上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

大规模工程计算

第二课

集群架构与编译器

王 一 超

2023年9月27日

课程大纲持续更新中

计算机背景

时间	授课主题	时间	授课主题
第2周 9月20日	课程介绍 高性能计算发展史	第10周 11月15日	OpenFOAM软件实战
第3周 9月27日	集群架构与编译器	第11周 11月22日	Abaqus软件实战
第4周	中秋/国庆休假	第12周 11月29日	Fluent软件实战（待定）
第5周 10月11日	文件系统与数据管理	第13周 12月6日	GPU加速应用
第6周 10月18日	MATLAB并行计算	第14周 12月13日	人工智能实战
第7周 10月25日	计算材料学	第15周 12月20日	代码实战（上）一步步优化性能
第8周 11月1日	LAMMPS软件实战	第16周 12月27日	代码实战（下）真实应用优化
第9周 11月8日	计算流体力学	第17周 1月3日	现场答疑

报告提纲

以“思源一号”为例

计算资源的基础概念

编译器基础

“交我算”课堂实践

杨元庆校董出资一亿元捐建“思源一号”

2021.4.8 捐赠仪式

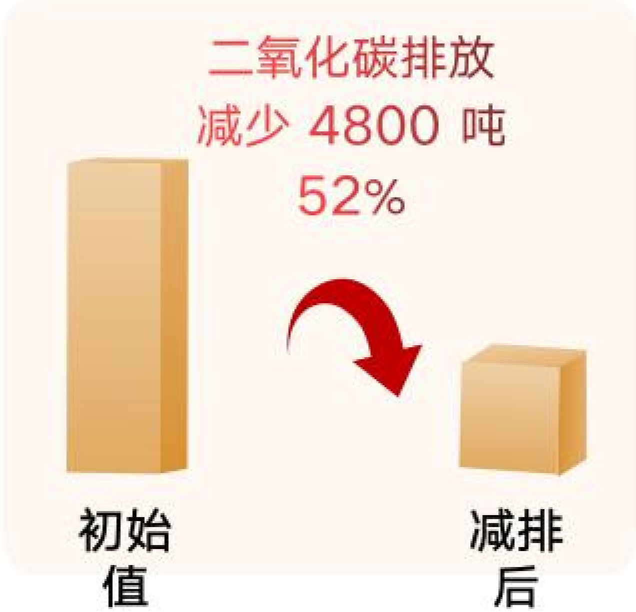
我希望为母校打造的这个计算中心，…成为**交大一道独特风景线**。

——杨元庆

2021.12.14 开机仪式

有什么特色？低碳减排

采用国际最先进的温水冷却技术；
回收超算产生的热量，为大楼供暖



温水冷技术减排： 3900吨，42%

余热回收碳补偿： 950吨，10%

每年节省能源成本：150万

这台超算有多快？国内高校领先（TOP500排名第132位）

计算

- 6 PFlops CPU+GPU 双精度
- CPU：938台2路Intel 8358
- GPU：23台4卡Nvidia A100

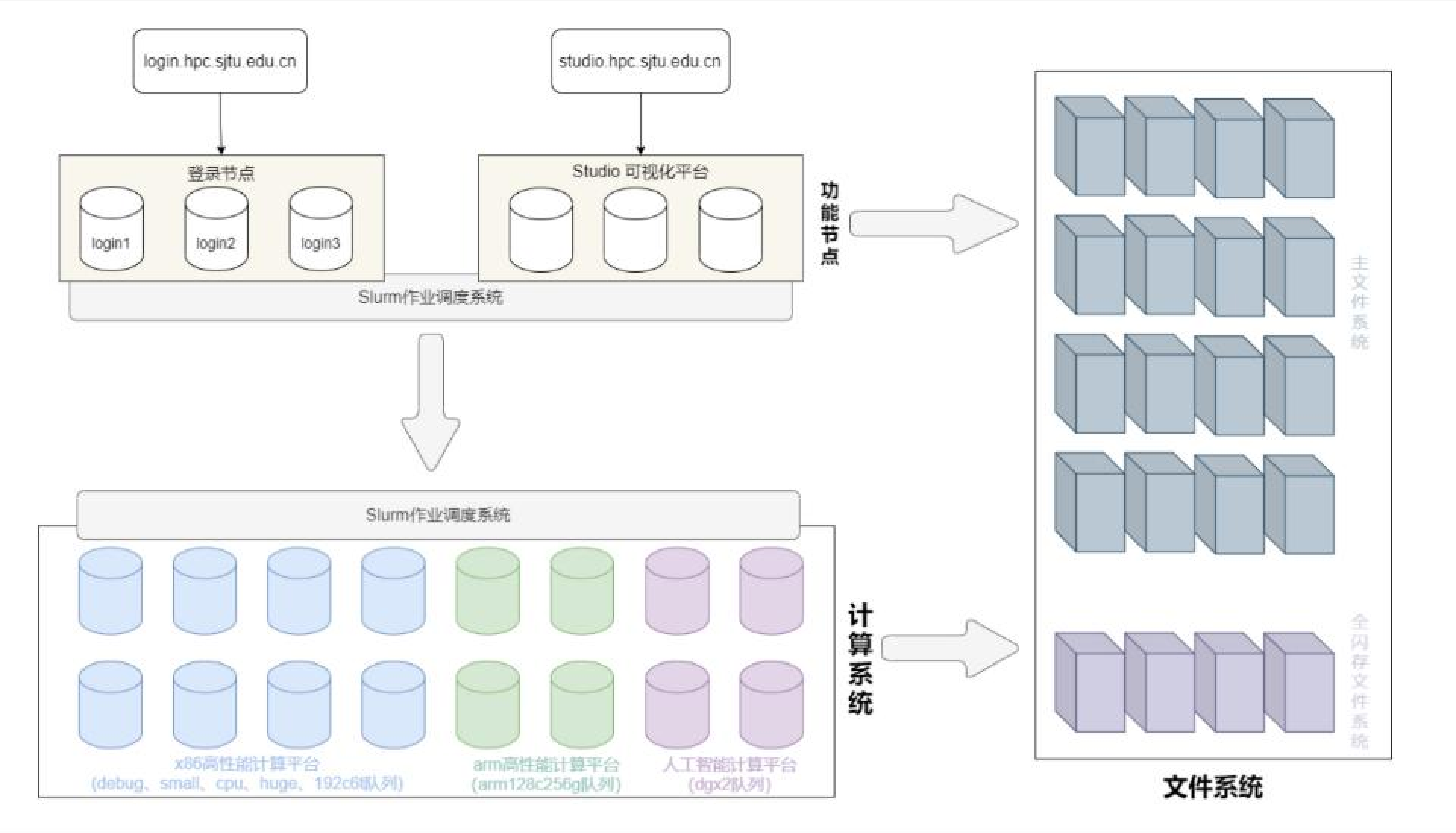
存储

- 存储容量 10 PB
- GPFS 并行文件系统

互联

- Mellanox HDR 交换机
- 计算节点 100Gbps 高速互联

交我算集群架构



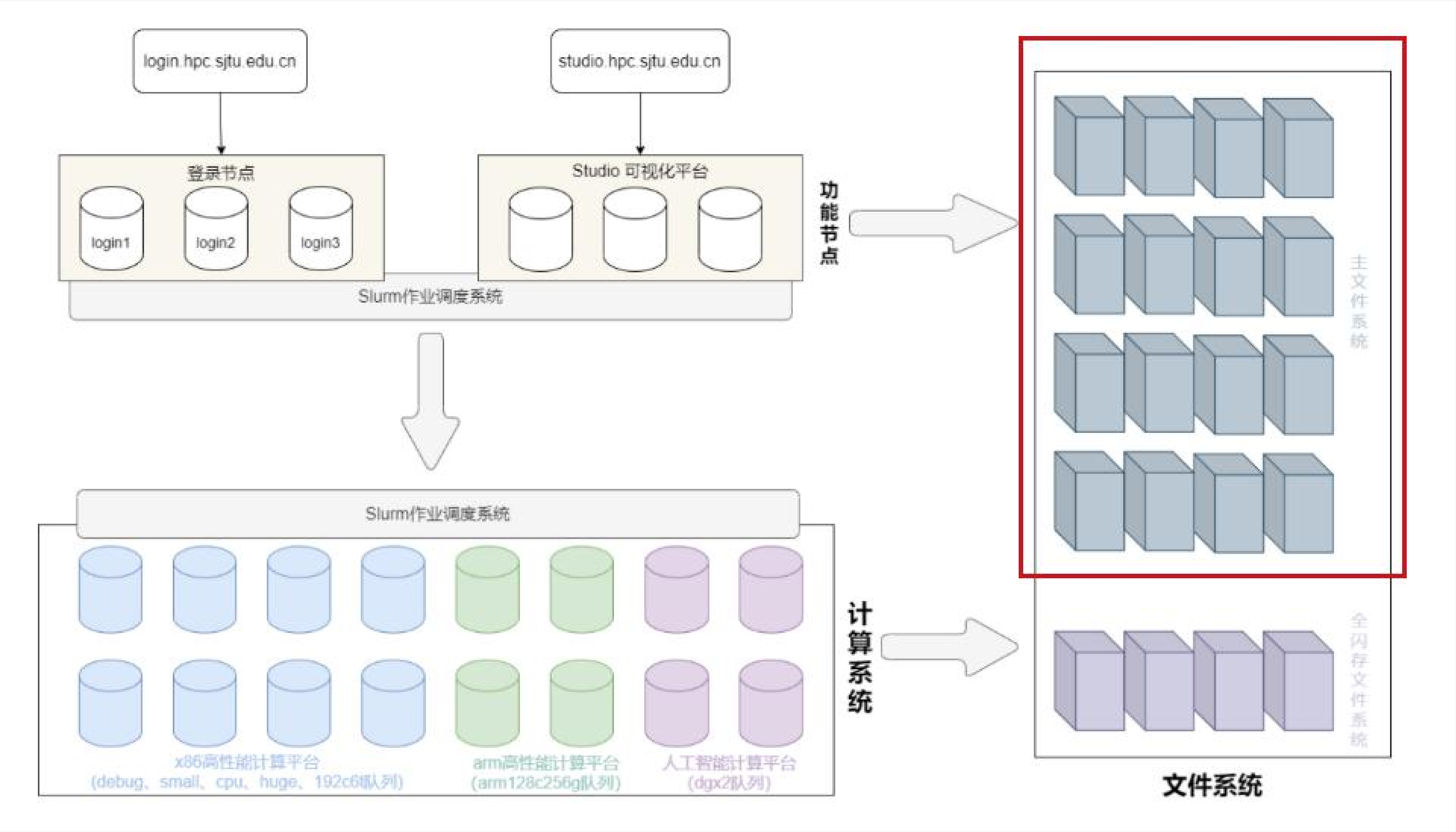
login.hpc.sjtu.edu.cn
studio.hpc.sjtu.edu.cn
登录节点
login1
login2
login3
Studio 可视化平台
Slurm作业调度系统
功能节点
主文件系统
Slurm作业调度系统
计算系统
全闪存文件系统
x86高性能计算平台
(debug、small、cpu、huge、192c6队列)
arm高性能计算平台
(arm128c256g队列)
人工智能计算平台
(dgx2队列)
文件系统

交我算集群文件系统

集群	文件系统	特点	个人目录路径	快捷环境变量
闪行集群（π2.0+ARM+人工智能）	主文件系统	默认文件系统 登录后的默认系统 HDD盘，大容量、高可用、较高性能	/lustre/home/acct-xxxx/yyyy	$HOME
	全闪存文件系统	临时文件系统 适合作为临时工作目录 SSD盘，高性能、容量较小、安全性不高	/scratch/home/acct-xxxx/yyyy	$SCRATCH
思源一号	主文件系统	思源一号目前唯一的文件系统	/dssg/home/acct-xxxx/yyyy	$SIYUANHOME

*相关内容参考：https://docs.hpc.sjtu.edu.cn/job/slurm.html

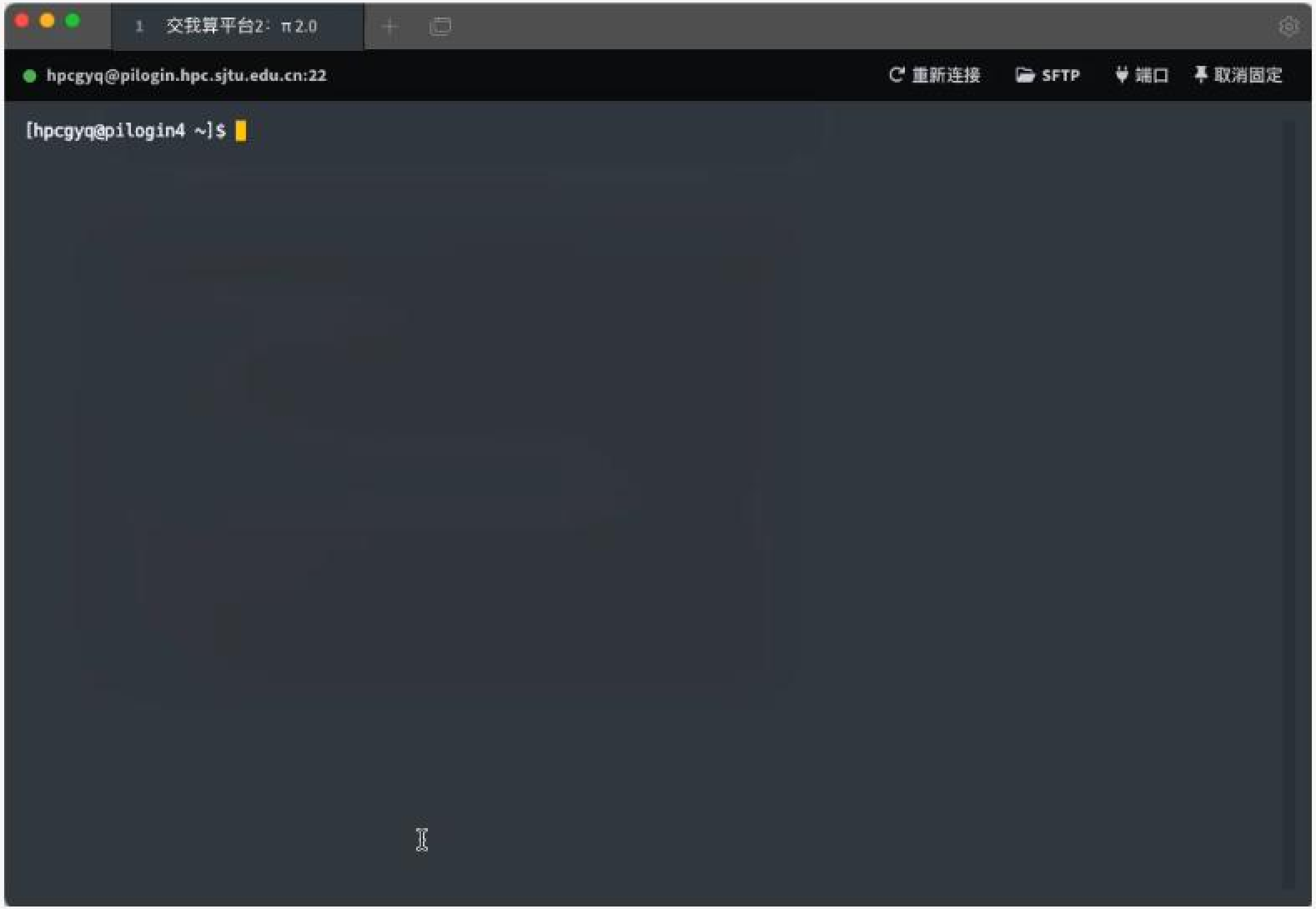
交我算集群架构



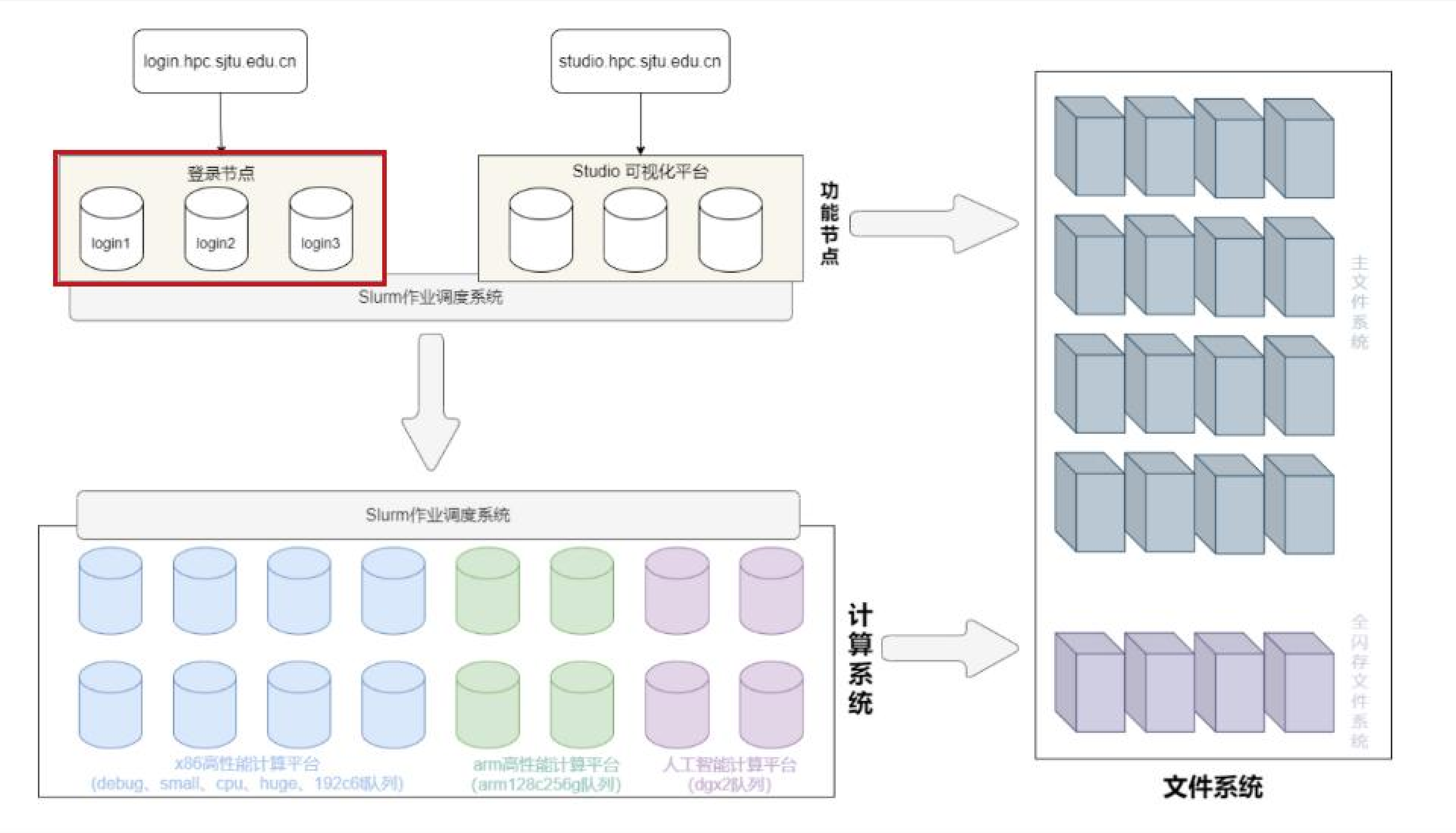
login.hpc.sjtu.edu.cn
studio.hpc.sjtu.edu.cn
登录节点
login1
login2
login3
Studio 可视化平台
功能节点
Slurm作业调度系统
Slurm作业调度系统
计算系统
x86高性能计算平台
(debug、small、cpu、huge、192c6队列)
arm高性能计算平台
(arm128c256g队列)
人工智能计算平台
(dgx2队列)
主文件系统
全闪存文件系统
文件系统

Linux终端命令演示

Linux终端命令	功能
ls	列出目录内容
pwd	显示当前目录绝对路径
cd	切换到指定目录
mkdir	创建目录
cp	复制文件或目录
mv	移动或重命名文件或目录
rm	删除文件或目录



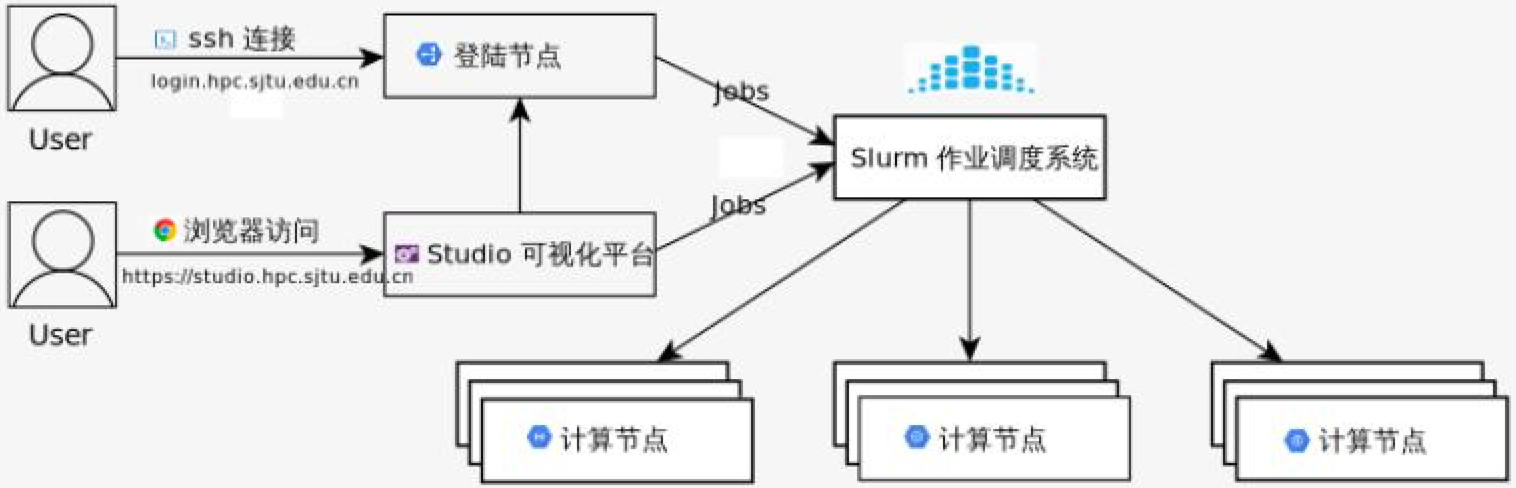
交我算集群架构



login.hpc.sjtu.edu.cn
studio.hpc.sjtu.edu.cn
登录节点
login1
login2
login3
Studio 可视化平台
Slurm作业调度系统
功能节点
Slurm作业调度系统
x86高性能计算平台
(debug、small、cpu、huge、192c6队列)
arm高性能计算平台
(arm128c256g队列)
人工智能计算平台
(dgx2队列)
计算系统
主文件系统
全闪存文件系统
文件系统

Slurm作业提交流程

- 用户将需要执行的工作编写进一个Slurm作业脚本中
- 用户从登录节点或可视化平台提交作业
- 由Slurm将作业分配到不同计算节点上运行



*相关内容参考：https://docs.hpc.sjtu.edu.cn/job/slurm.html

Slurm作业运行队列

➢ 用户提交作业脚本时需要指定作业运行的队列

集群	节点类型	节点数	单节点核数	单节点内存	队列	允许单作业核数	可否共享	最长运行时间
π2.0	CPU节点（x86）	656个	40核	192G	small	1–20	可共享	7天
					cpu	40–24000	需独占	7天
					debug	测试节点	可共享	20分钟
	CPU节点（大内存）	3个	80核	3T	huge	6–80	可共享	2天
			192核	6T	192c6t	48–192	可共享	2天
AI平台	GPU节点	8个，每节点配16张V100卡	96核	1.45T	dgx2	最高CPU配比为1:6，GPU卡数为1–128	可共享	7天
ARM平台	CPU节点（ARM）	100个	128核	256G	arm128c256g	1–12800	可共享	3天
					debugarm	测试节点	可共享	20分钟
思源一号	CPU节点	936个	64核	512G	64c512g	1–60000	可共享	7天
					debug64c512g	测试节点	可共享	1小时
	GPU节点	23个，每节点配4张A100卡	64核	160G	a100	最高CPU配比为1:16，GPU卡数为1–92	可共享	7天

*相关内容参考：https://docs.hpc.sjtu.edu.cn/job/slurm.html

报告提纲

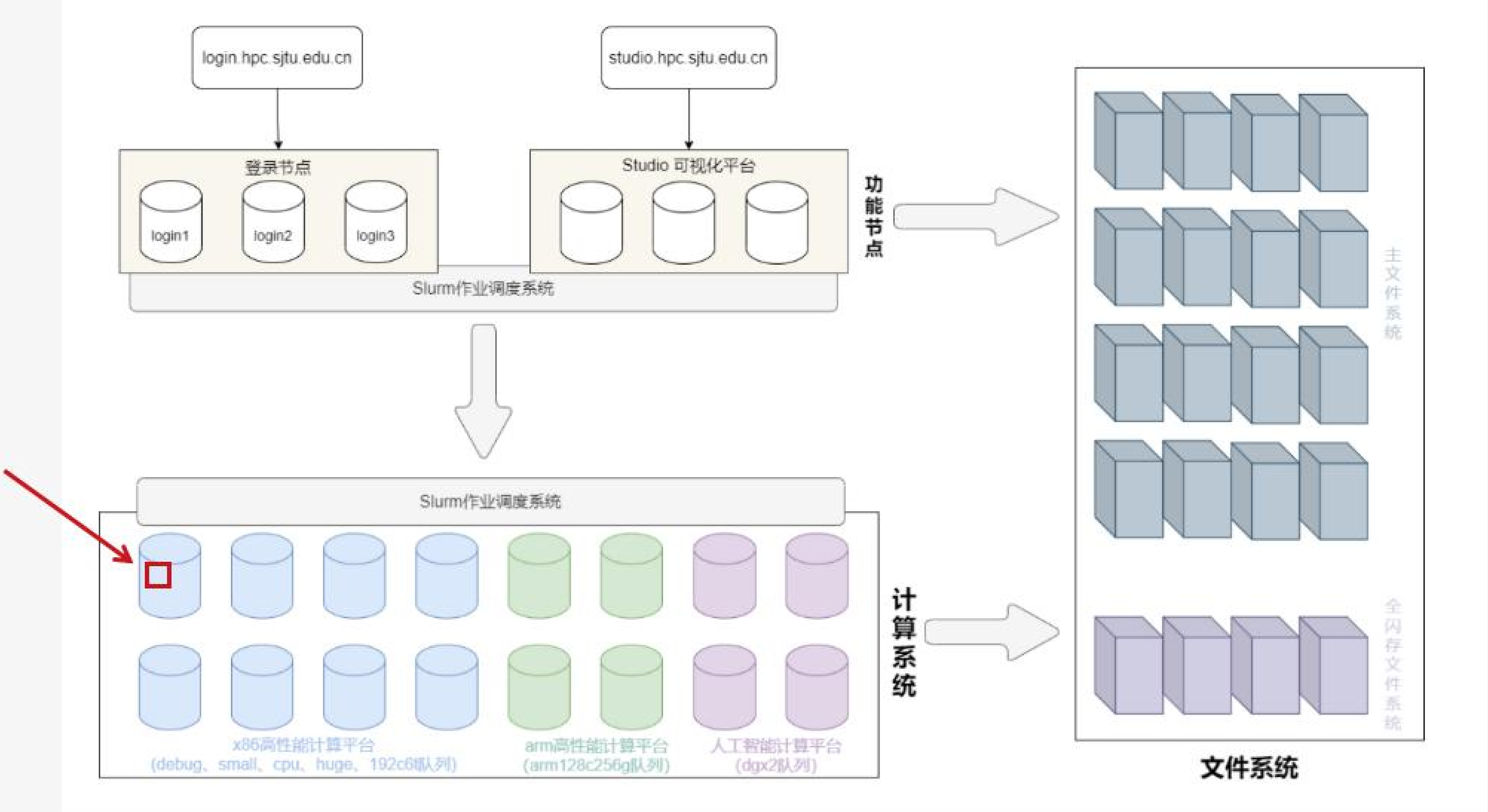
以“思源一号”为例

计算资源的基础概念

编译器基础

“交我算”课堂实践

单核串行（single-core/serial version）



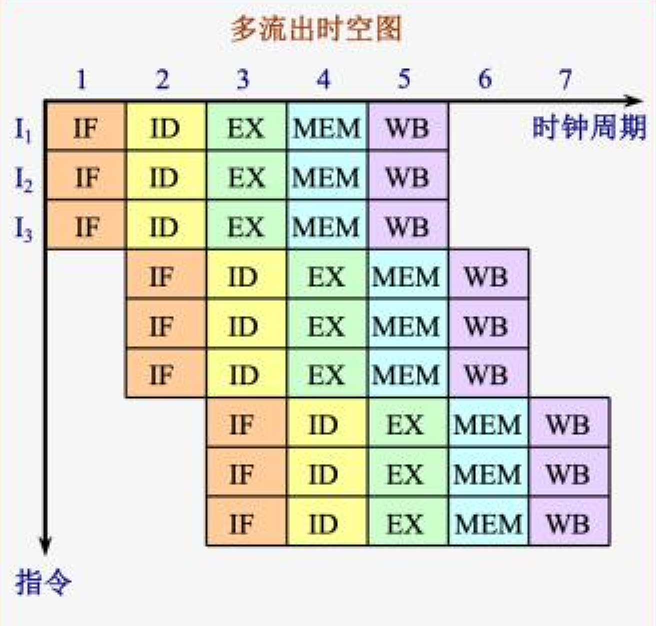
CPU峰值性能计算公式

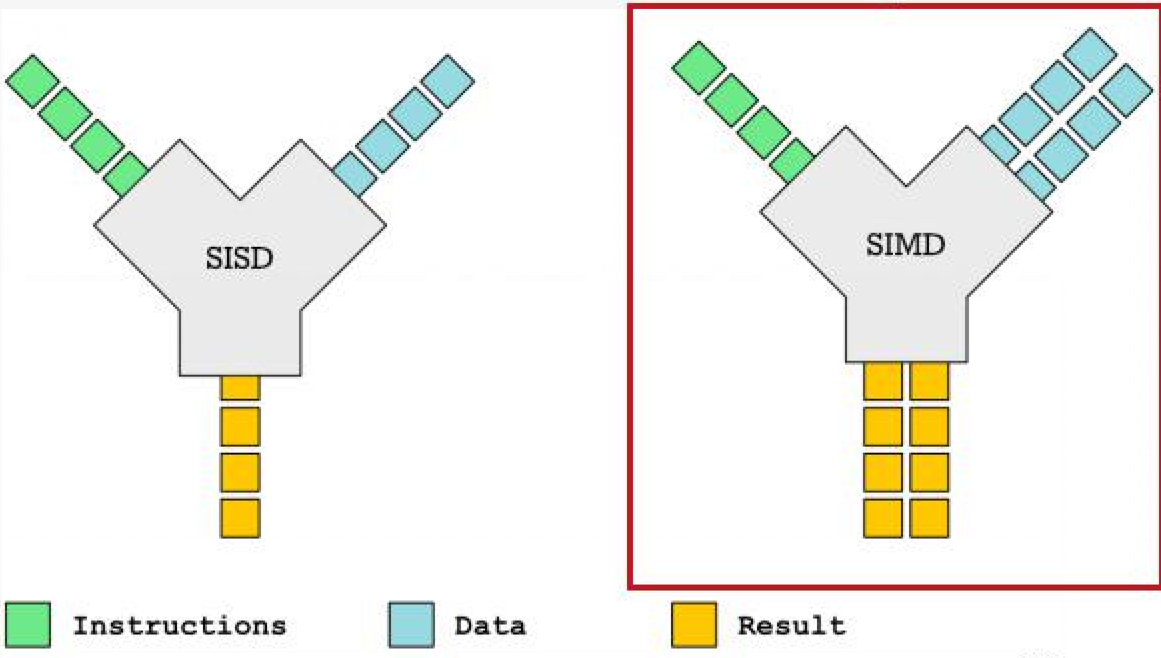
CPU峰值性能 = 处理器主频 × 单周期指令数 × 单指令处理位宽 × 核数

672 GFlops　　2.8 GHz　　3　　8（512位）　　10

(双精度 DP/FP64)

双精度浮点数 64bit
单精度浮点数 32bit
半精度浮点数 16bit





单核串行代码的运行效率

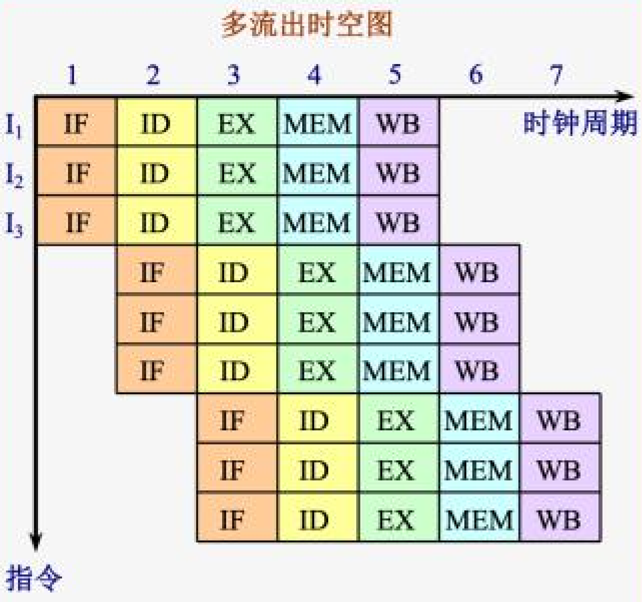
CPU峰值性能 = 处理器主频 × 单周期指令数 × 单指令处理位宽 × 核数

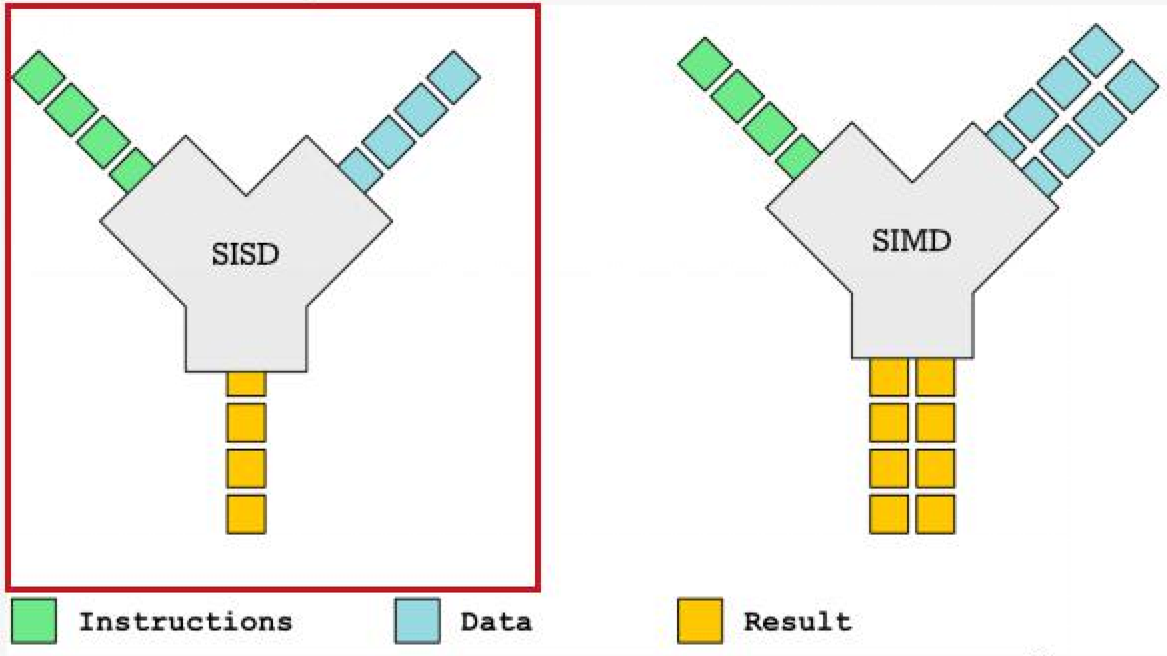
~~672 GFlops~~ （双精度 DP/FP64）　2.8 GHz　3　~~8（512位）~~　10

1（64位）

8.4 GFlops

80倍的差异！



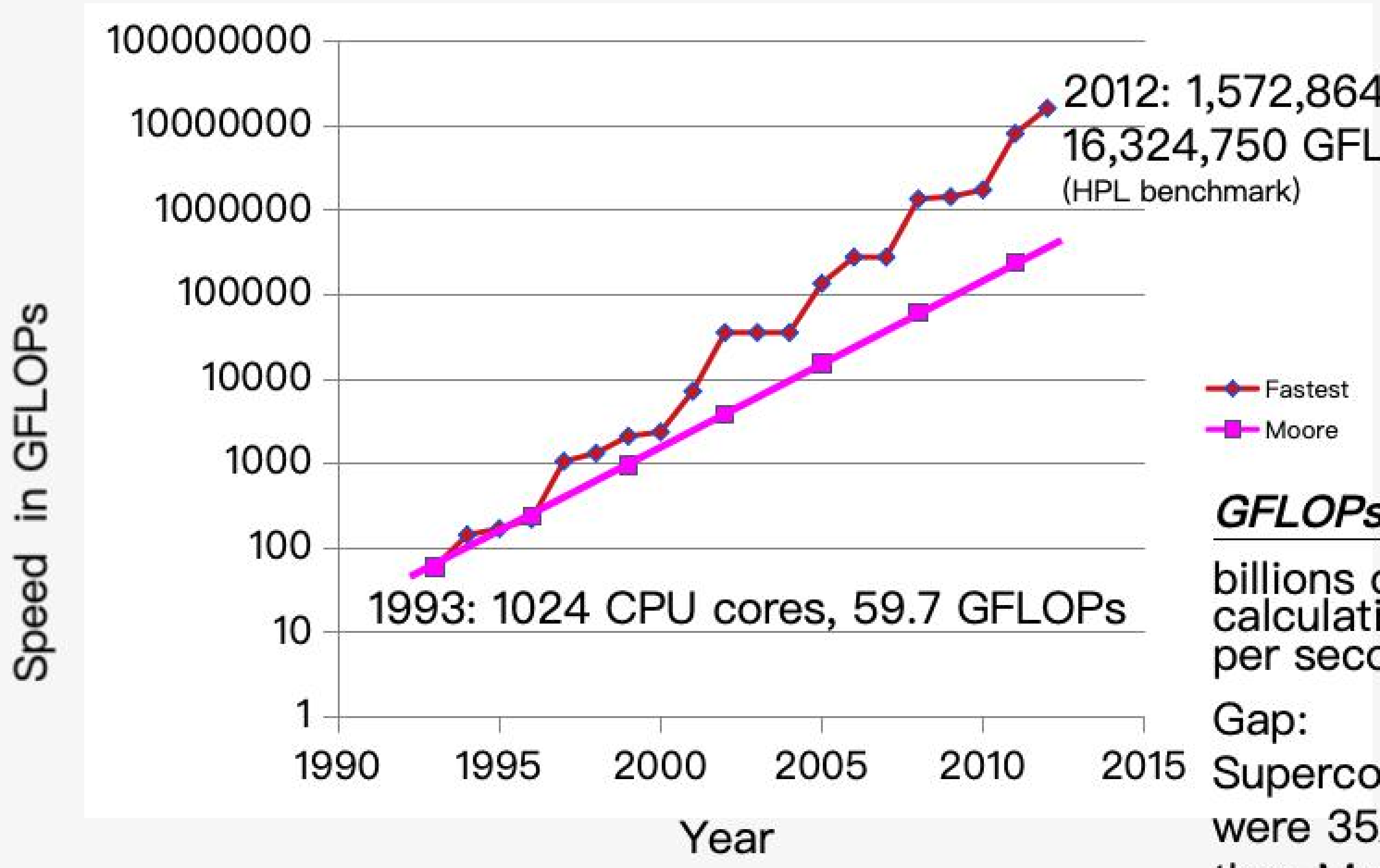


摩尔定律（Moore’s Law）

- In 1965, Gordon Moore was an engineer at Fairchild Semiconductor, later was co-founder and CEO of Intel
- He noticed that the number of transistors that could be squeezed onto a chip was doubling about every 2 years.
- It turns out that computer speed is roughly proportional to the number of transistors per unit area.
- Moore wrote a paper about this concept, which became known as ***"Moore's Law."***

Further Reading:

Fastest Supercomputer vs. Moore

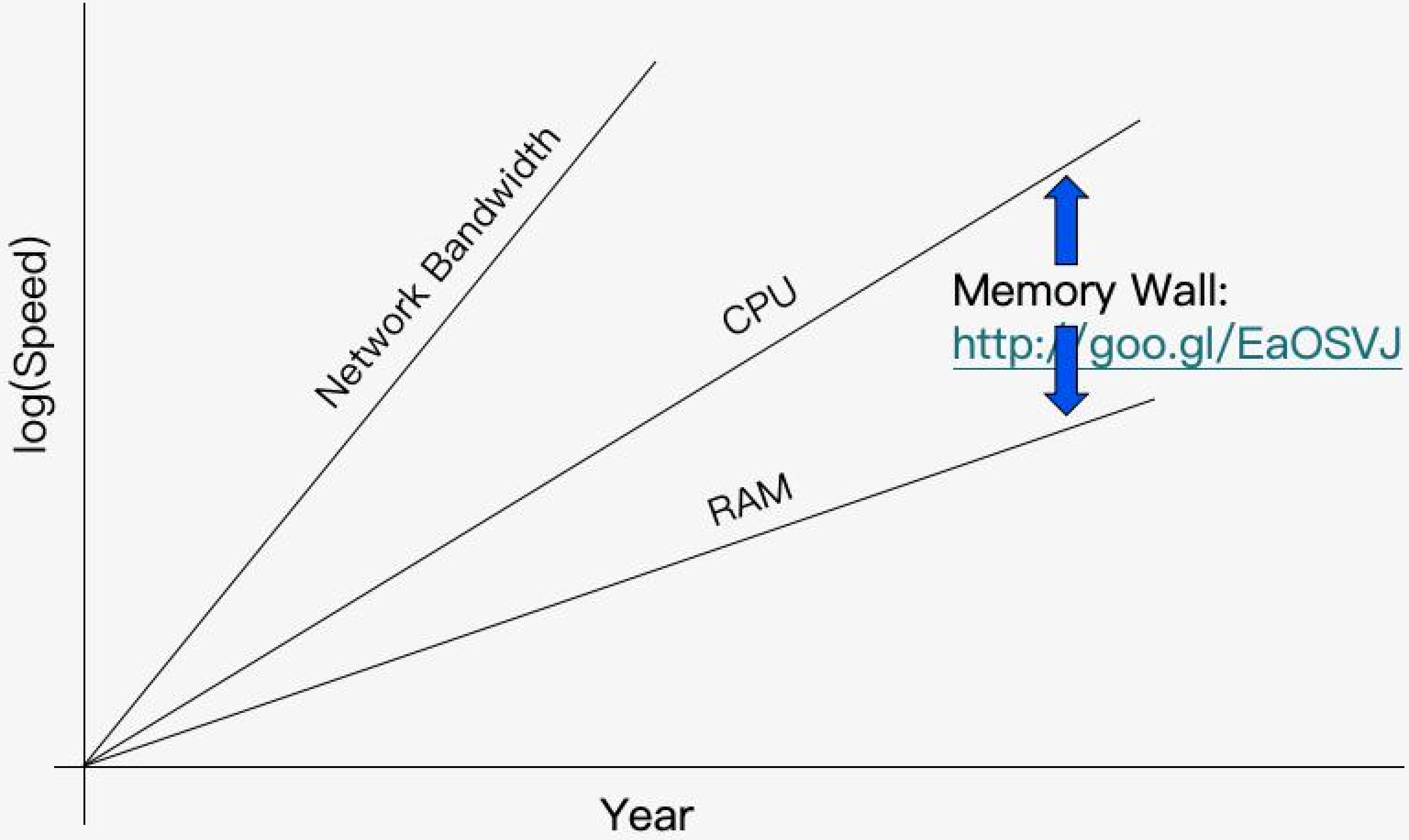


***GFLOPs*:**

billions of calculations per second

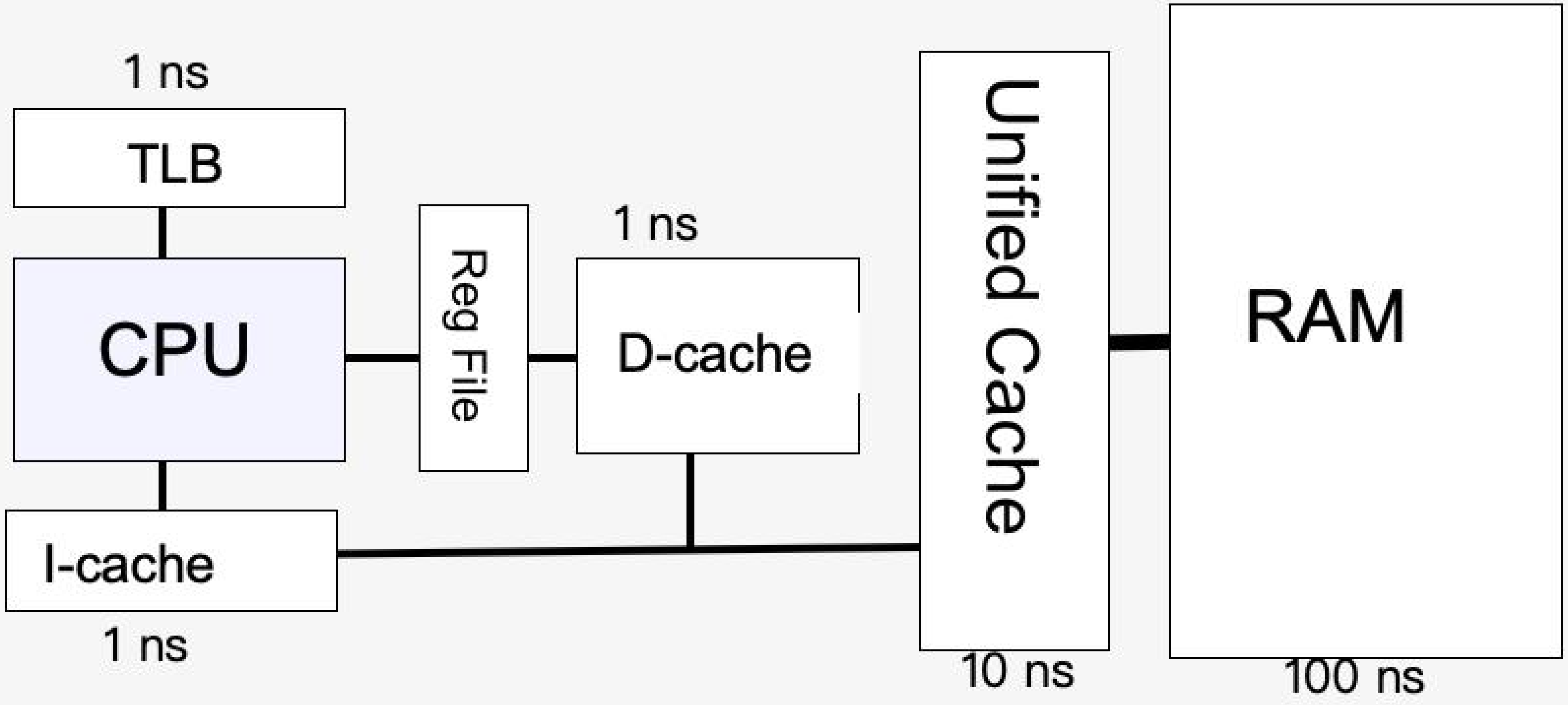
Gap: Supercomputers were 35x higher than Moore in 2011.

Moore’s Law in Practice



CPU的内存架构

A typical microprocessor memory hierarchy



- Instruction cache and data cache pull data from a unified cache that maps onto RAM.
- TLB implements virtual memory and brings in pages to support large memory foot prints.

阿姆达尔定律（Amdahl's Law）

Gene M. Amdahl

(1922–2015)

并行计算先驱

曾任 IBM 计算机

架构师

于1967年提出

Amdahl's Law

- What is the maximum speedup you can expect from a parallel program?
- Approximate the runtime as a part that can be sped up with additional processors and a part that is fundamentally serial.

$$Time_{par}(P) = (serial\_fraction + \frac{parallel\_fraction}{P}) * Time_{seq}$$

- If serial_fraction is α and parallel_fraction is (1- α) then the speedup is:

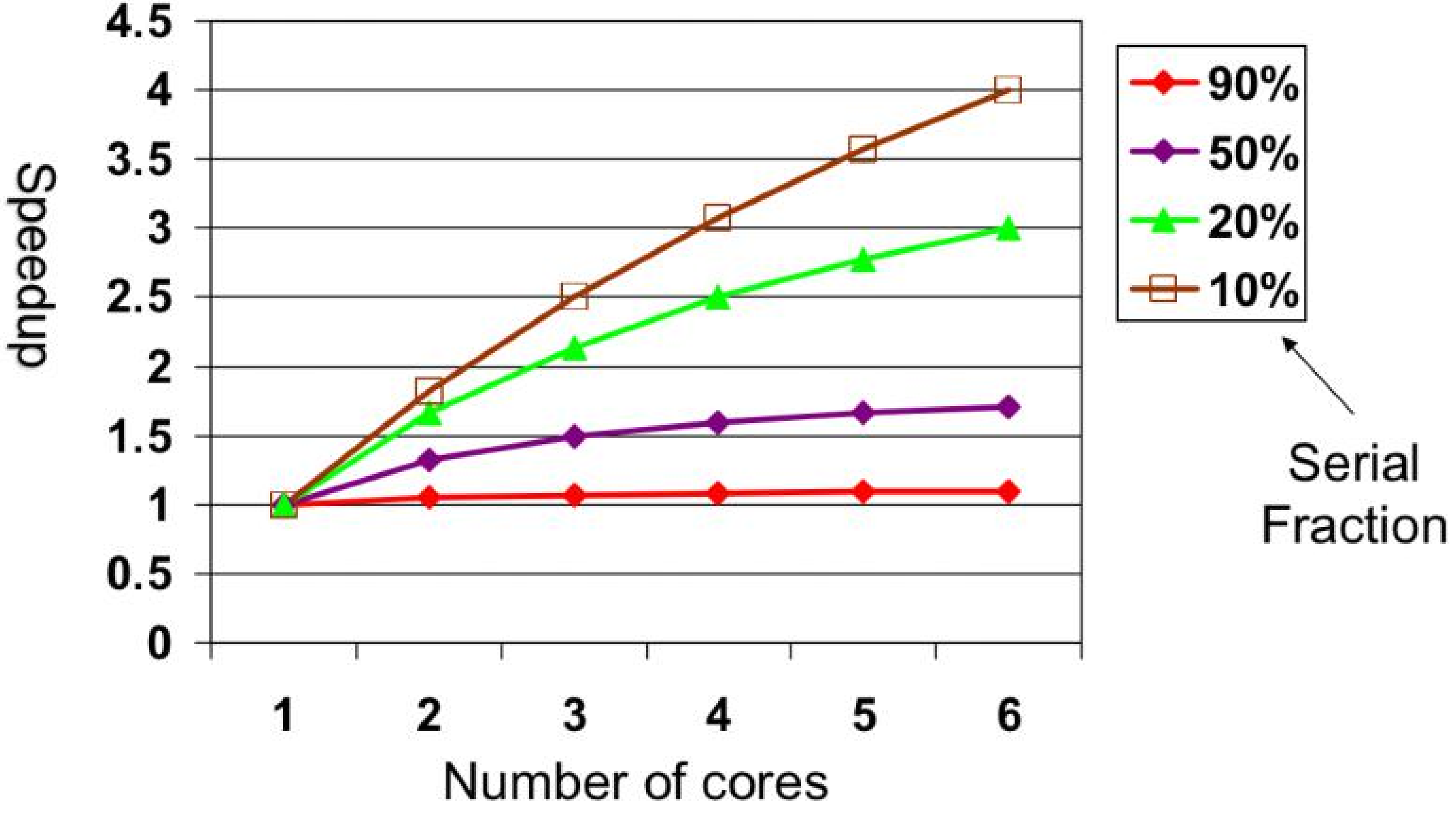
$$S(P) = \frac{Time_{seq}(1)}{(a + \frac{1-a}{P}) * Time_{seq}(1)} = \frac{1}{a + \frac{1-a}{P}}$$

- If you had an unlimited number of processors: $P \to \infty$

- The maximum possible speedup is: $S = \frac{1}{a}$ ← Amdahl's Law

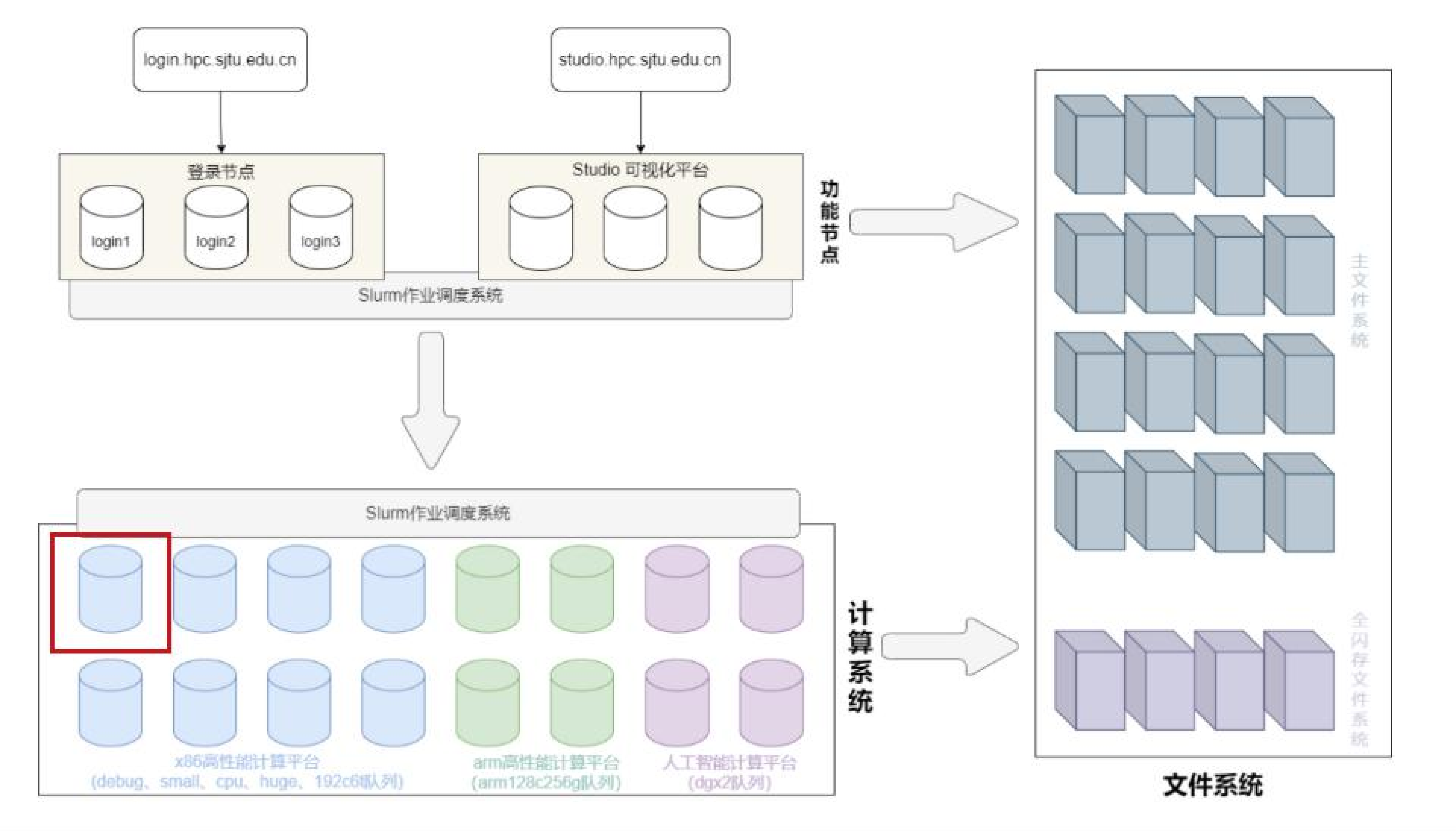
阿姆达尔定律（Amdahl's Law）的应用意义

- Consider benefits of adding processors to your parallel program for different serial fractions.
- Note: getting a serial fraction under 10% is challenging for the typical application



Acknowledgement: , 2009 DAC Tutorial – Tom Spyrou

单节点多核并行（Multi-core）



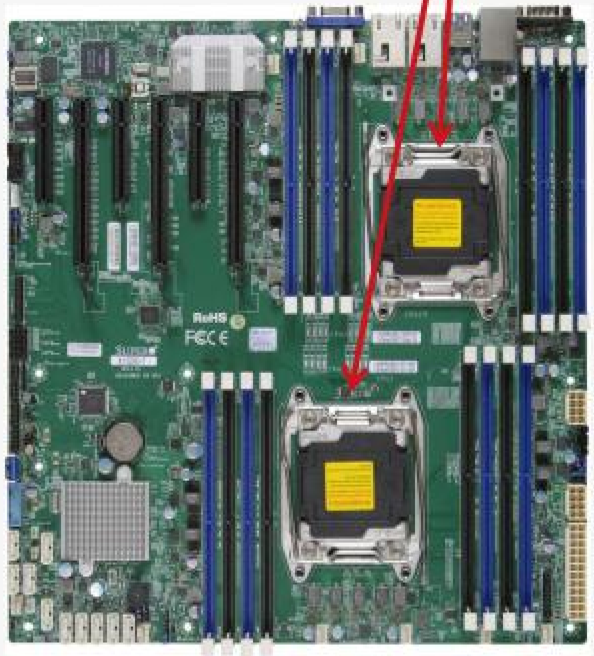
单节点多核的运行效率

CPU峰值性能 = 处理器主频 × 单周期指令数 × 单指令处理位宽 × 核数

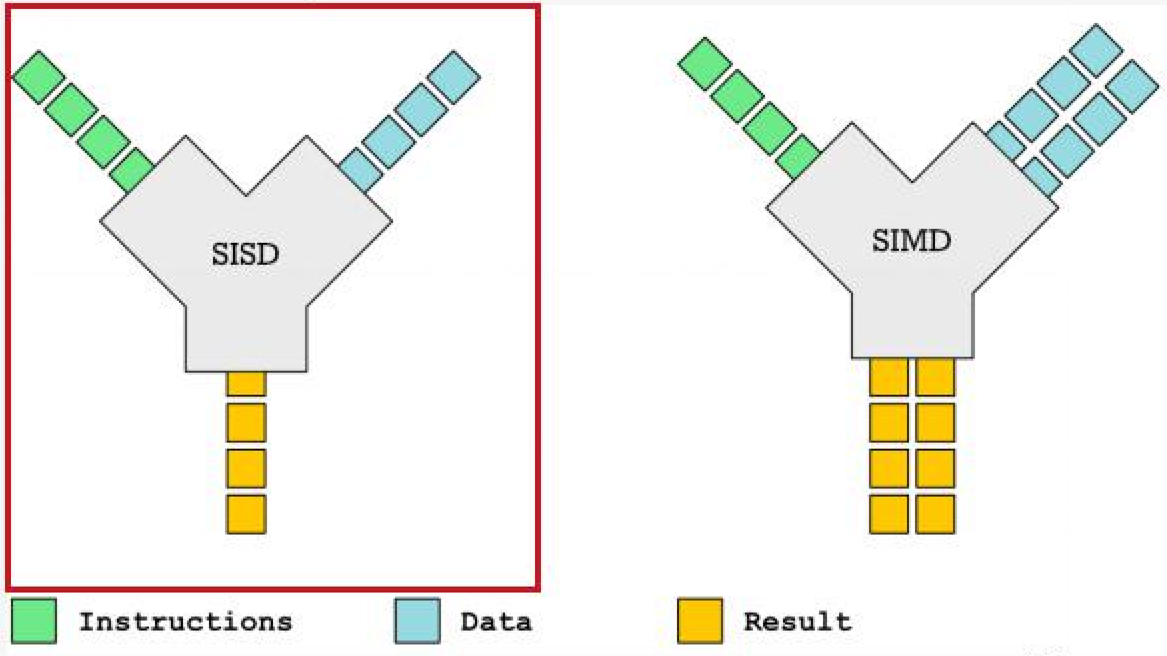
~~672 GFlops~~ (双精度 DP/FP64)	2.8 GHz	3	~~8（512位）~~	10

1（64位）

84 GFlops × 2 = 168 GFlops

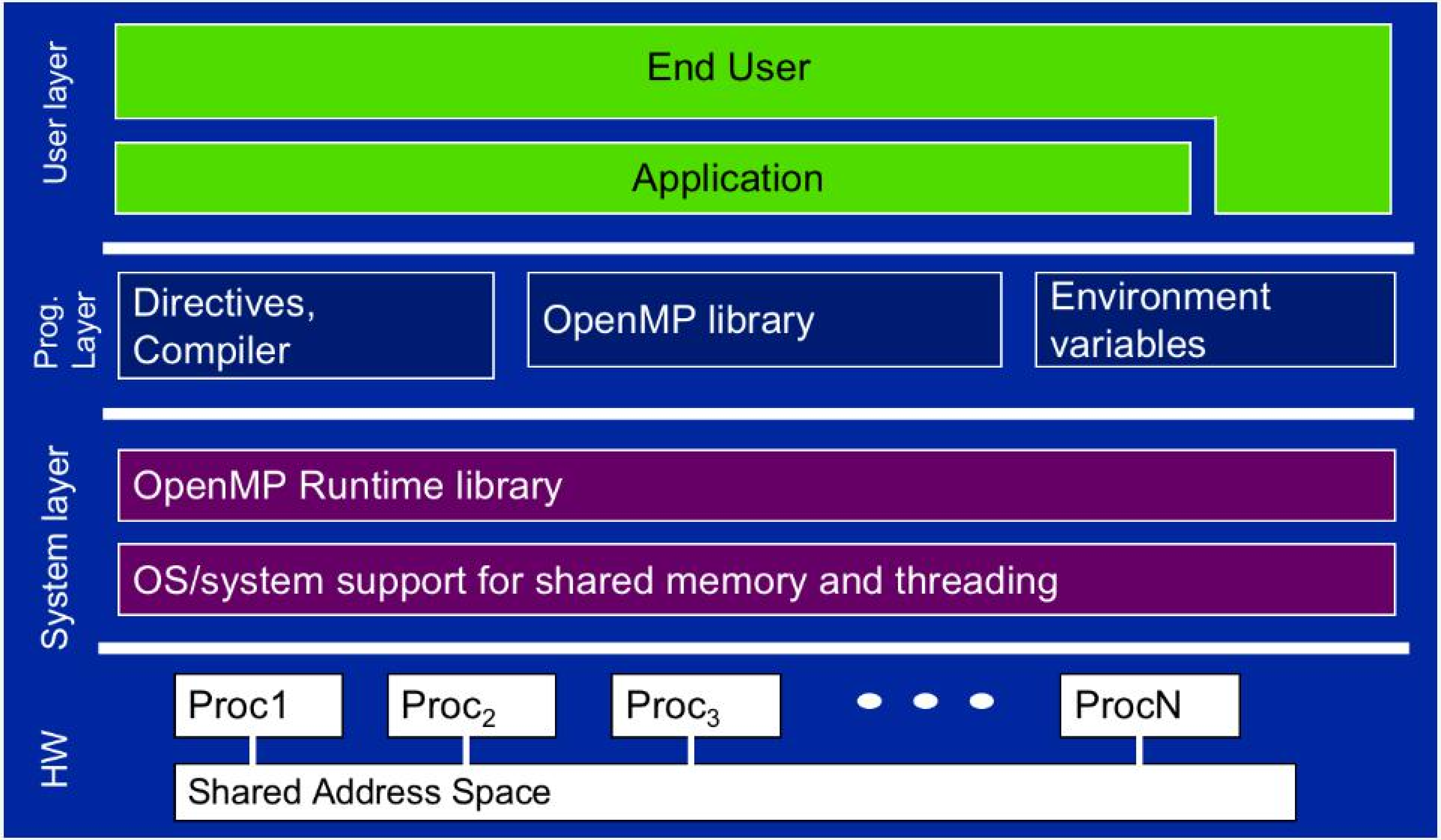


双路服务器



OpenMP并行编程模型

OpenMP: An API for Writing Multithreaded Applications



OpenMP并行编程模型示例

- **Write a multithreaded program where each thread prints “hello world”.**

```
#include <omp.h>
#include <stdio.h>
int  main()
{
#pragma omp parallel
 {
    int ID = omp_get_thread_num();
    printf(" hello(%d) ", ID);
    printf(" world(%d) \n", ID);
  }
}
```

OpenMP include file

Parallel region with default number of threads

Runtime library function to return a thread ID.

End of the Parallel region

Sample Output:

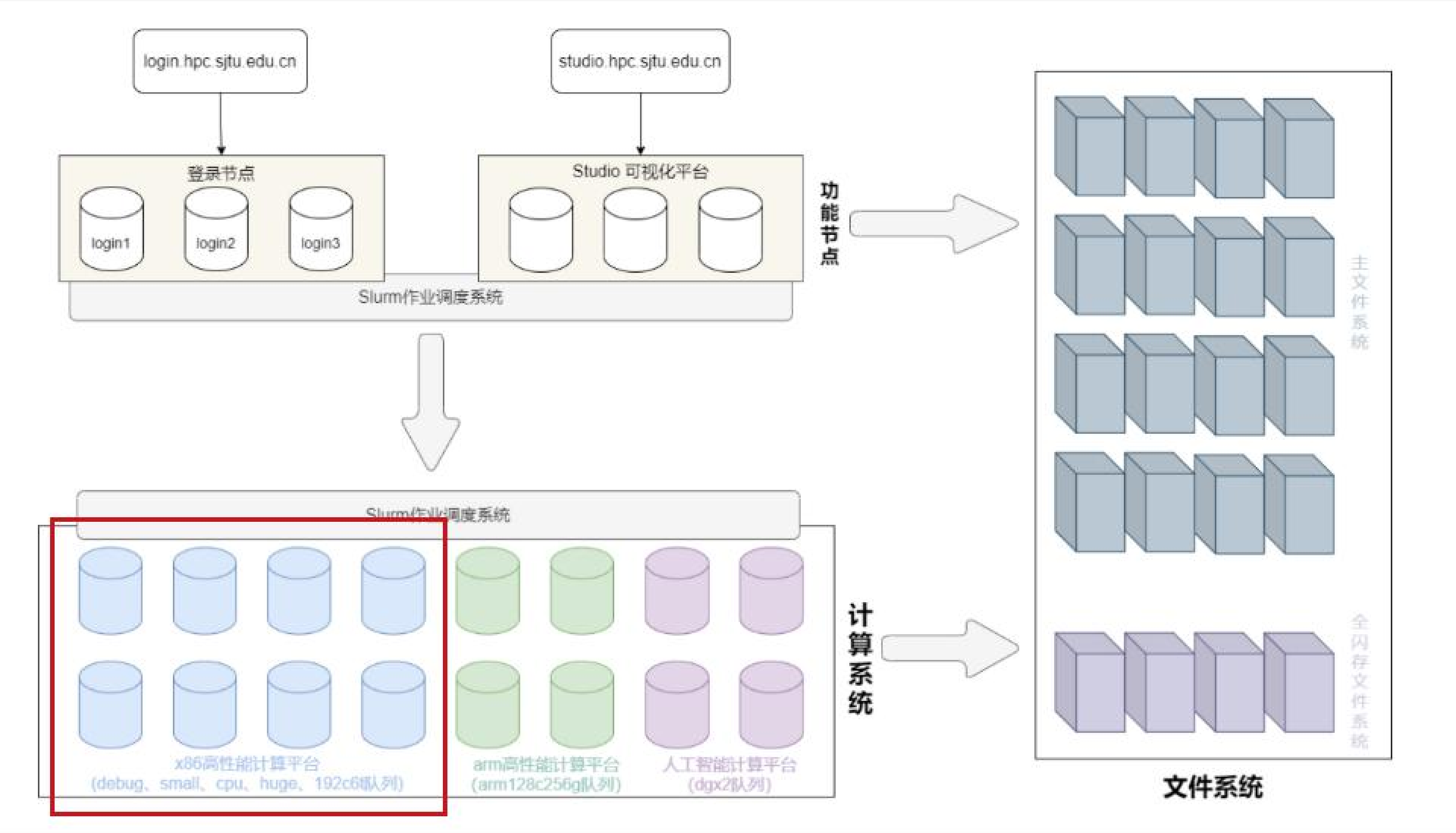
hello(1) hello(0) world(1)

world(0)

hello (3) hello(2) world(3)

world(2)

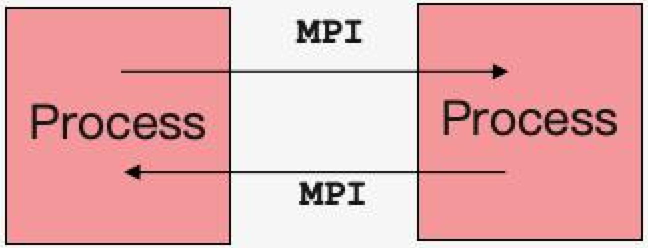
多节点并行（Multi-node）



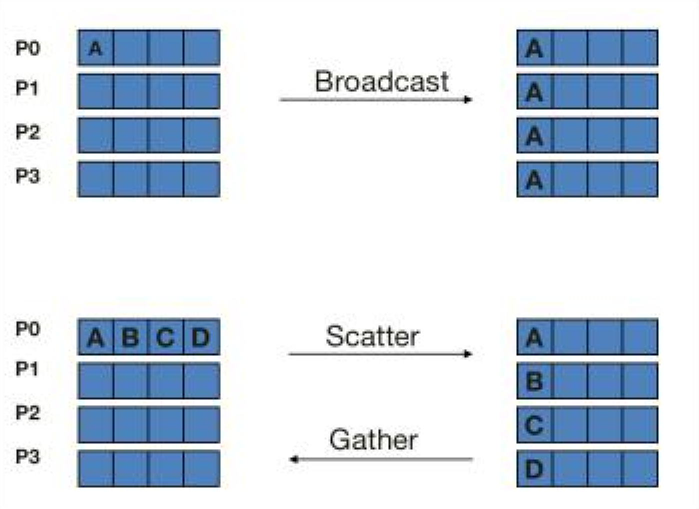
MPI并行编程模型

- MPI: Message Passing Interface
  - The MPI Forum organized in 1992 with broad participation by:
    - Vendors: IBM, Intel, TMC, SGI, Convex, Meiko
    - Portability library writers: PVM, p4
    - Users: application scientists and library writers
    - MPI-1 finished in 18 months
  - Incorporates the best ideas in a “standard” way
    - Each function takes fixed arguments
    - Each function has fixed semantics
      - Standardizes what the MPI implementation provides and what the application can and cannot expect
      - Each system can implement it differently as long as the semantics match
- MPI is not…
  - a language or compiler specification
  - a specific implementation or product

基础功能



进阶功能

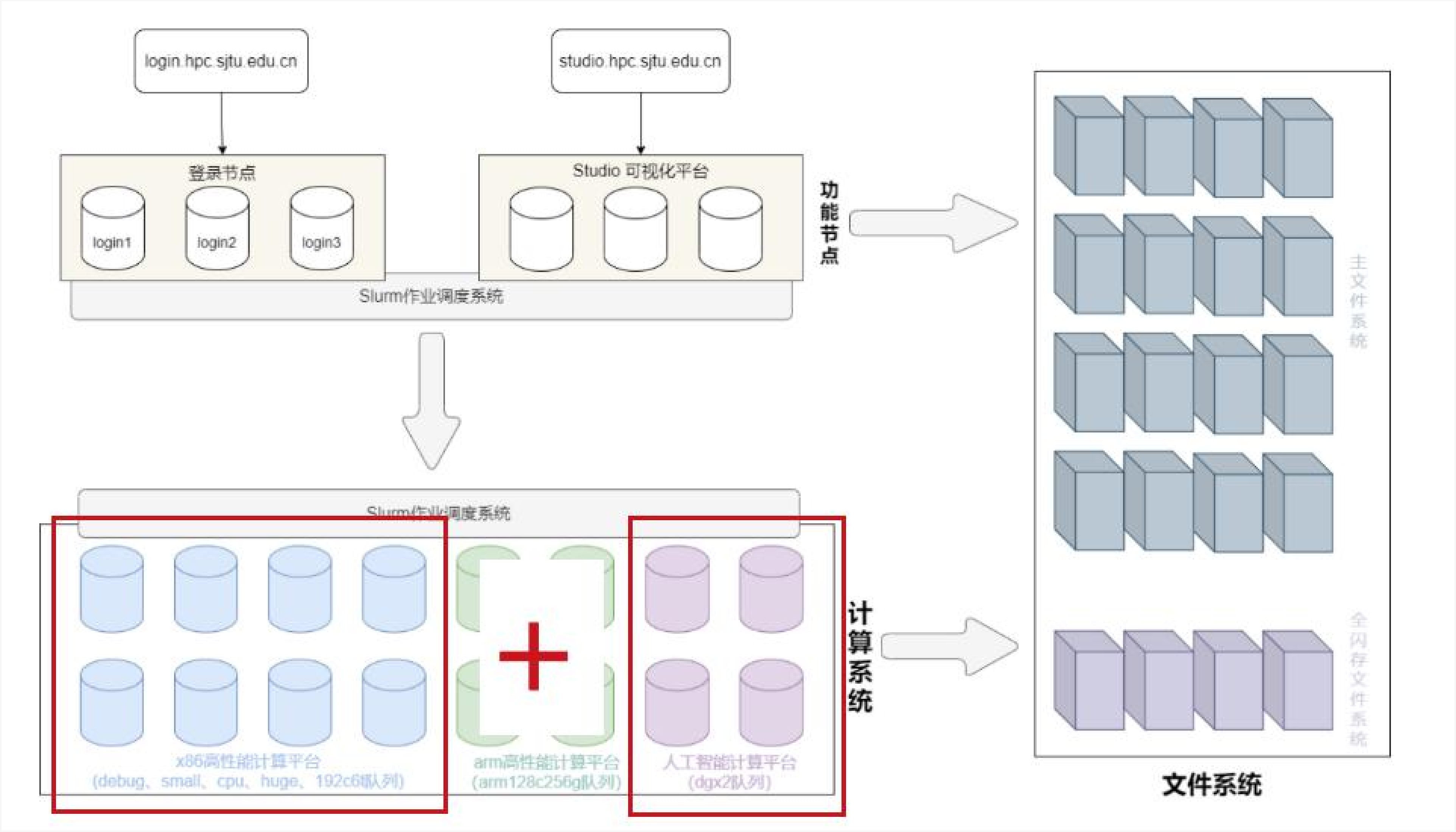


MPI并行编程模型示例

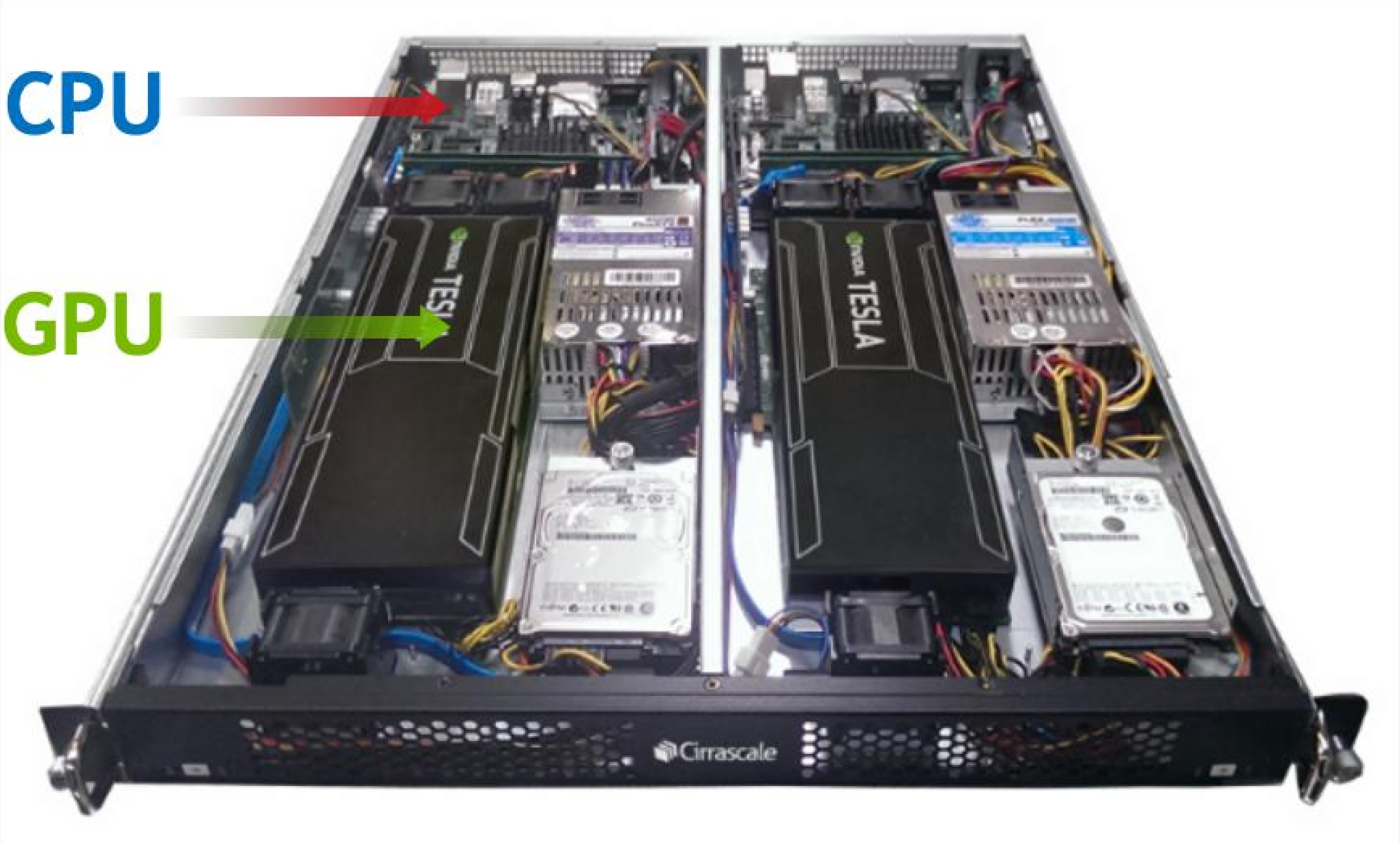
```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char ** argv)
{
    int rank, size;

    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    printf("I am %d of %d\n", rank, size);

    MPI_Finalize();
    return 0;
}
```

Basic requirements for an MPI program

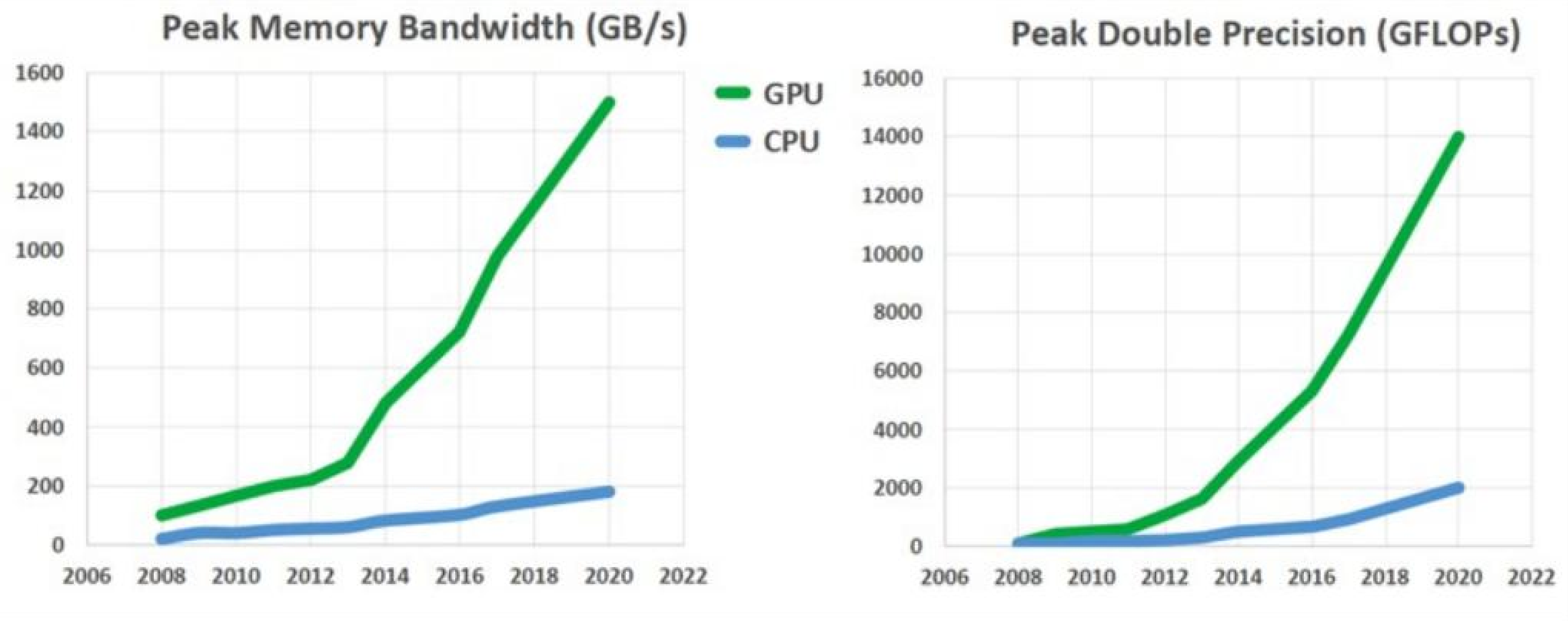
异构并行加速（Accelerator）



GPUs in HPC server

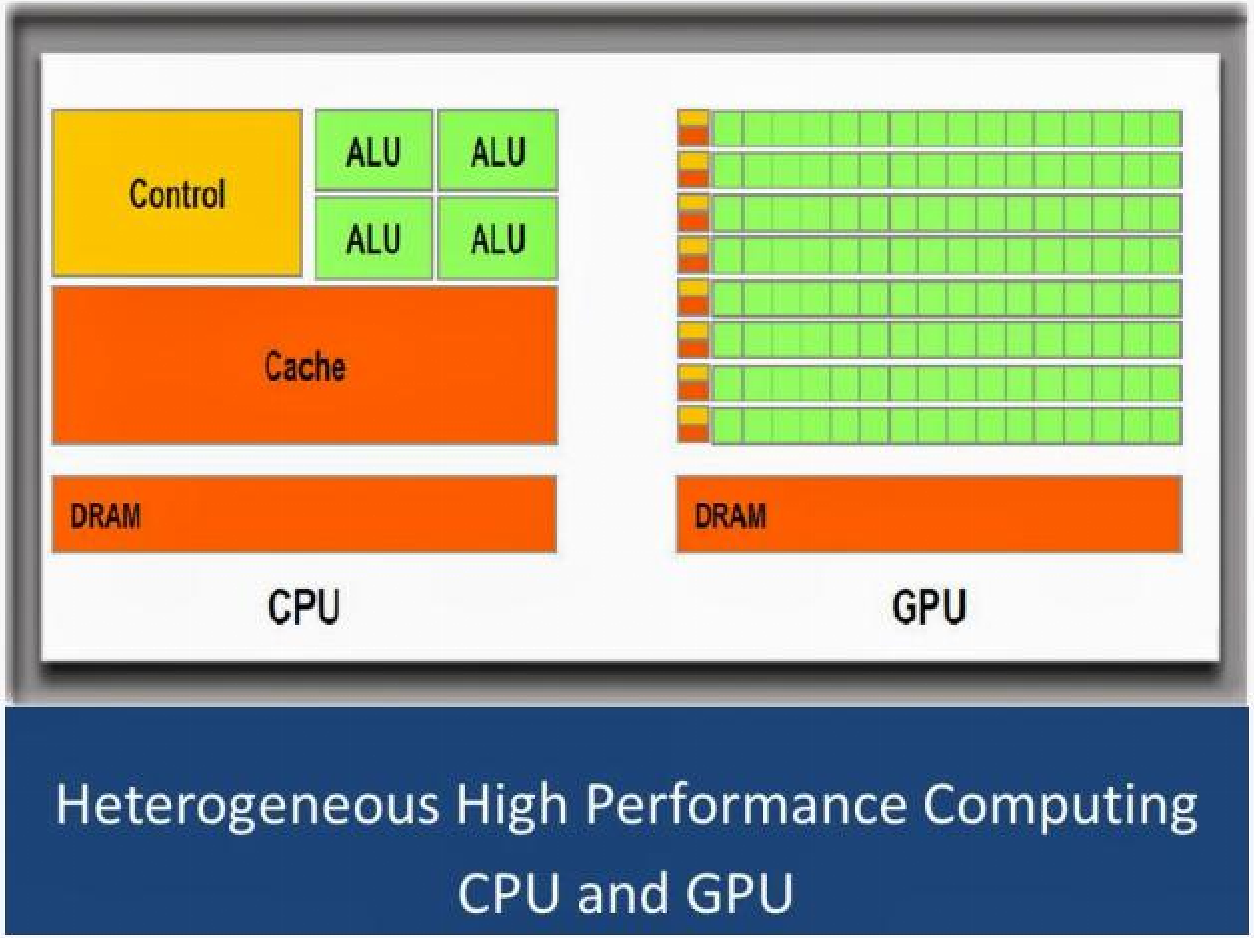


CPU vs GPU

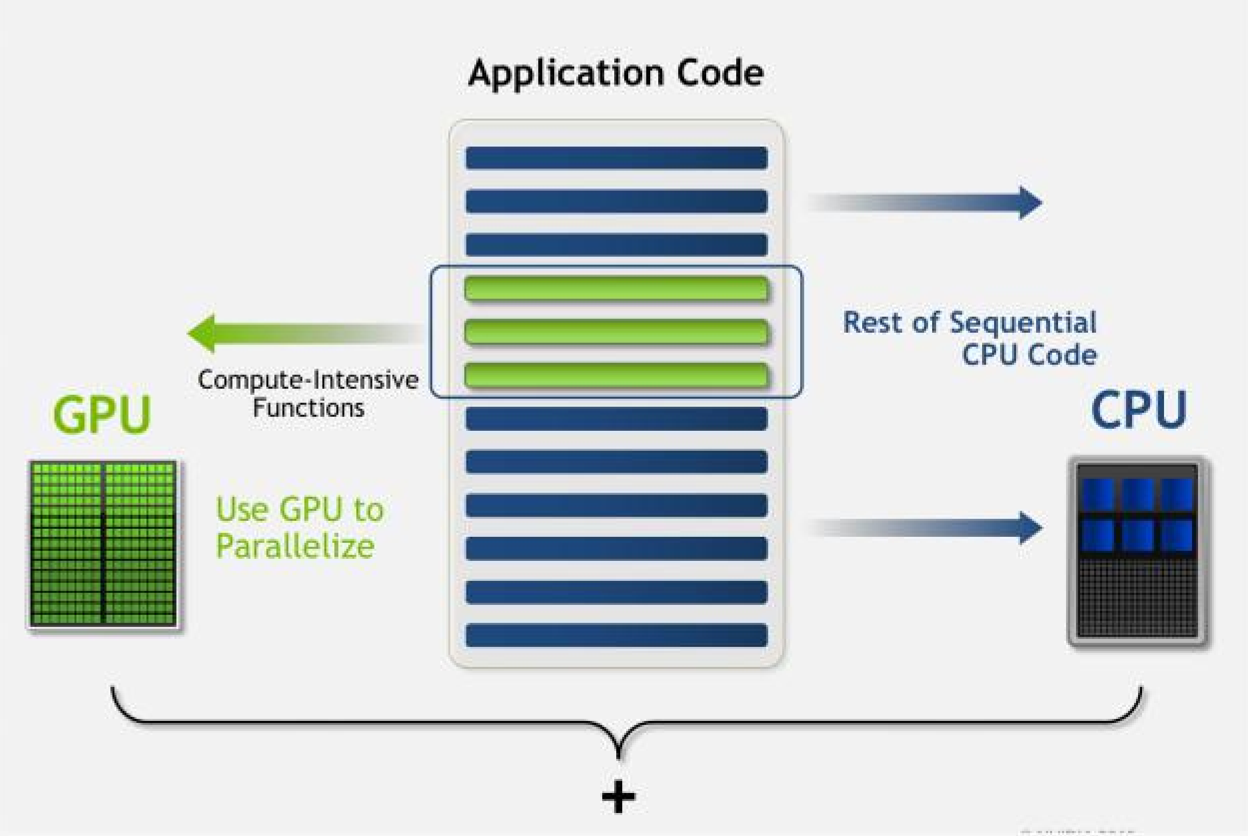

Peak Memory Bandwidth (GB/s)
1600
1400
1200
1000
800
600
400
200
0
2006
2008
2010
2012
2014
2016
2018
2020
2022
GPU
CPU
Peak Double Precision (GFLOPs)
16000
14000
12000
10000
8000
6000
4000
2000
0
2006
2008
2010
2012
2014
2016
2018
2020
2022

CUDA异构并行计算编程模型

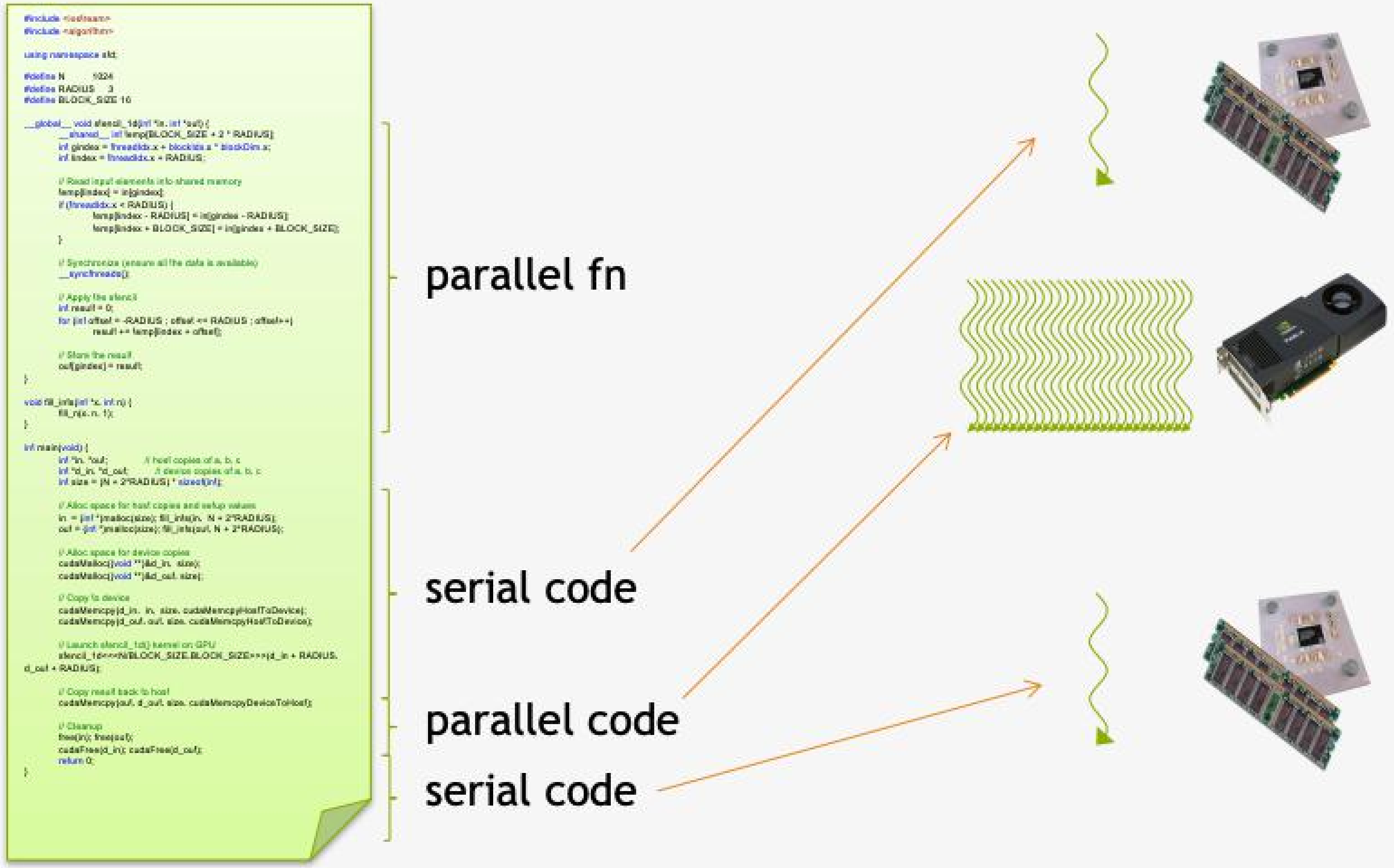
□CPU与GPU并行架构的区别



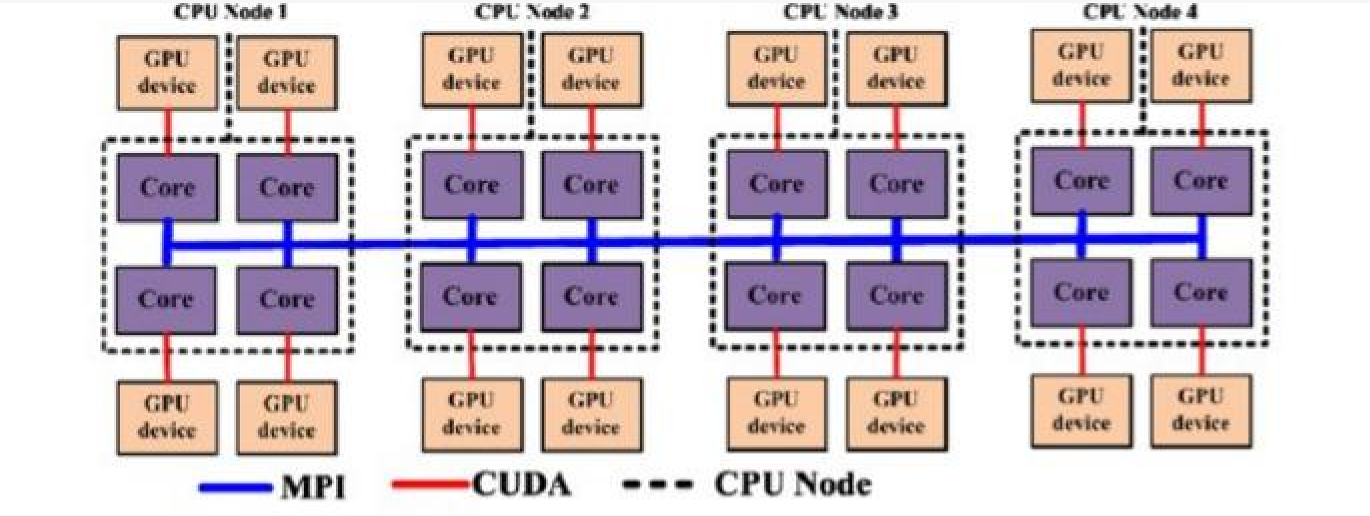
□CUDA异构并行的原理

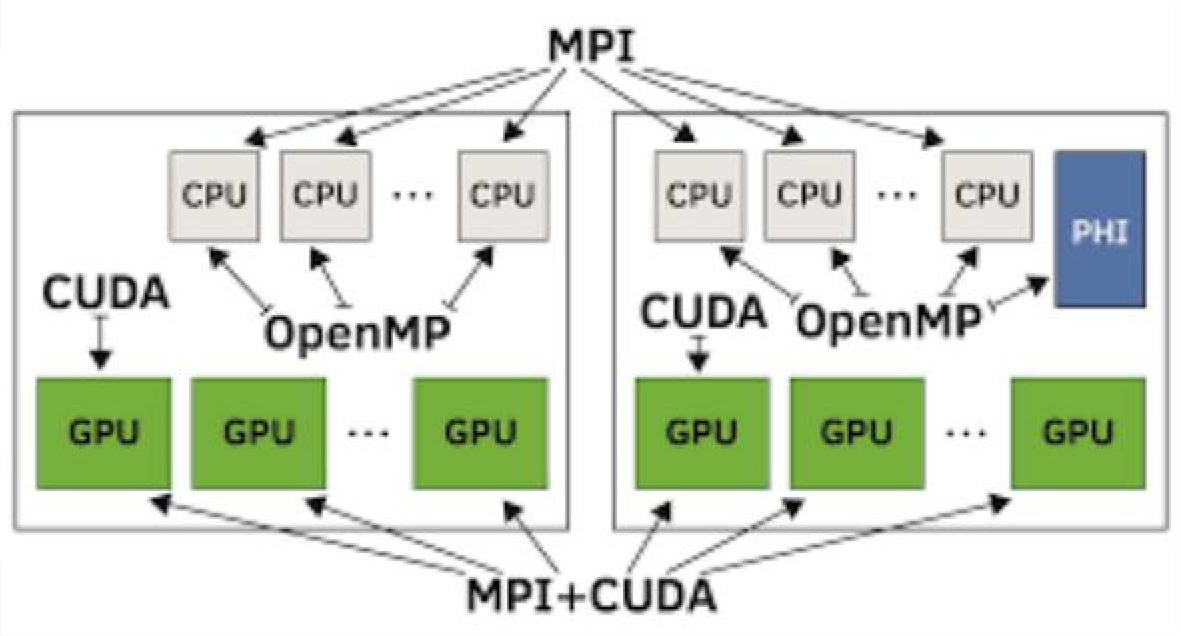


CUDA异构并行计算编程模型示例



多GPU卡、duoGPU节点：CUDA+MPI





上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

报告提纲

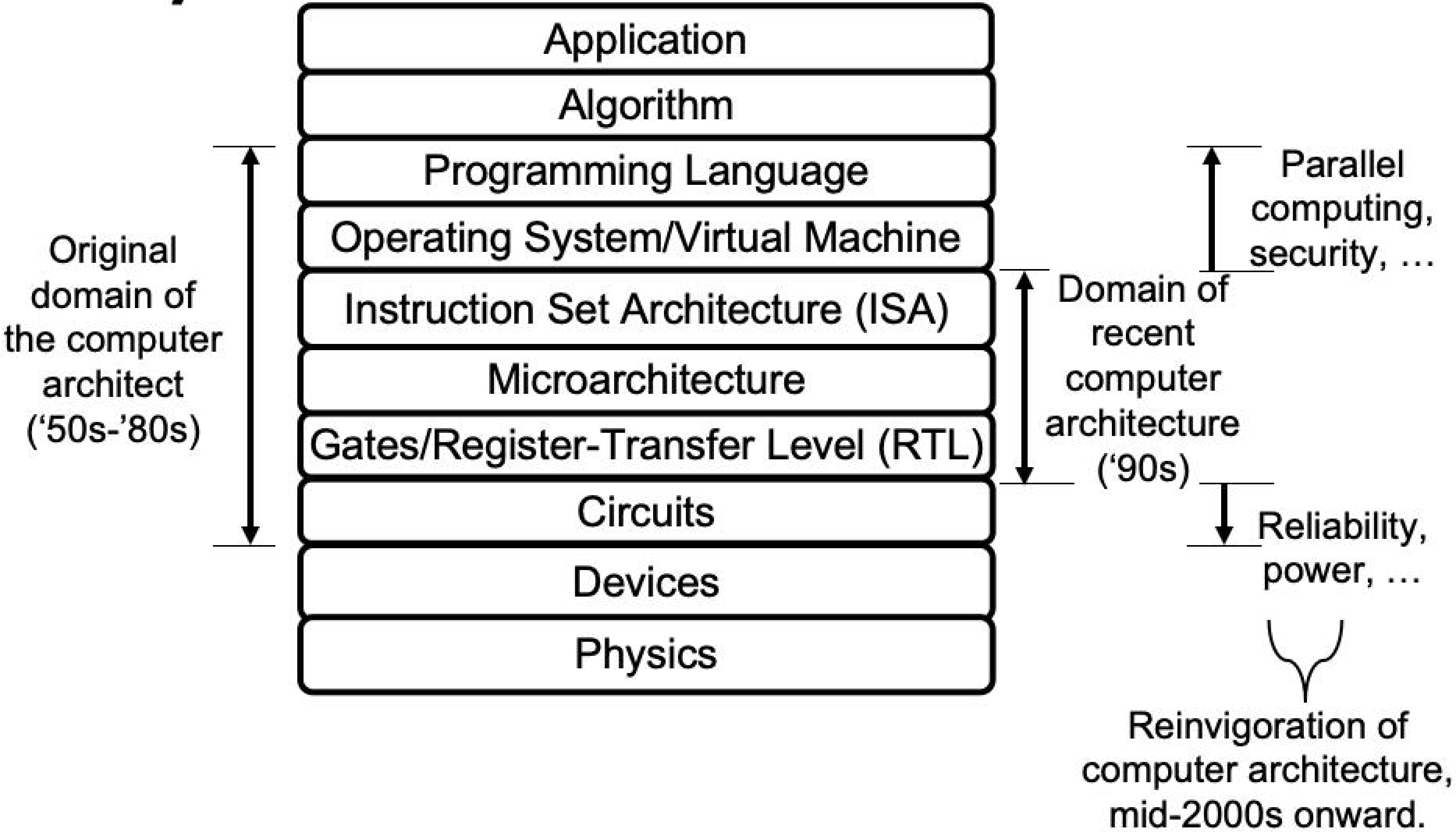
以“思源一号”为例

计算资源的基础概念

编译器基础

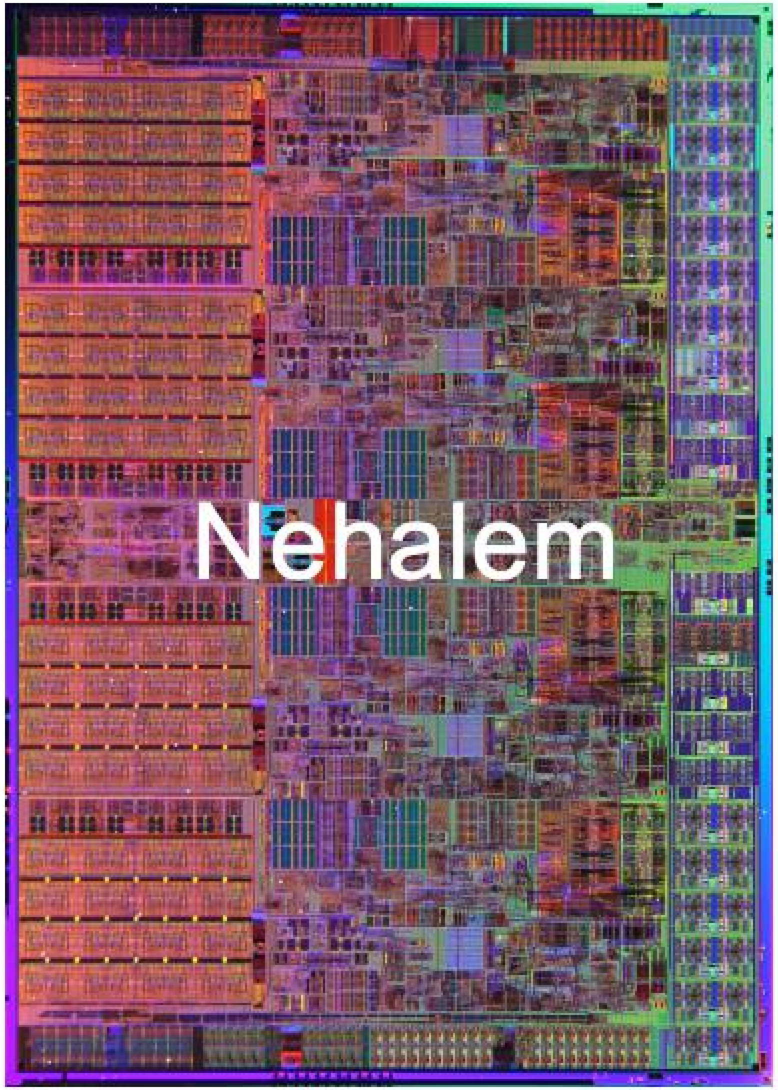
“交我算”课堂实践

Abstraction Layers in Modern Systems

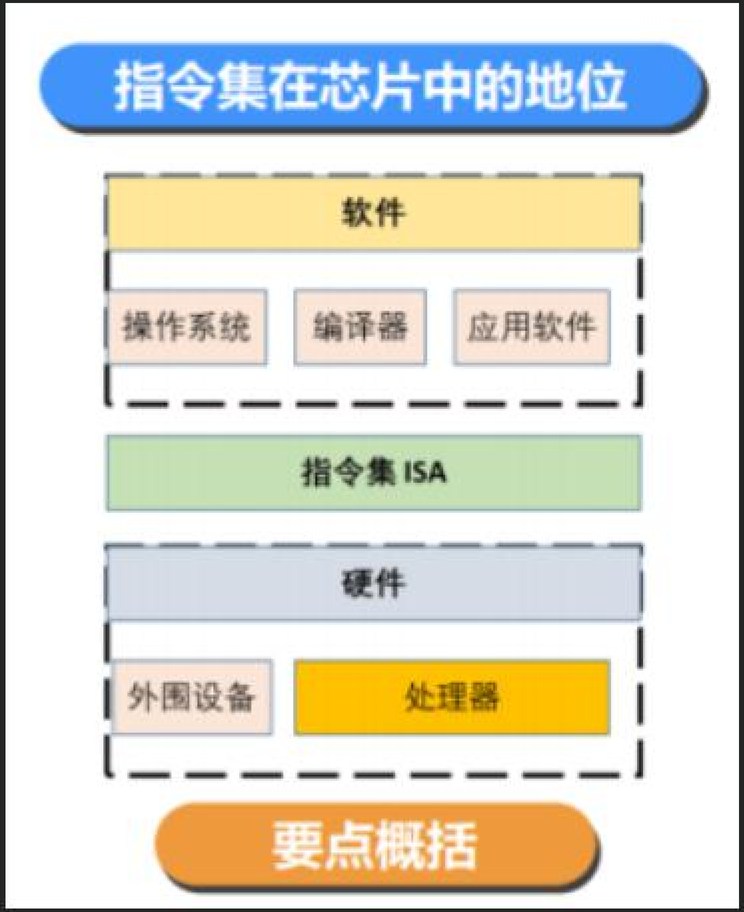


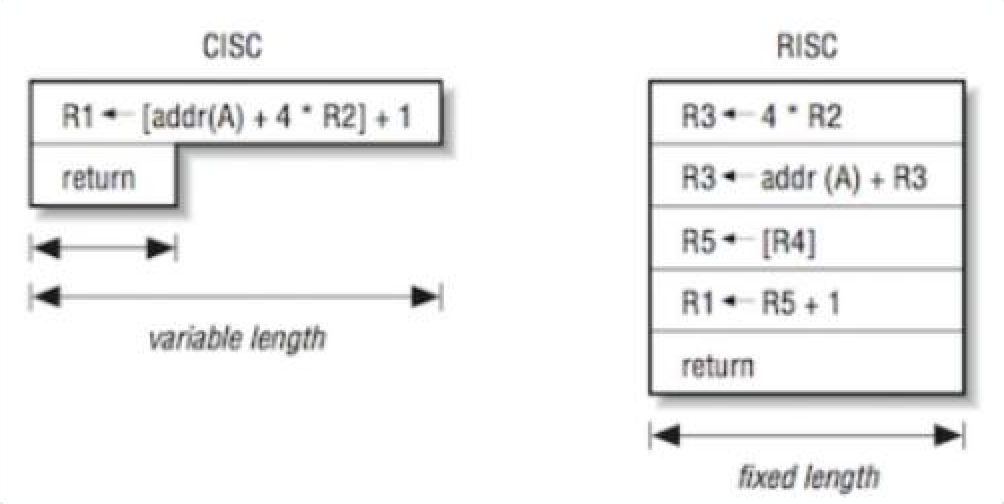
Technology constantly on the move!

- Num of transistors not limiting factor
  - Currently ~ 1 billion transistors/chip
  - Problems:
    - Too much Power, Heat, Latency
    - Not enough Parallelism
- 3-dimensional chip technology?
  - Sandwiches of silicon
  - "Through-Vias" for communication
- On-chip optical connections?
  - Power savings for large packets
- The Intel® Core™ i7 microprocessor ("Nehalem")
  - 4 cores/chip
  - 45 nm, Hafnium hi-k dielectric
  - 731M Transistors
  - Shared L3 Cache - 8MB
  - L2 Cache - 1MB (256K x 4)



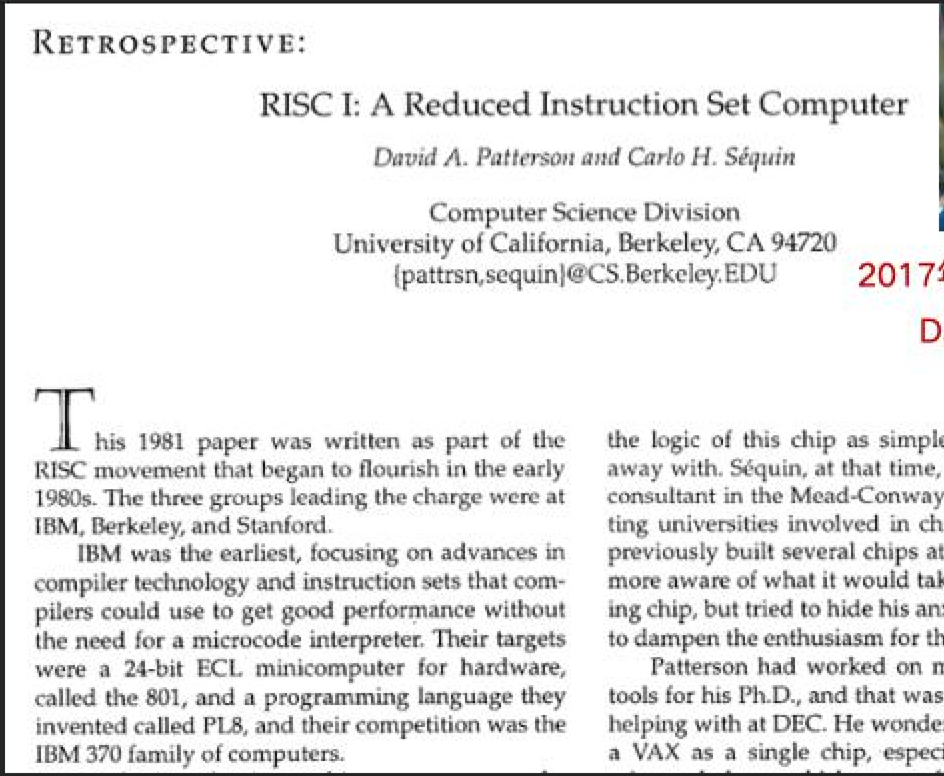
两种计算机指令集：CISC和RISC





CISC	RISC
可变长格式	定长格式
可用指令多 使用频率差别大	可用指令少 使用频率相似

40年前的一场学术争论：RISC与CISC哪个更高效



RETROSPECTIVE:

RISC I: A Reduced Instruction Set Computer

David A. Patterson and Carlo H. Séquin

Computer Science Division
University of California, Berkeley, CA 94720
{pattrsn,sequin}@CS.Berkeley.EDU

This 1981 paper was written as part of the RISC movement that began to flourish in the early 1980s. The three groups leading the charge were at IBM, Berkeley, and Stanford.

IBM was the earliest, focusing on advances in compiler technology and instruction sets that compilers could use to get good performance without the need for a microcode interpreter. Their targets were a 24-bit ECL minicomputer for hardware, called the 801, and a programming language they invented called PL8, and their competition was the IBM 370 family of computers.

the logic of this chip as simple as we could get away with. Séquin, at that time, was involved as a consultant in the Mead-Conway revolution of getting universities involved in chip design. Having previously built several chips at Bell Labs, he was more aware of what it would take to make a working chip, but tried to hide his anxieties in order not to dampen the enthusiasm for the project.

Patterson had worked on microprogramming tools for his Ph.D., and that was what he had been helping with at DEC. He wondered about building a VAX as a single chip, especially given all the

2017年图灵奖获得者

David Patterson

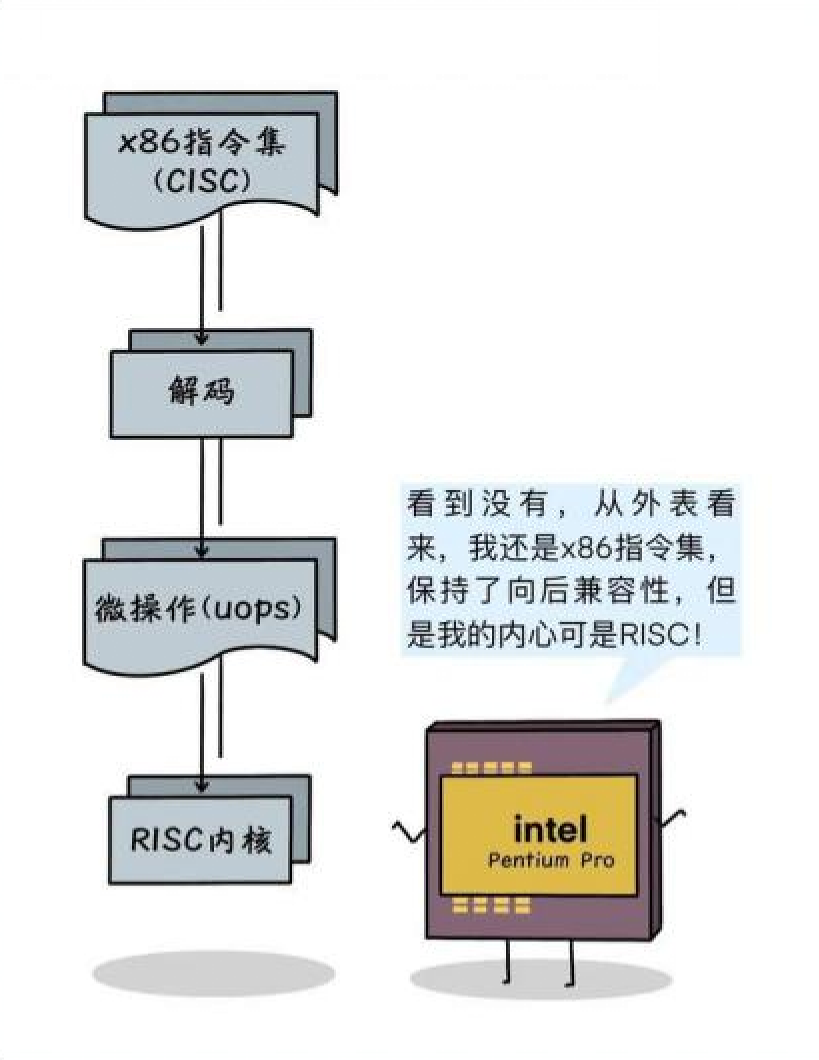
论文发表于ISCA 1981

CISC指令集日益衰落，只剩x86；而且x86也针对RISC进行了优化

上世纪70-90年代

基于CISC指令集的处理器逐渐退出历史舞台

Intel x86成为CISC的遗产



RISC指令集日益兴盛，最成功的是ARM

基于RISC的指令集阵营

MIPS TECHNOLOGIES

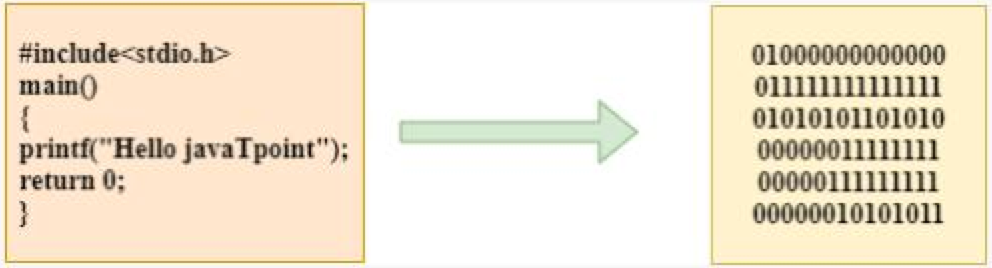
sunway 申威

PowerPC™

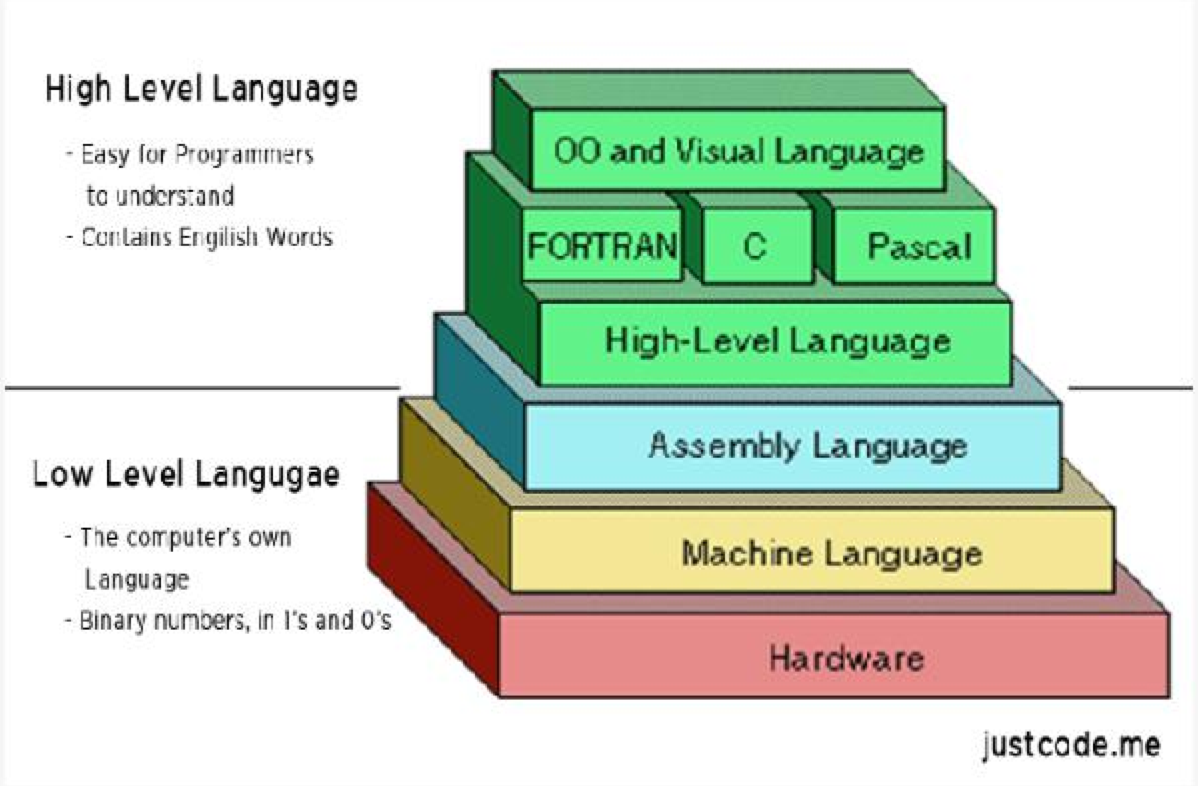
RISC-V®

为低功耗处理器而生的ARM指令集逐渐脱颖而出

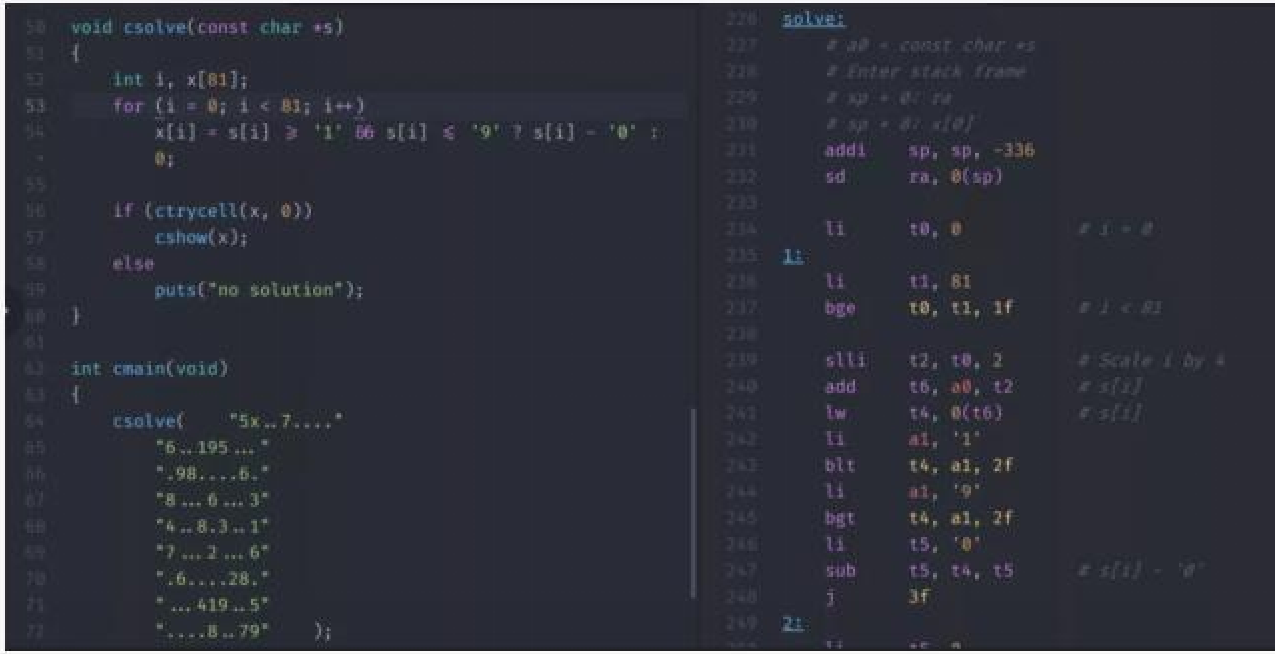
什么是编译



#include<stdio.h>
main()
{
printf("Hello javaTpoint");
return 0;
}
01000000000000
01111111111111
01010101101010
00000011111111
00000111111111
00000010101011

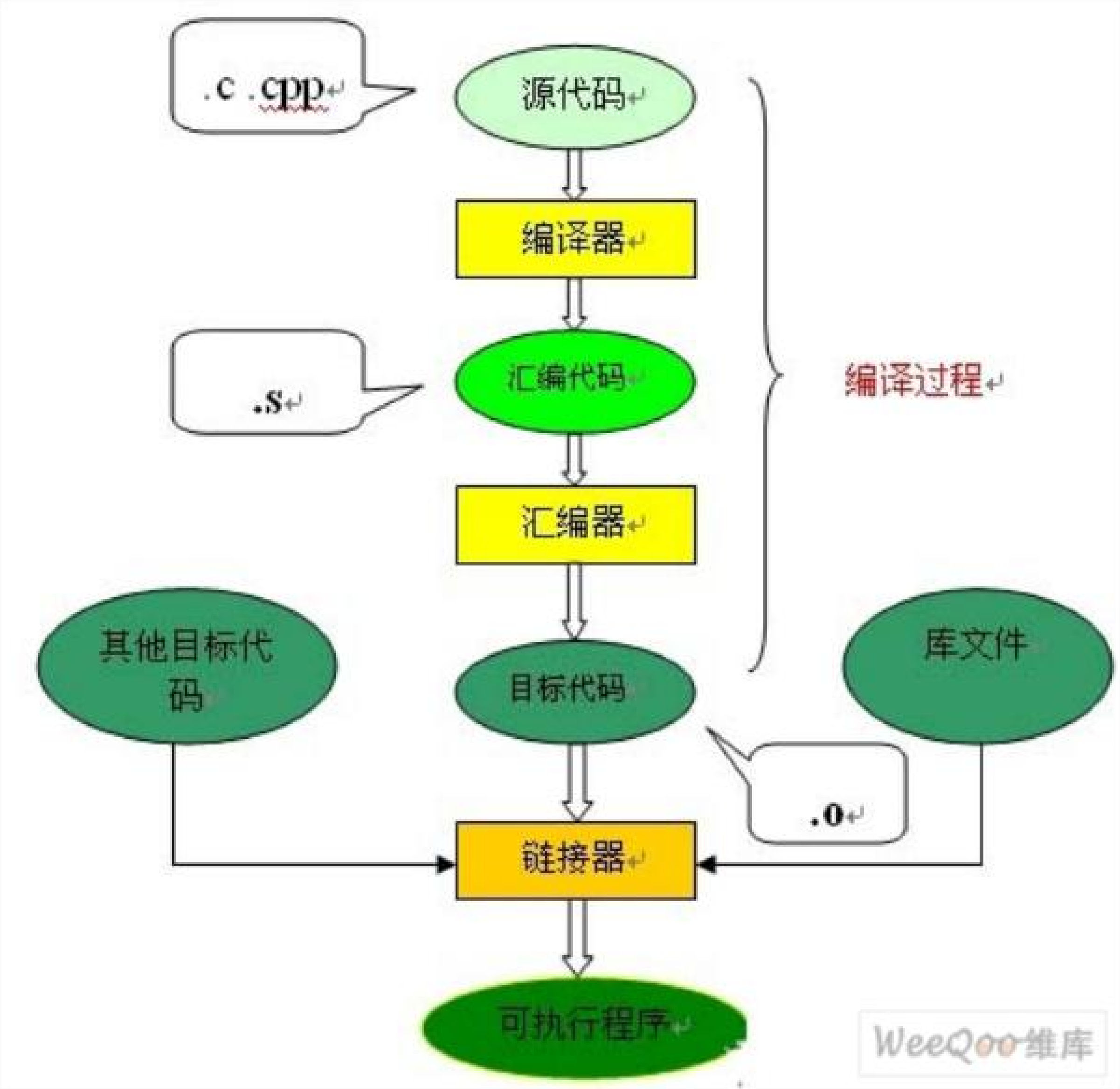


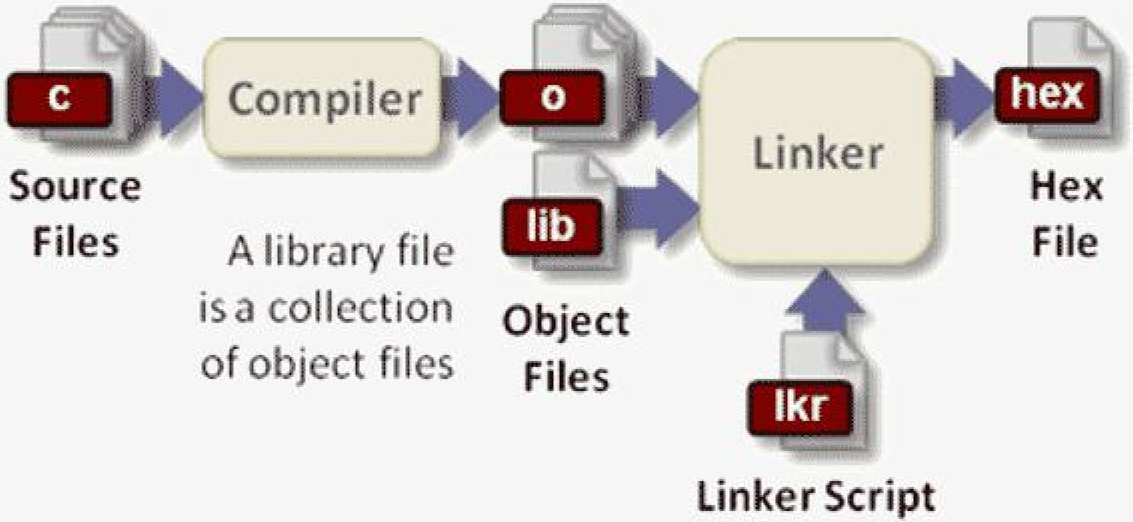
High Level Language
- Easy for Programmers to understand
- Contains Engilish Words
Low Level Langugae
- The computer's own Language
- Binary numbers, in 1's and 0's
OO and Visual Language
FORTRAN
C
Pascal
High-Level Language
Assembly Language
Machine Language
Hardware
justcode.me



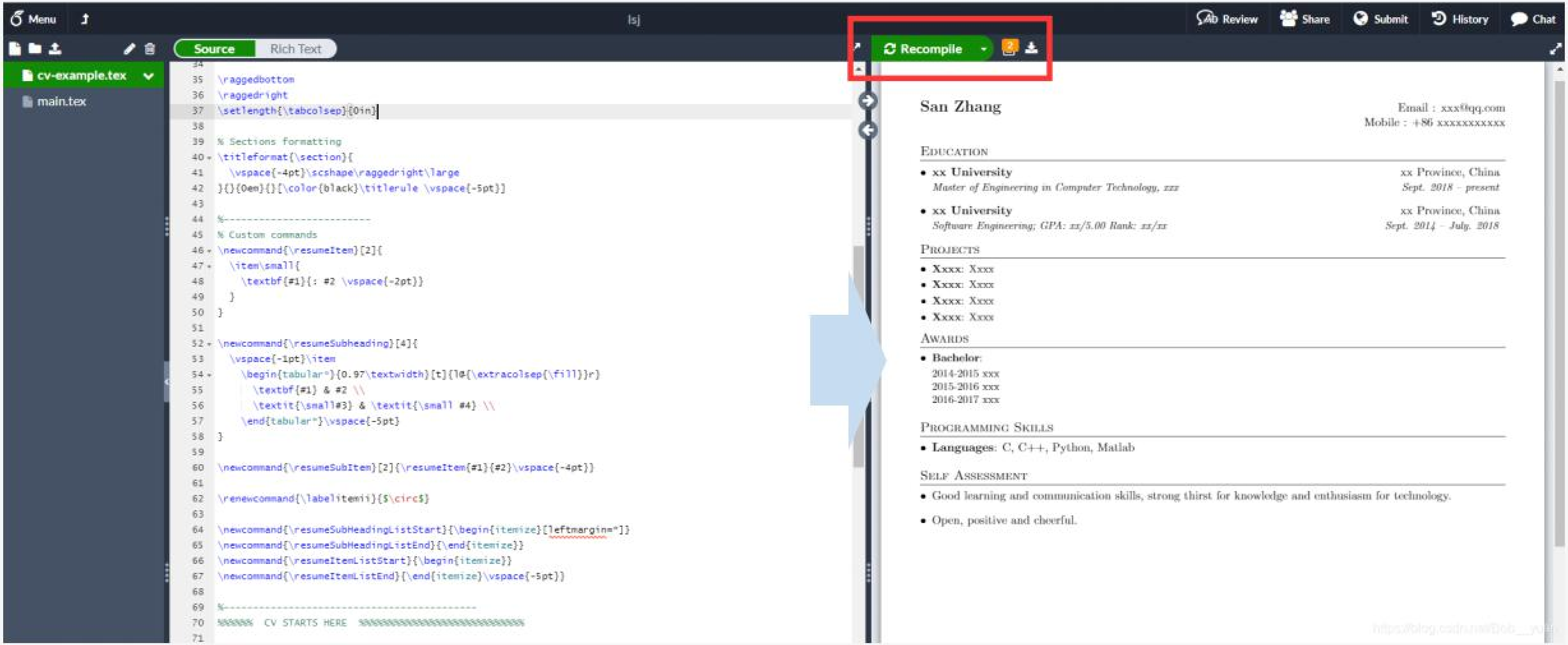
void csolve(const char *s)
{
int i, x[81];
for (i = 0; i < 81; i++)
x[i] = s[i] ≥ '1' && s[i] ≤ '9' ? s[i] - '0' :
0;
if (ctrycell(x, 0))
cshow(x);
else
puts("no solution");
}
int cmain(void)
{
csolve("5x..7...."
"6..195..."
".98....6."
"8...6...3"
"4..8.3..1"
"7...2...6"
".6....28."
"...419..5"
"....8..79");
solve:
addi sp, sp, -336
sd ra, 0(sp)
li t0, 0
1:
li t1, 81
bge t0, t1, 1f
slli t2, t0, 2
add t6, a0, t2
lw t4, 0(t6)
li a1, '1'
blt t4, a1, 2f
li a1, '9'
bgt t4, a1, 2f
li t5, '0'
sub t5, t4, t5
j 3f
2:

编译器工作流

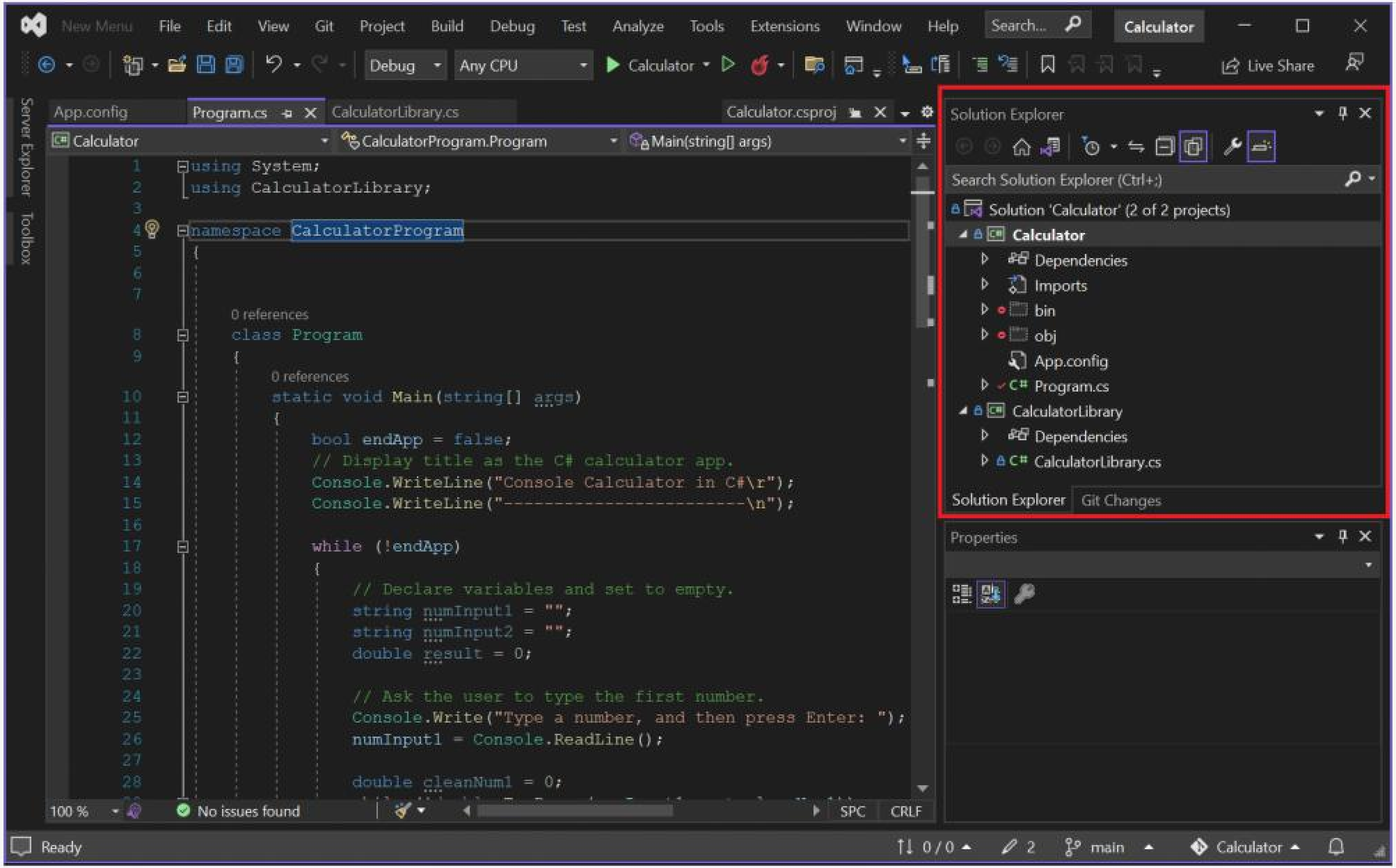

.c .cpp
源代码
编译器
.s
汇编代码
汇编器
编译过程
其他目标代码
目标代码
库文件
.o
链接器
可执行程序
WeeQoo维库


c
Source Files
Compiler
A library file is a collection of object files
o
lib
Object Files
Linker
lkr
Linker Script
hex
Hex File

Latex编译（IDE环境下）



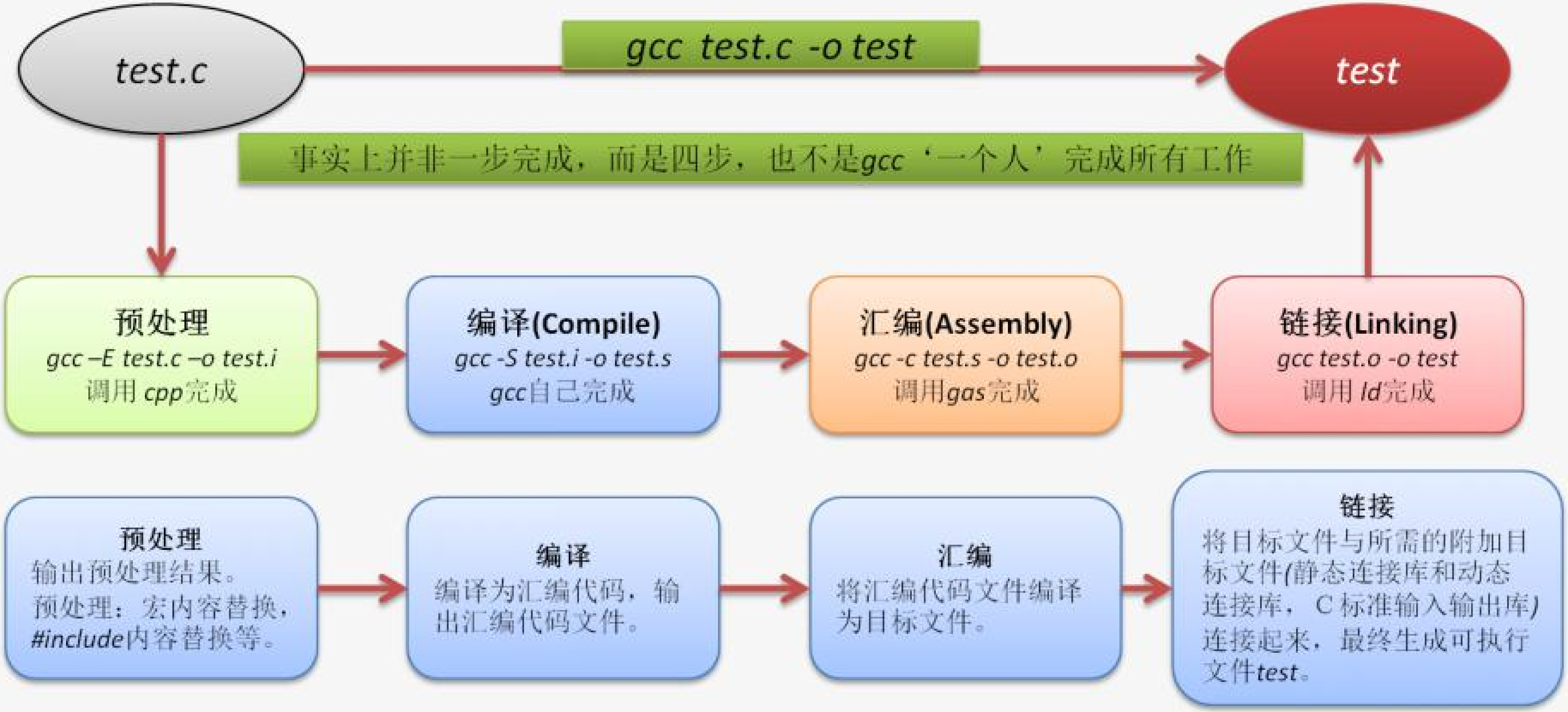
IDE (Integrated Development Environment)



Linux下的程序编译过程

GCC（GNU Compiler Collection，GNU编译器套件）

```
root@kali:~/Desktop# ls
Shubh  source.c
root@kali:~/Desktop# gcc -Wall source.c -o opt -lm
root@kali:~/Desktop# ls
opt  Shubh  source.c
root@kali:~/Desktop#
```



ICC编译器

Command Line Build Environment
Linux*, OS X*

An unique source script *compilervars.(c)sh* configures the environment for compilers, libraries and debuggers

```
> source /opt/intel/compilers_and_libraries_2016.0.109/linux/bin/compilervars.sh intel64
> icc -V
Intel(R) C Intel(R) 64 Compiler for applications running on Intel(R) 64, Version 16.0.0.109 Build 20150815
Copyright (C) 1985-2015 Intel Corporation.  All rights reserved.

> ifort -V
Intel(R) Fortran Intel(R) 64 Compiler for applications running on Intel(R) 64, Version 16.0.0.109 Build 20150815
Copyright (C) 1985-2015 Intel Corporation.  All rights reserved.
```

Running Compiler drivers **icc** (C/C++), **ifort** (Fortran)

```
> gdb-ia
No symbol table is loaded.  Use the "file" command.
GNU gdb (GDB) 7.8-16.0.458
Copyright (C) 2014 Free Software Foundation, Inc; (C) 201
License GPLv3+: GNU GPL version 3 or later <http://gnu.or
This is free software: you are free to change and redistr
There is NO WARRANTY, to the extent permitted by law.  Ty
and "show warranty" for details.
This GDB was configured as "x86_64-unknown-linux-gnu".
Type "show configuration" for configuration details.
For information about how to find Technical Support, Prod
User Forums, FAQs, tips and tricks, and other support inf
<http://www.intel.com/software/products/support/>.For hel
Type "apropos word" to search for commands related to "wo
(gdb)
```

Running Intel enhanced GDB Debugger **gdb-ia**.

```
> gdb-mic
No symbol table is loaded.  Use the "file" command.
GNU gdb (GDB) 7.8-16.0.452
Copyright (C) 2014 Free Software Foundation, Inc; (C) 2013-2015 Intel Corp.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-unknown-linux-gnu".
Type "show configuration" for configuration details.
For information about how to find Technical Support, Product Updates,
User Forums, FAQs, tips and tricks, and other support information, please visit:
<http://www.intel.com/software/products/support/>.For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb)
```

GDB Debugger with Intel enhancements for Intel® MIC architecture (**gdb-mic**) available on Linux only

ICC常用编译选项

Common Optimization Options

	Windows*	Linux*, OS X*
Disable optimization	/Od	-O0
Optimize for speed (no code size increase)	/O1	-O1
Optimize for speed (default)	/O2	-O2
High-level loop optimization	/O3	-O3
Create symbols for debugging	/Zi	-g
Multi-file inter-procedural optimization	/Qipo	-ipo
Profile guided optimization (multi-step build)	/Qprof-gen /Qprof-use	-prof-gen -prof-use
Optimize for speed across the entire program ("prototype switch") ***fast* options definitions changes over time!**	/fast same as: /O3 /Qipo /Qprec-div-, /fp:fast=2 /QxHost)	-fast same as: Linux: -ipo –O3 -no-prec-div –static –fp-model fast=2 -xHost) OS X: -ipo -mdynamic-no-pic -O3 -no-prec-div -fp-model fast=2 -xHost
OpenMP support	/Qopenmp	-qopenmp
Automatic parallelization	/Qparallel	-parallel

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

报告提纲

以“思源一号”为例

计算资源的基础概念

编译器基础

“交我算”课堂实践