec 10000 2
| Metric | Sum | Core 0 | Core 1 | Core 2 | Core 3 |
|-----|----|-----|-----|-----|-----|
| Runtime [ms] | 5721.50 | 2860.75 | 2860.75 | 0.50 | 0.80 |
| Power [W] | 7.01 | -- | -- | -- | -- |
| Energy [J] | 40.14 | -- | -- | -- | -- |

(knuth)% pin matvec [2,3]
Program matvec pinned to cores [2,3]
(knuth)% profile ./matvec 10000 2
| Metric | Sum | Core 0 | Core 1 | Core 2 | Core 3 |
|-----|----|-----|-----|-----|-----|
| Runtime [ms] | 50 | -- | -- | -- | -- |
| Power [W] | -- | -- | -- | -- | -- |
| Energy [J] | -- | -- | -- | -- | -- |
```

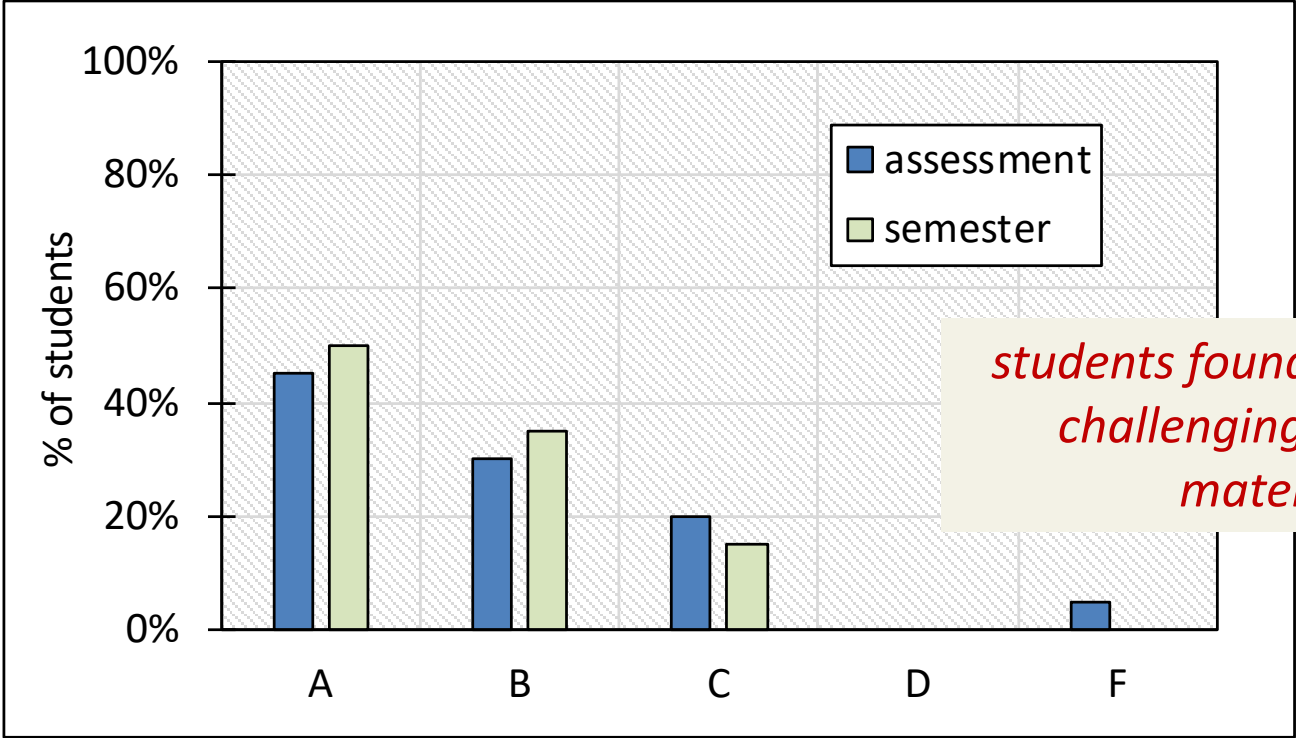


Assessment

- Module was implemented in a CS1 course at Texas State in Fall 2018
 - class teaches sequential programming in C++
- The class is designated as an Honors section
 - over achievers
 - lower cap
- Learning outcome evaluated via exam questions and the lab
- Student engagement and interest evaluated via SALG survey

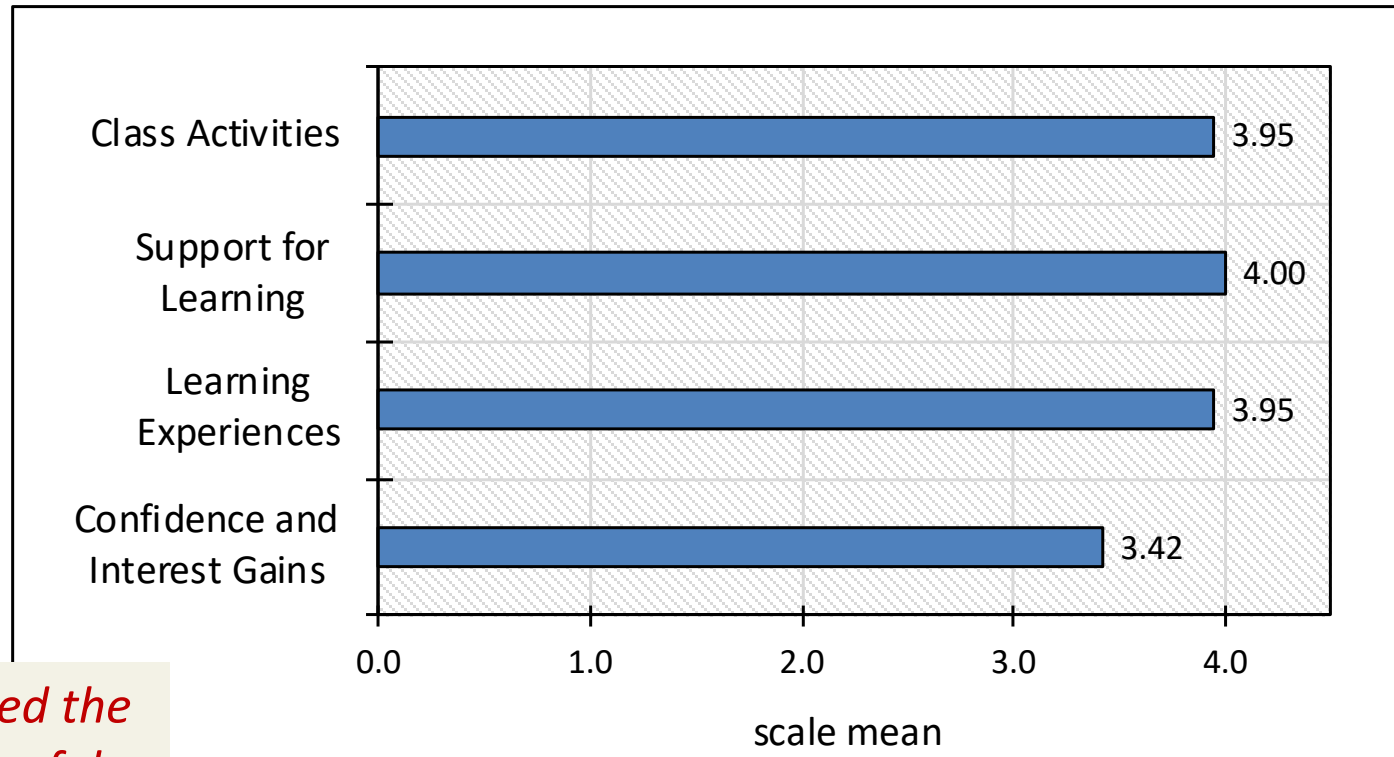


Learning Outcome



students found HC topics a little more challenging than the rest of the material in the class

Student Engagement and Interest Gains



students perceived the module to be useful

Conclusions and Future Work

- Preliminary evaluation is promising both in terms of student learning outcome and engagement
 - specifically the hands-on assignment and use of experimental tools
- Material might be too much to cover in a regular CS1 class
- Calibrate content for future installments
- Implement a more structured evaluation plan

Thank you!

Acknowledgements



David Bunde, Knox College
Phil Schielke, Concordia Texas



OAC-1829644