



中山大學  
SUN YAT-SEN UNIVERSITY



国家超级计算广州中心  
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

# Computer Architecture 计算机体系结构

---

## 第0讲：课程介绍、概述

张献伟

[xianweiz.github.io](https://xianweiz.github.io)

DCS329, 9/1/2021



中山大學  
SUN YAT-SEN UNIVERSITY



# 关于课程

- 年级专业
  - 19级信息与计算科学（专选，**16**/39人）
- 先修课程
  - 计算机组成原理
  - 程序设计、数据结构、操作系统等
- 计算机体系结构
  - 如何设计一台符合系统设计目标的计算设备？
  - 使用量化分析方法，对计算机瓶颈问题定位、描述、分析、评估
    - 包括CPU流水线、指令集并行、存储层次结构，多核结构等

主要有数据科学方向和计算科学方向。培养具有扎实的数学与统计学基础，掌握计算机应用的主要知识与技能，熟悉现代数据分析的主要理论、方法与技巧的复合型人才。数据科学方向面向现代商业、金融、企业数据分析；计算科学方向面向科学与工程计算问题。

# 课程教材

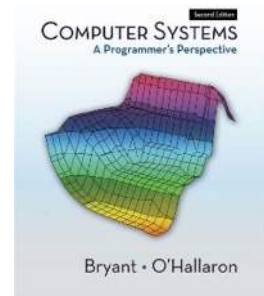
- 主要教材

- John L. Hennessy and David A. Patterson, 计算机体系结构：量化研究方法（英文版·原书第6版）
- 非必须购买



- 参考资料

- ECE 447, Onur Mutlu (Carnegie Mellon U.)
- CIS 501, Milo Martin (U Penn)
- Computer Organization and Design RISC-V Edition: The Hardware Software Interface (2nd Edition), Hennessy and Patterson
- Computer Systems: A Programmer's Perspective (CSAPP), Bryant and O'Hallaron



# 任课教师



博士，2011 – 2017，University of Pittsburgh



学士，2007 – 2011，西北工业大学

中山大學

副教授，2020.10 – 今

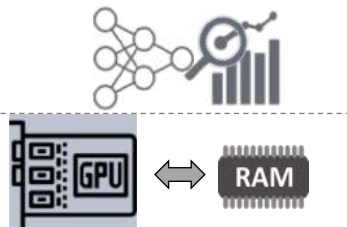


工程师/研究员，2017.08 – 2020.09



实习研究员，2016.05 – 2016.08

计算机体系结构  
高性能及智能计算  
软硬件设计及优化



学院个人主页: <http://cse.sysu.edu.cn/content/5592>

# 课件及答疑

---

- 课件

- [xianweiz.github.io/teach/dcs329/f2021.html](http://xianweiz.github.io/teach/dcs329/f2021.html)
  - 课后上传更新

- 课程QQ群: **TBD**

- 教师

- 张献伟（超算中心）
    - Email: [zhangxw79@mail.sysu.edu.cn](mailto:zhangxw79@mail.sysu.edu.cn)
    - 课前课间答疑，其他时间需预约

- 助教

- 翁跃（超算中心）
    - Email: [wuyue0828@163.com](mailto:wuyue0828@163.com)
    - 研究方向：高性能、系统结构

# 课时及考核

- 课时（2学分，36学时）

- 排课：1-9，11-19周
- 每次授课包括2个课时
  - 周三，7-8节（16:20-18:00）
- 地点：教学大楼 C103

- 考核

- 课堂参与（15%）- 点名、提问、测试
- 实验（15%）- 课下
- 作业（20%）- 理论
- 期末考试（50%）- 闭卷


- 课堂

- 随机点名
  - 缺席优先
- 随机提问
  - 后排优先
- 随机测试
  - 不定时间

- 实验/作业

- 个人完成
  - 杜绝抄袭
- 按时提交
  - 超算习堂

# 上课安排

| Week/Date    | Topic  | Note              |
|--------------|--|-------------------|
| wk1: Sep 1   |  <a href="#">Overview</a> |                   |
| wk2: Sep 8   |  |                   |
| wk3: Sep 15  |  |                   |
| wk4: Sep 22  |  |                   |
| wk5: Sep 29  |  |                   |
| wk6: Oct 6   | NO CLASS   | Holiday           |
| wk7: Oct 13  |  |                   |
| wk8: Oct 20  |  |                   |
| wk9: Oct 27  |  |                   |
| wk10: Nov 3  | NO CLASS   | Midterm Exercises |
| wk11: Nov 10 |  |                   |
| wk12: Nov 17 |  |                   |
| wk13: Nov 24 |  |                   |
| wk14: Dec 1  |  |                   |
| wk15: Dec 8  |  |                   |
| wk16: Dec 15 |  |                   |
| wk17: Dec 22 |  |                   |
| wk18: Dec 29 |  |                   |
| wk19: Jan 5  |  |                   |
| wk20: Jan 12 | NO CLASS   | FINAL EXAM        |

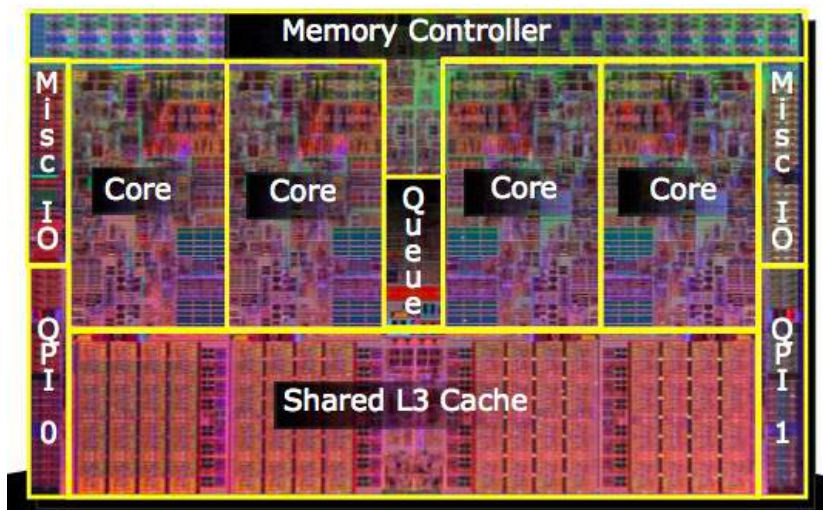
# 内容安排

---

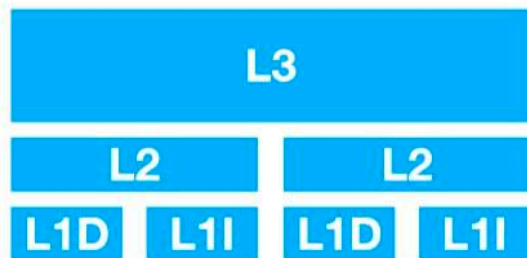
- Overview and Fundamentals[概览与基础]
- Instruction Set Architecture[指令集架构]
  - Review
- Instruction Level Parallelism[指令级并行]
  - Pipelining, Branch Prediction, Instruction Scheduling
- Memory Hierarchy[存储层级]
  - Memory, Cache, Virtual Memory
- Data/Thread-Level Parallelism[数据/线程级并行]
  - SIMD, GPU
- Domain-specific Architectures[领域专用架构]
- And more ...



# Examples

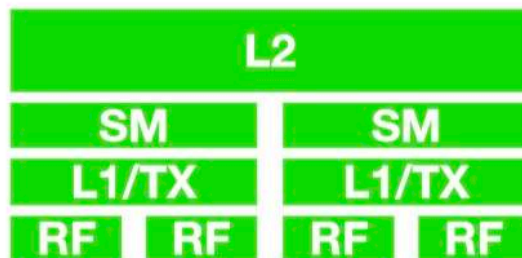


**CPU**



*implicitly managed*

**GPU**



*mixed*

**'TPU'**



*explicitly managed*

# 课程目标

---

- This course covers HW/SW, and the interface[内容覆盖]
  - We will focus on performance analysis and design tradeoffs
- Two key goals of this course are[主要目标]
  - To understand how **hardware** components works underneath the **software** layer and how decisions made in hardware affect the software/programmer
  - To enable you to be comfortable in making **design and optimization** decisions that cross the boundaries of different layers and system components
- Two other goals of this course[额外目标]
  - Enable you to think **critically**
  - Enable you to think **broadly**

# 一些问题？ ？ ？

“摩尔定律”真的要终结吗？量子计算是未来吗？

GPU怎么就从图形专用到了计算通用？

NPU, TPU, 加速器？异构计算？

为什么Nvidia过去几年市值增长了40倍？没有竞争对手的吗？

Intel挤牙膏？AMD YES！

DRAM、DDR、HBM、LPDDR，什么关系？

Meltdown, Spectre漏洞为什么会发生？

华为被制裁，卡脖子卡的是什么？x86, ARM, RISC-V？

太阳位置会影响计算机稳定？

为什么要有cuBLAS基础库？

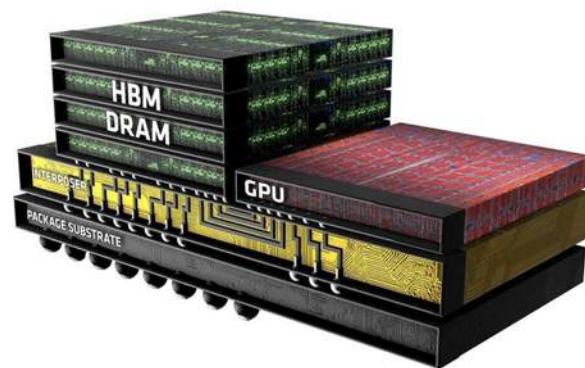
GPU任务怎么被执行的？profiling怎么实现的？

数据中心、超算、云，什么关系？

“冯·诺伊曼”架构要被淘汰了？

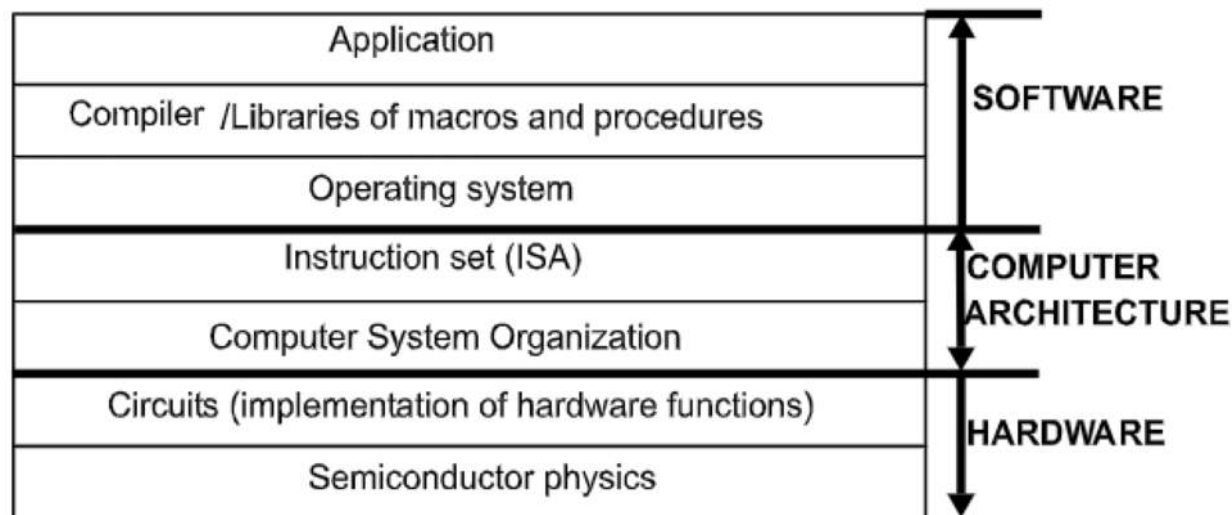
# 什么是体系结构？

- Computer Architecture
  - The science and art of designing, selecting, and interconnecting **hardware components** and designing the **hardware/software interface** to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals[硬件设计 – 软硬件接口 – 系统目标]
- Computer architecture is the glue that binds SW and HW
  - Inter-disciplinary in nature[连接软硬件、跨学科]



# 什么是体系结构？ (cont.)

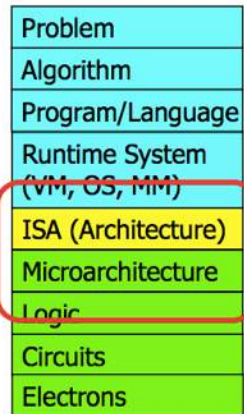
- Hardware organization of computers[硬件组成]
  - How to build computers?
- Layered view of computer systems[分层角度]



- Role of computer architect[架构师]
  - To make design **trade-offs** across the hw/sw interface to meet functional, performance and cost **requirements**

# 计算机分层与抽象

- Layering to deal with complex systems[分层]
  - Computers are complex, built in **layers**
    - Software: applications, compiler, operating system
    - Hardware: logic, circuits, electrons
- Layering brings abstraction[抽象]
  - A higher level only needs to know about the interface to the lower level, not how the lower level is implemented
    - 99% of users don't know hardware layers implementation
    - 90% of users don't know implementation of any layer
    - That's okay, world still works just fine
- Then, **why** would you want to know what goes on underneath or above?



# 计算机分层与抽象 (cont.)

---

- As long as everything **goes well**, not knowing what happens in the underlying level (or above) is **not a problem**
- What if
  - The program you wrote is running slow?[慢]
  - The program you wrote does not run correctly?[错误]
  - The program you wrote consumes too much energy?[高能耗]
- What if
  - The hardware you designed is too hard to program?[难用]
  - The hardware you designed is too slow because it does not provide the right primitives to the software?[性能差]
- What if
  - You want to design a much more efficient and higher performance system?[效率更高、性能更高]

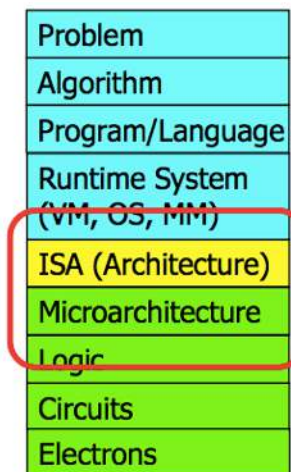


# 计算机分层与抽象 (cont.)

- How would you solve the problem?[怎样解决]
- What is the right place to solve the problem?

[哪里解决]

- Programmer?
- System software? Compiler?
- Hardware? Circuits?



- Breaking the abstraction layers and knowing what is underneath enables you to solve problems and design better future systems[打破抽象]
- Cooperation between multiple components and layers enable more effective solutions and systems[协同考虑]



# 架构和架构师

---

- Computer architect[计算机架构师]
  - To make **design trade-offs** across the hw/sw interface to meet functional, performance and cost requirements
- Being an architect is not easy[并不容易]
  - You need to consider many things in designing a **new** system + have good **intuition/insight** into ideas/tradeoffs
- But, it is fun and can be very technically rewarding[会很有意思]
- And, enables a great future[影响未来]
  - Advancement of computer architecture is vital to all other areas of computing
    - E.g., IoT, Embedded, Mobile, Data centers, HPC

# Role of [Computer] Architect[职责]

---

- Look **backward** (to the past)
  - Understand tradeoffs and designs, upsides/downsides, past workloads. Analyze and evaluate the past.
- Look **forward** (to the future)
  - Be the dreamer and create new designs. Listen to dreamers.
  - Push the state of the art. Evaluate new design choices.
- Look **up** (towards problems in the computing stack)
  - Understand important problems and their nature.
  - Develop architectures and ideas to solve important problems.
- Look **down** (towards device/circuit technology)
  - Understand the capabilities of the underlying technology.
  - Predict and adapt to the future of technology (you are designing for  $N$  years ahead). Enable the future technology.

# 为什么要学习体系结构？

---

- Understand why computers work the way they do
  - Source code --> instructions --> executions
  - Processors --> cache/memory --> storage
- Understand where computers are going
  - Future capabilities drive the (computing) world
  - Real world-impact: no computer architecture -> no computers!
- Understand high-level design concepts and performance
  - The best architects understand all the levels (hw, OS, apps, alg)
  - Need to understand hardware to write fast software
- Job?
  - Your MS/PhD research may involve CA
  - Your job positions (design or research, hw or sw) may need CA
  - You may manage a team working on systems

# 工作职位？ ？ ？

## CAREERS AT NVIDIA

### Deep Learning Architect – New College Grad

US, CA, Santa Clara

Apply

NVIDIA is seeking computer architects to help design processor and system architectures that will enable compelling Deep Learning performance, architecture and efficiency improvements. This role offers the opportunity to directly impact the future hardware roadmap in a fast-growing technology company that leads the AI revolution. If you are obsessed with improving deep learning performance beyond anything possible with today's hardware and software, this is the place to be.

#### What you'll be doing:

- Understand, analyze, profile, and optimize deep learning training workloads on state-of-the-art hardware and software platforms.
- Guide development of future generations of deep learning processors and computing platforms.
- Develop detailed performance models and simulators for computing systems accelerating DL training.
- Collaborate across the company to guide the direction of machine learning at NVIDIA; spanning teams from hardware to software and research to production.
- Drive HW/SW co-design of NVIDIA's full deep learning platform stack, from silicon to DL frameworks.

#### What we'd like to see:

- You are pursuing a PhD or MS or have equivalent in CS, EE or CSEE (or equivalent experience).
- Strong background in computer architecture, preferably with focus on high-performance parallel processors.
- Background in machine learning and neural networks, in particular training.
- Experience analyzing and tuning application performance.
- Experience with processor and system-level performance modelling.
- Programming skills in C++ and Python.
- Familiarity with GPU computing (CUDA, OpenCL).

### 达摩院-芯片性能架构师-北京

|       |            |       |    |       |      |
|-------|------------|-------|----|-------|------|
| 发布时间: | 2021-04-20 | 工作地点: | 北京 | 工作年限: | 五年以上 |
| 所属部门: | 阿里集团       | 学 历:  | 硕士 | 招聘人数: | 若干   |

#### 团队介绍:

阿里巴巴集团达摩院旗下的数据计算-计算技术实验室致力于前瞻性研究,探索异构计算,存储,和互联的系统架构,软硬件协同设计,体系结构,电路设计,编译和编程环境等方面的技术问题,研制高性能、低功耗的异构计算系统,人工智能计算芯片,以及其他芯片架构及系统。通过自上而下基于应用驱动和自下而上基于新技术的研究方法,利用计算机体系结构设计优化, VLSI等领域的技术积累与合作伙伴在计算资源优化、新计算体系方向等构建创新系统,提升阿里巴巴集团计算能力并复用于国民经济的各行各业中。

#### 岗位描述:

参与新型计算芯片的开发,作用包括:

- 分析业务软件的性能特征,在系统和仿真环境中分析性能如何,以帮助芯片架构的设计决策。
- 开发和验证性能分析工具,实现数据分析。
- 参与SOC模块的性能建模,和设计团队一起优化芯片的性能,并和验证。
- 开发性能测试用例,测试系统的微架构特征和设计参数。
- 在新兴计算中,领域(机器学习,视频计算等),分析业务特点,推动软硬件协同设计。

#### 岗位要求:

- 热爱芯片架构,愿意学习。
  - 软件编程能力(如C++, python)
  - 具有计算机体系结构背景,熟悉SOC组件。
- 职位加分:
- 有相关的计算机体系结构研究经验。
  - 有性能模型建模的经验。

# Golden Age for CA

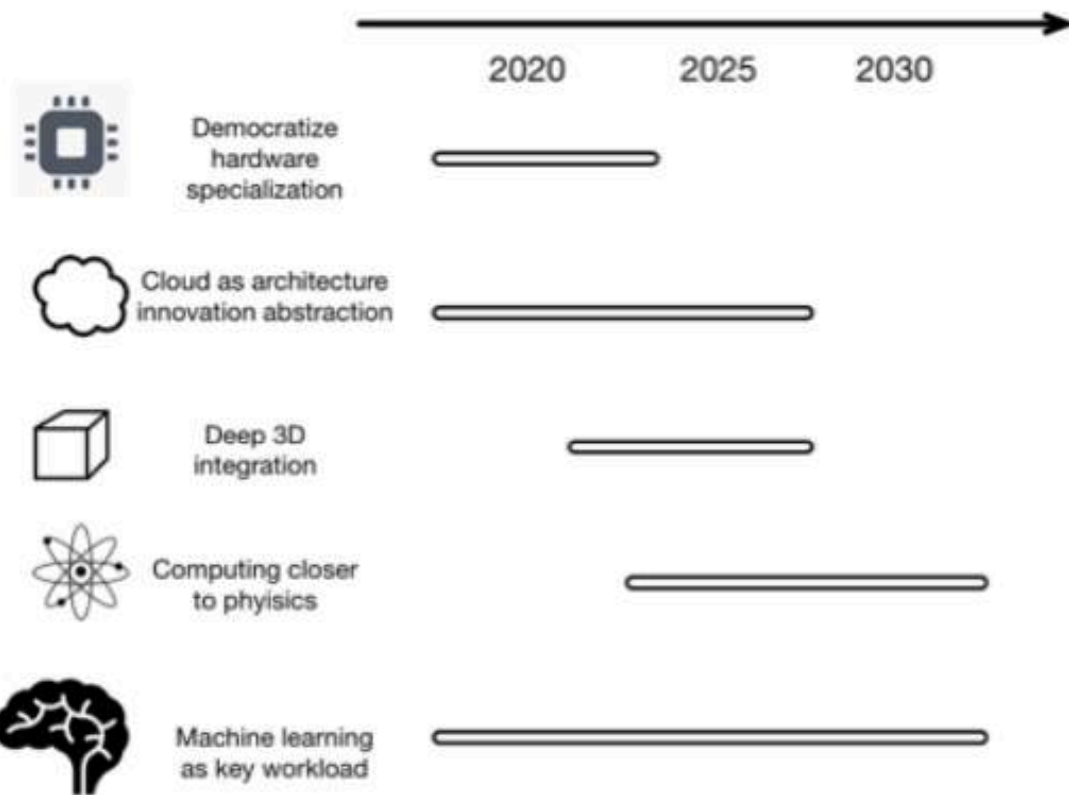
---

- **Today** is a very exciting time to study architecture
  - Many new demands from the top
  - Fast changing demands and personalities of users
  - Many new issues at the bottom
- Computing landscape is **very different** from 10-20 years ago
  - Every component and its interfaces, as well as entire system designs are being re-examined
  - You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)
- **No clear, definitive answers to these problems[有问题, 缺方案]**

# 学术界

## Architecture 2030 Workshop @ ISCA 2016

John L. Hennessy, David A. Patterson



- Current challenges[问题]
  - End of Moore's Law and Dennard Scaling
  - Overlooked security
- Future opportunities in computer architecture[机遇]
  - Domain-specific architectures
  - Domain-specific languages
  - Open architectures
  - Agile hardware development

[1] Arch2030, <https://arxiv.org/pdf/1612.03182.pdf> (2016)

[2] [A New Golden Age for Computer Architecture](#) (2019)

# 工业界

---

- Nvidia, 04/2021: Grace, ARM-based data-center CPU<sup>[1]</sup>
- Apple, 11/2020: M1, ARM-based SoC<sup>[2]</sup>
- AMD, 10/2020: Acquire Xilinx<sup>[3]</sup>
- Intel, 09/2020: Xe GPU<sup>[4]</sup>
- Samsung, 11/2019: Cease CPU development<sup>[6]</sup>
- Amazon, 11/2018: AWS Graviton<sup>[5]</sup>
- Intel/IBM/ARM, 01/2018: Meltdown and Spectre
- Micron, 03/2021: Cease 3D-XPoint, invest CXL<sup>[7]</sup>
- AI Chips: Graphcore, Habana Labs, Cerebras, Cambricon...

[1] <https://www.nvidia.com/en-us/data-center/grace-cpu/>

[2] <https://pdf.wondershare.com/macros/everything-about-apple-m1-chip.html>

[3] <https://www.amd.com/en/corporate/xilinx-acquisition>

[4] <https://www.intel.com/content/www/us/en/products/discrete-gpus/iris-xe-aic.html>

[5] <https://aws.amazon.com/ec2/graviton/>

[6] <https://www.anandtech.com/show/15061/samsung-to-cease-custom-cpu-development>

[7] <https://investors.micron.com/news-releases/news-release-details/micron-updates-data-center-portfolio-strategy-address-growing>



# SC: Aurora

- Aurora @ Argonne National Laboratory\*
  - 1 exaFLOPS,  $\leq 60$  MW, 2018-2021-2022
  - Compute node:
    - 2 Intel Xeon Sapphire Rapids CPUs, 6 Xe GPUs
      - First enterprise CPUs to support CXL standard
      - GPUs communicates directly to each other via CXL
    - Unified memory architecture
  - Interconnect
    - CPU-GPU: PCIe, GPU-GPU: Xe Link
    - System: HPE Slingshot 11; Dragonfly topology with adaptive routing
  - Programming models
    - Intel oneAPI, MPI, OpenMP, C/C++, Fortran, SYCL/DPC++
  - Applications: climate change, cancer, new materials

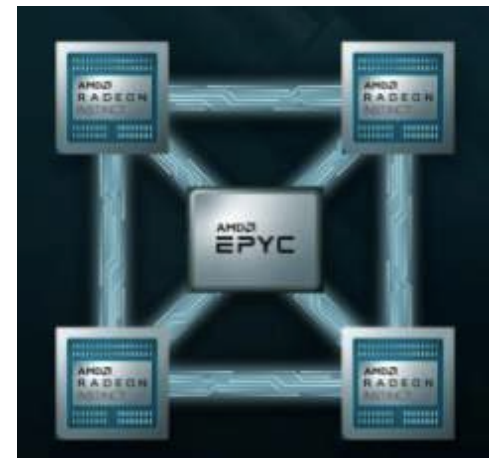


\* <https://www.alcf.anl.gov/aurora>



# SC: Frontier

- Frontier @ Oak Ridge National Laboratory\*
  - 1.5 exaFLOPS, 30 MW, 2021
  - Compute node
    - 1 AMD EPYC CPU (Zen 3) + 4 AMD Radeon Instinct GPU (MI 200)
  - Interconnect
    - Node: AMD Infinity Fabric, coherent memory across the node
    - System:
      - Multiple Slingshot NICs providing 100 GB/s network bandwidth.
      - Slingshot dragonfly network w/adaptive routing
  - Programming models:
    - AMD ROCm, MPI, OpenMP, HIP C/C++, Fortran
  - Applications: modeling and simulation, data analytics, AI



\* [https://www.olcf.ornl.gov/wp-content/uploads/2019/05/frontier\\_specsheet.pdf](https://www.olcf.ornl.gov/wp-content/uploads/2019/05/frontier_specsheet.pdf)

# SC: Tianhe-3

- Tianhe-3 @ Tianjin
  - China's native manycore Armv8-based Phytium 2000+ (FTP)
  - Matrix 3000 (MTP) Accelerator

