A bunch of HPC

Yiwei Yang, Feiran Qin
GeekPie_HPC

上海科技大学
ShanghaiTech University

Geek
Pie_

2021/4/9

Table of Contents

① Why HPC?

② How HPC?

- ① Parallel Computing
- ② DevOps
- ③ GPU & CPU AI
- ④ Distributed Operating Systems

③ Why we need you?

- ① Class recomendations

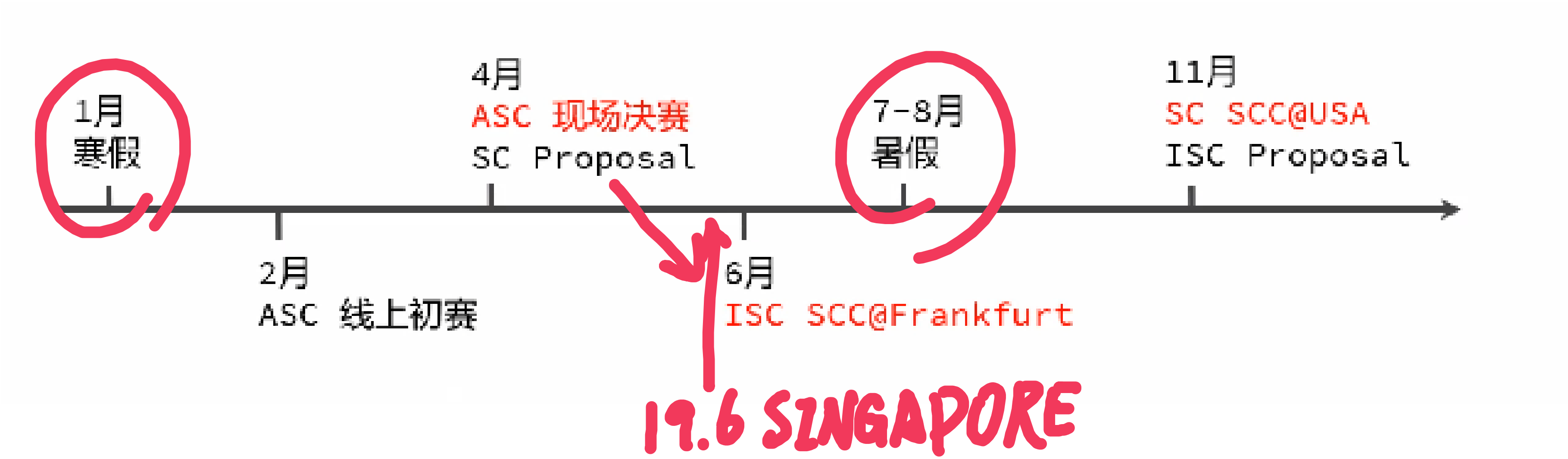
The Goal of this talk

① Let more people know HPC.

② Recommend some quality classes in school like PL/OS/Network/CA/Parallel computing to strengthen yourself and hope you become a better member of the HPC.

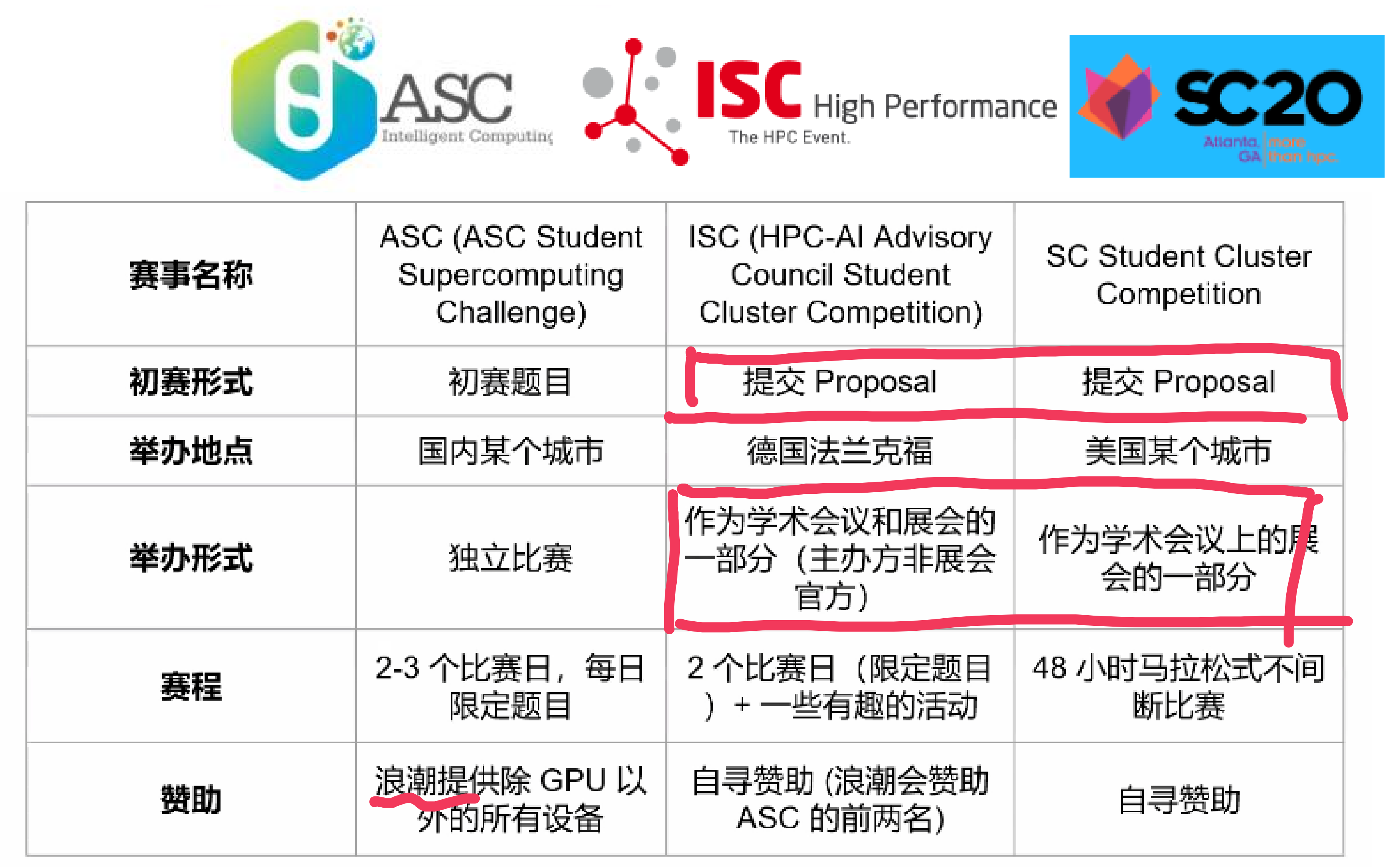
③ Let you join our team first and you can watch the whole process from the sidelines or as an reverse team member.

HPC Basics



Pic Credit: Tsinghua HPC Team

HPC Basics


ASC
Intelligent Computing
ISC High Performance
The HPC Event.
SC20
Atlanta, GA | more than hpc.

赛事名称	ASC (ASC Student Supercomputing Challenge)	ISC (HPC-AI Advisory Council Student Cluster Competition)	SC Student Cluster Competition
初赛形式	初赛题目	提交 Proposal	提交 Proposal
举办地点	国内某个城市	德国法兰克福	美国某个城市
举办形式	独立比赛	作为学术会议和展会的一部分（主办方非展会官方）	作为学术会议上的展会的一部分
赛程	2-3 个比赛日，每日限定题目	2 个比赛日（限定题目）+ 一些有趣的活动	48 小时马拉松式不间断比赛
赞助	浪潮提供除 GPU 以外的所有设备	自寻赞助 (浪潮会赞助 ASC 的前两名)	自寻赞助

HPC Rules

① 上场队员为 5-6 名本科生，比赛时其他人不得操作集群

② 正式开始前需确定集群硬件配置，非意外/特殊规则允许不得重启或更改配置

③ 任何时间集群功耗不得超过 3kw 4k5w否则会被惩罚（罚分、挂程序等）

④ 不允许使用或搭建外部网络 !!! (违反将取消资格，亦可能遭到 Euro100000 罚款)

HPC Rules（Cond）

① SC 专属

  ① 比赛中途会有至少 1 次组委会设计的断电，队员需在断电后 启动集群恢复工作

  ② 官方提供若干云服务器资源，需要合理利用

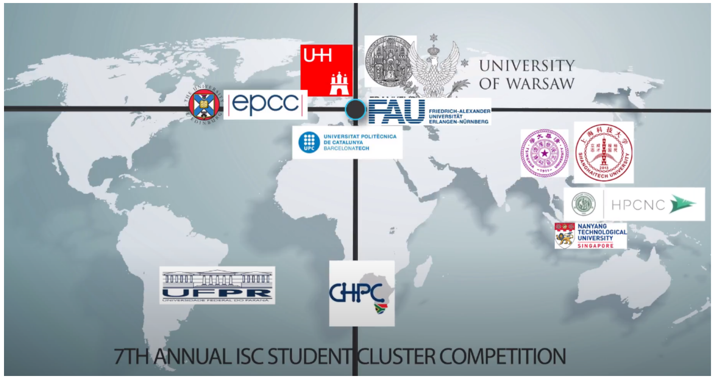
  ③ 参与学术会议，听若干讲座（SC19 改为由导师参加讲座）

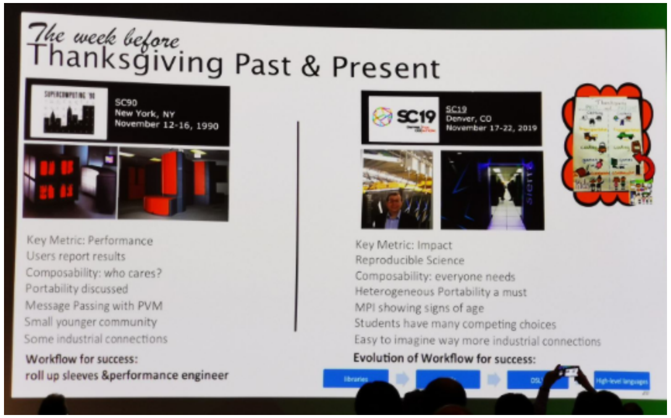
② ISC 专属

  ① 。开始前有 Tshirt challenge 环节，需要在会场找齐本队 队服方能开始比赛

  ② 。展位装饰美观程度作为评分标准之一

  ③ 。所有参展人员可以投票选出最喜爱的队伍


UH
epcc
FAU
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
UNIVERSITY OF WARSAW
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH
HPCNC
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
UFPR
CHPC
7TH ANNUAL ISC STUDENT CLUSTER COMPETITION



The week before
Thanksgiving Past & Present

SC90
New York, NY
November 12-16, 1990

Key Metric: Performance
Users report results
Composability: who cares?
Portability discussed
Message Passing with PVM
Small younger community
Some industrial connections
Workflow for success:
roll up sleeves &performance engineer

SC19
Denver, CO
November 17-22, 2019

Key Metric: Impact
Reproducible Science
Composability: everyone needs
Heterogeneous Portability a must
MPI showing signs of age
Students have many competing choices
Easy to imagine way more industrial connections
Evolution of Workflow for success:

HPC Rules（Cond）

① SC 2020 专属

- ① 使用Azure 云计算资源，有4500USD 的预算购买相关集群，选择储存、计算节点。
- ② 超功耗的惊喜改为突然增加 500USD的预算。
- ③ 考察脚本的能力，尤其是启动脚本。24小时连轴转的能力，slurm的活用。
- ④ IO500 MadFS burst cache with tokio ucx

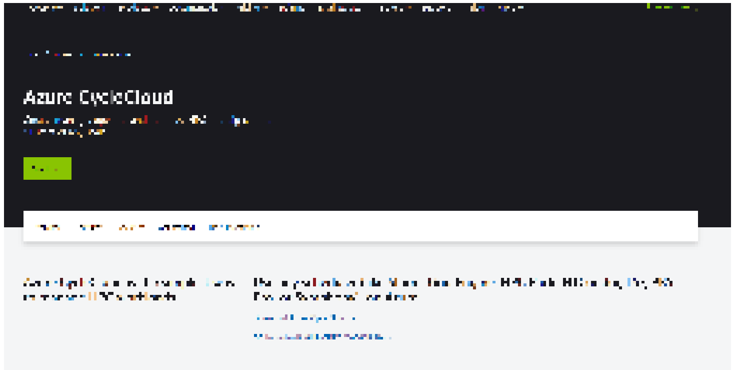
② ISC 2021 专属

- ① MPI Profiler hook on alltoallv of WRF(另外一个天气应用）
- ② ucberkeley 开源的类openmpi 框架。

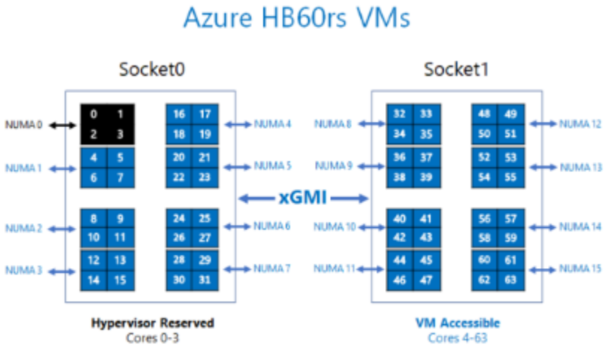
MadFS

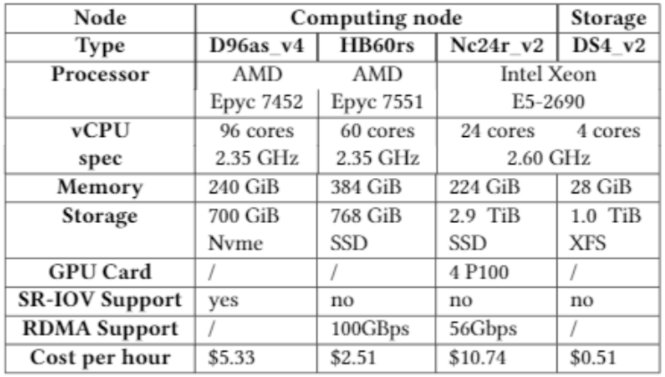
即将开源，欢迎关注
https://github.com/madsys-dev/MadFS

TODO:
- 实用化，产品化
- 优化方向
  - 数据：io_uring（绕开内核）
  - 元数据：NVM（快速持久化）



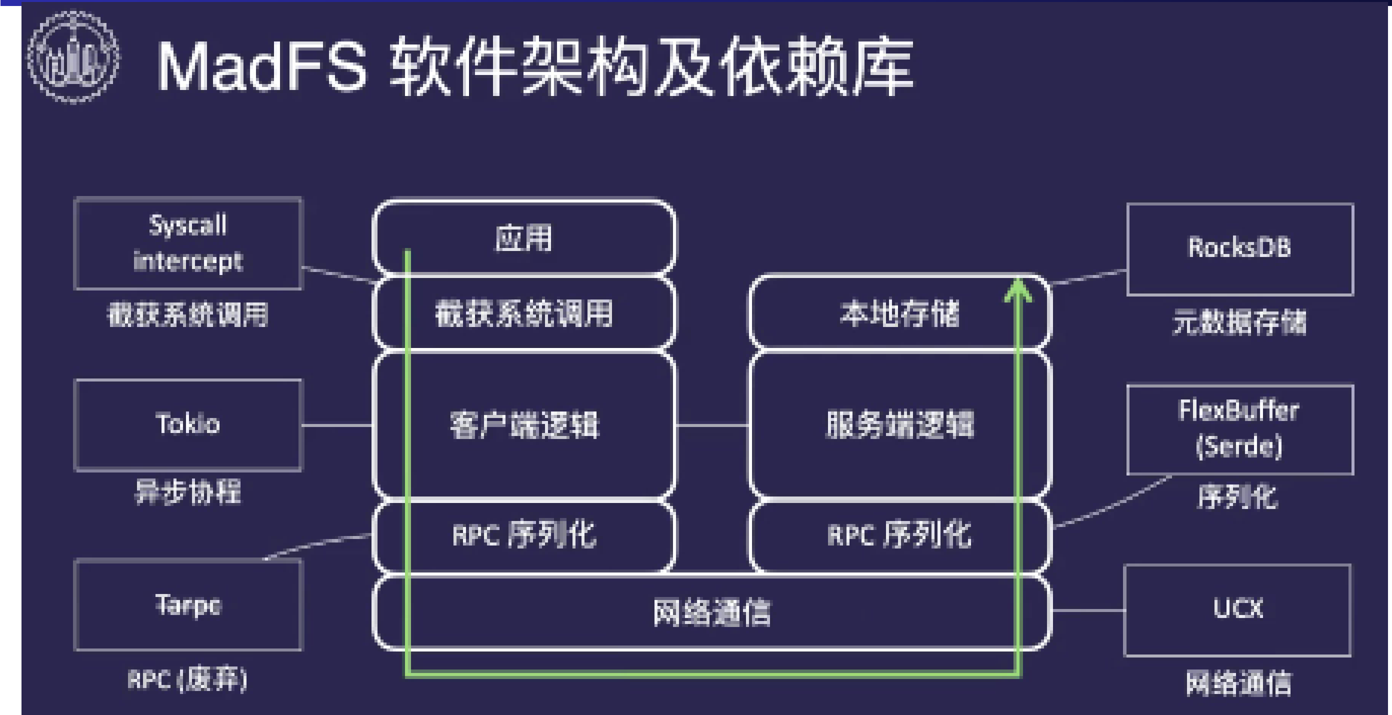
Azure CycleCloud





Node	Computing node			Storage
Type	D96as_v4	HB60rs	Nc24r_v2	DS4_v2
Processor	AMD Epyc 7452	AMD Epyc 7551	Intel Xeon E5-2690	
vCPU spec	96 cores 2.35 GHz	60 cores 2.35 GHz	24 cores 2.60 GHz	4 cores
Memory	240 GiB	384 GiB	224 GiB	28 GiB
Storage	700 GiB Nvme	768 GiB SSD	2.9 TiB SSD	1.0 TiB XFS
GPU Card	/	/	4 P100	/
SR-IOV Support	yes	no	no	no
RDMA Support	/	100GBps	56Gbps	/
Cost per hour	$5.33	$2.51	$10.74	$0.51

HPC Rules（Cond）



MadFS 软件架构及依赖库

Syscall intercept
截获系统调用

Tokio
异步协程

Tarpc
RPC (废弃)

应用
截获系统调用
客户端逻辑
RPC 序列化

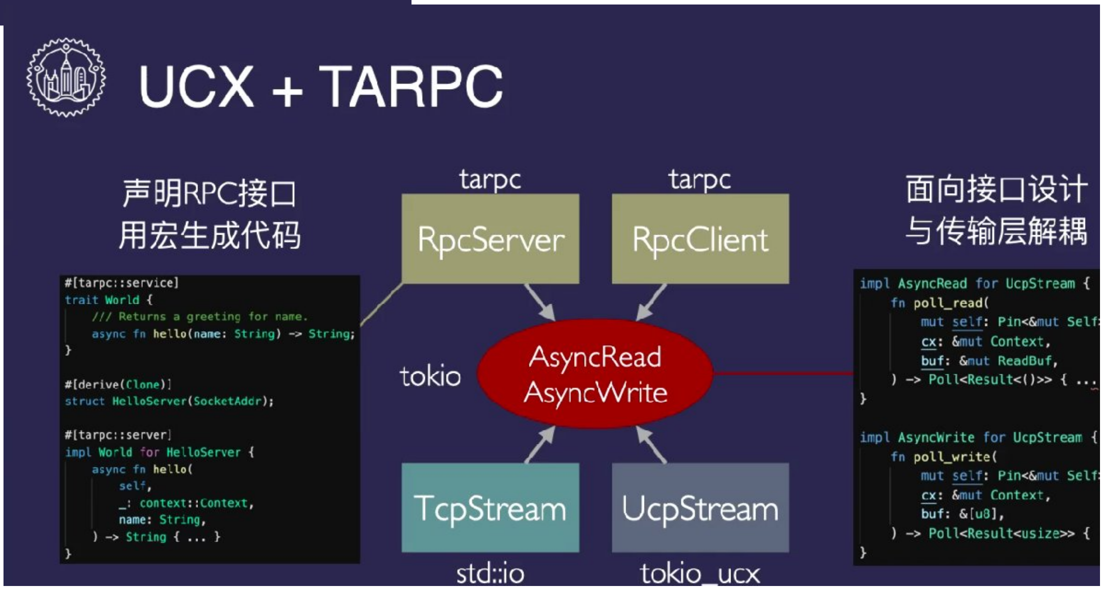
本地存储
服务端逻辑
RPC 序列化

网络通信

RocksDB
元数据存储

FlexBuffer (Serde)
序列化

UCX
网络通信



UCX + TARPC

声明RPC接口
用宏生成代码

```
#[tarpc::service]
trait World {
    /// Returns a greeting for name.
    async fn hello(name: String) -> String;
}

#[derive(Clone)]
struct HelloServer(SocketAddr);

#[tarpc::server]
impl World for HelloServer {
    async fn hello(
        self,
        _: context::Context,
        name: String,
    ) -> String { ... }
}
```

tarpc
RpcServer

tarpc
RpcClient

tokio
AsyncRead
AsyncWrite

TcpStream
std::io

UcpStream
tokio_ucx

面向接口设计
与传输层解耦

```
impl AsyncRead for UcpStream {
    fn poll_read(
        mut self: Pin<&mut Self
        cx: &mut Context,
        buf: &mut ReadBuf,
    ) -> Poll<Result<()>> { ...
}

impl AsyncWrite for UcpStream {
    fn poll_write(
        mut self: Pin<&mut Self
        cx: &mut Context,
        buf: &[u8],
    ) -> Poll<Result<usize>> {
}
```

HPC Rank

① 题目构成与评分规则

Matrix mul　　challenge overall

- ① Benchmark 基准测试程序：HPL，HPCG，HPCC
- ② 每场比赛每年规则和内容几乎相同　Conjagated gradient
- ③ 在比赛正式开始前进行（意思是：超功耗不扣分，早上温度低/换cpu/超频）
- ④ 可以换配置，但是最终成绩需基于最终配置
- ⑤ 公开题目
  - ① 赛前 3-6 个月公布大致内，比赛开始什下发具体任务
  - ② 通常由正确性分数 + 性能分数构成　半精度
  - ③ 需要进行全面细致的优化方能获胜
  - ④ 通常由正确性分数 +性能分数构成（编译）
- ⑥ 神秘应用（Mystery Application）
  - ① 在比赛开始前一无所知，所有内容在比赛开始时下发
  - ② 拼手速 / 正确的硬件配置 / 运气
  - ③ spack / apt / pip / npm （快速开编译选项）

HPC Rank（Cond）

① 物理学

- ① SWIFTsim（ISC'19）：宇宙学模拟 (天体相互作用等)
- ② ShengBTE（ASC'19）：声子 Bolzmann 输运方程求解

② 生命科学

- ① WTDBG2（ASC'19）：基因序列片段拼接

③ 地球科学

- ① SeisSol (SC'18 Reproducibility)：地震模拟 (印尼海啸)
- ② NormalModes（SC'19 Reproducibility）：行星简正模式计算 (以月球为例)

④ 后候与气象学

- ① WRF (Weather Research and Forecasting model) (SC'18): 天气预报 (大气动力学)
- ② CESM (Community Earth System Model) (ASC'19, SC'20)：复杂气候模型

⑤ 计算机科学

- ① SST (Structural Simulation Toolkit) (SC'19)：计算机体系结构模拟
- ② QuEST (ASC'20)：量子电路模拟

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} + S = 0,$$

$$Q = \left(\rho', \rho u, \rho v, \rho w, (\rho e_T)', (\rho q)'\right)^{\mathrm{T}},$$

$$F = \left(\rho u, \rho u u + p', \rho u v, \rho u w, (\rho e_T + p)\, u, \rho u q\right)^{\mathrm{T}},$$

$$G = \left(\rho v, \rho v u, \rho v v + p', \rho v w, (\rho e_T + p)\, v, \rho v q\right)^{\mathrm{T}},$$

$$H = \left(\rho w, \rho w u, \rho w v, \rho w w + p', (\rho e_T + p)\, w, \rho w q\right)^{\mathrm{T}},$$

$$S = \left(0, \partial \bar{p}/\partial x - f \rho v, \partial \bar{p}/\partial y + f \rho u, \rho' g, 0, 0\right)^{\mathrm{T}},$$

HPC Rank（Cond）

① 炼丹（人工智能）

- ① C V
  - ① VGG over ImageNet (ISC'18)
  - ② ResNet over ImageNet (SC 18)
  - ③ DeepLab / Tiramisu in hurricane recognition (ISC'19)
  - ④ Facial Super-Resolution (ASC' 19)
- ② NLP
  - ① BERT / Transformer (Cloze Test) (ASC' 20)

    CET4完型填空
  - ② Pretraining BERT (ISC'20)
  - ③ M$ MARCO (ASC'18)

youtube shanghaitech
第二向的视频

刘建中学长在看
着你笑

HPC Rank（Cond）

水
Random

① ASC 答辩
  - ① 正式比赛完成后第二天进行
  - ② 先用 10 分钟时间演讲，再回答评委问题
  - ③ 各队单独进行，不能旁听其他队答辩亲

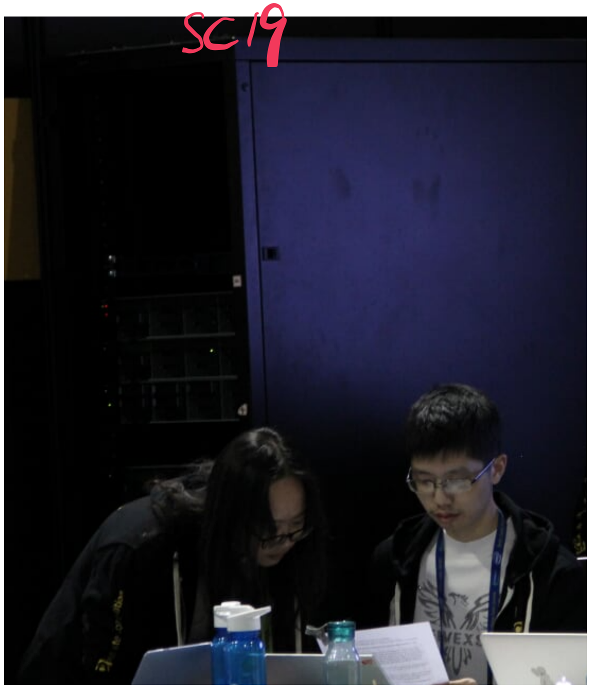
② ISC 面试
  - ① 在最后一个比赛日进行
  - ② 评委走到展位前与各队进行交流，内容宽泛 (diversity)
  - ③ SC 面试和 Poster

soft skills　女生 省份
~~黑人~~

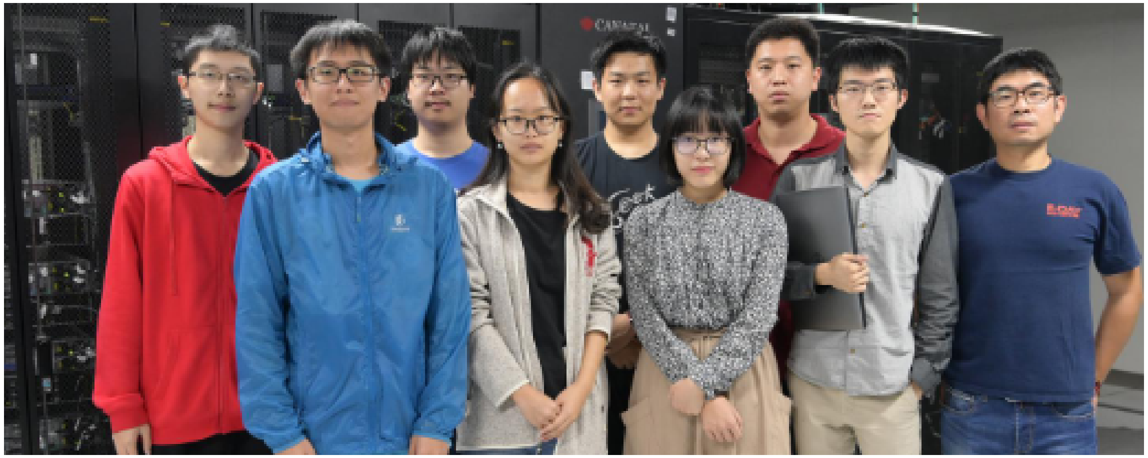
③ SC面试与 ISC 类似，但评委手里有详细的打分表
  - ① 每道题目由专门的评委进行专业面试，外加综合面试
  - ② Poster 类似学术会议上的 Poster 展示，也有评委

ASC 18

SC 19



HPC Friends（Cond）

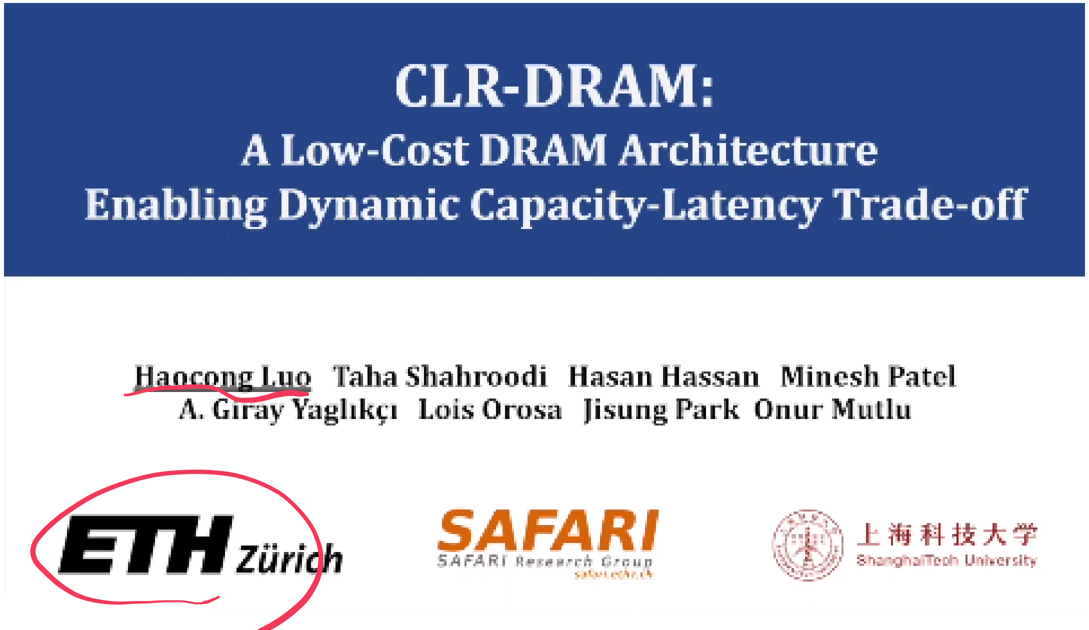


HPC Friends（Cond）

Ziji Shi(史子骥)

Nanyang Technological University, Singapore

AppleML/商汤



CLR-DRAM:

A Low-Cost DRAM Architecture

Enabling Dynamic Capacity-Latency Trade-off

Haocong Luo Taha Shahroodi Hasan Hassan Minesh Patel
A. Giray Yaglıkçı Lois Orosa Jisung Park Onur Mutlu

ETH Zürich SAFARI SAFARI Research Group safari.ethz.ch 上海科技大学 ShanghaiTech University

NVIDIA 港中深

Chenhao Wu (Vito)

General Thoughs

serilized → parallel
Math

{tvm
opencl
hipcc
cuda
CS

bit stream
板载石北网卡
没有内存overhead
细计算.
EE

Cache {unroll
blocking {temporal
space

{constant folding
Dead code
-O3

~~context switch~~
icc — {avx
unroll
-mavx512
ZDA

Algorithm > CPU to GPU&FPGA > CA Fine tune > Compiler Option

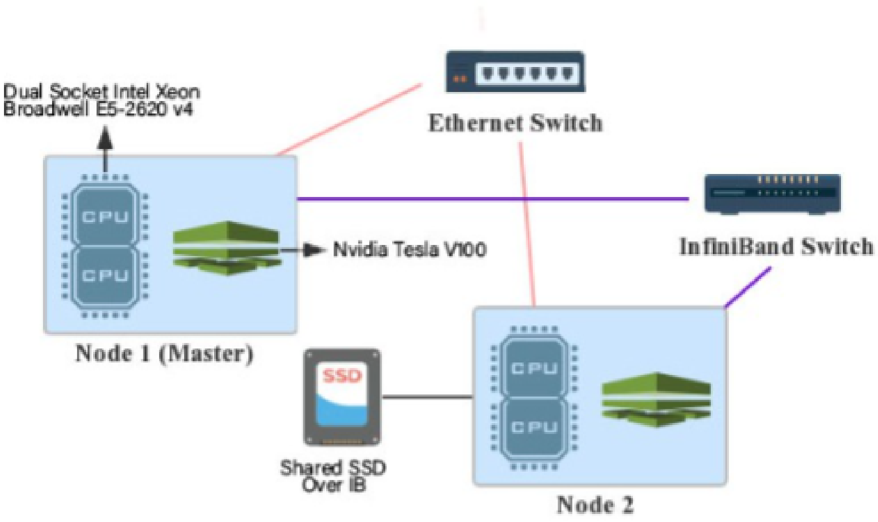
Time-consuming < Medium < Fast

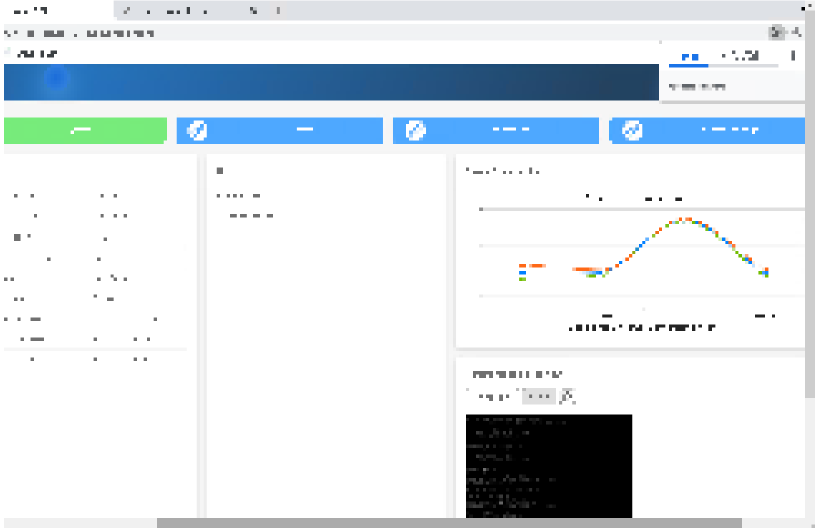
1 month ~ 3 months
1 day ~ 3 weeks
1 hour ~ 20 hours

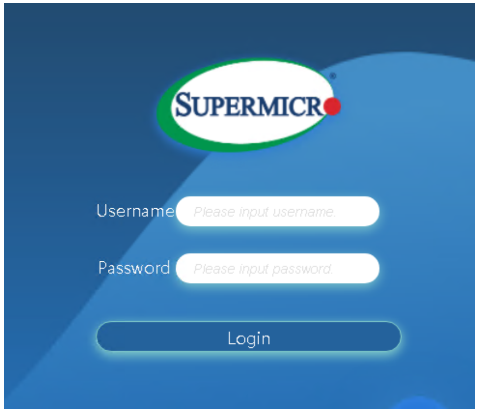
DevOps

HPL binary from Nvidia

Our HPC Composition



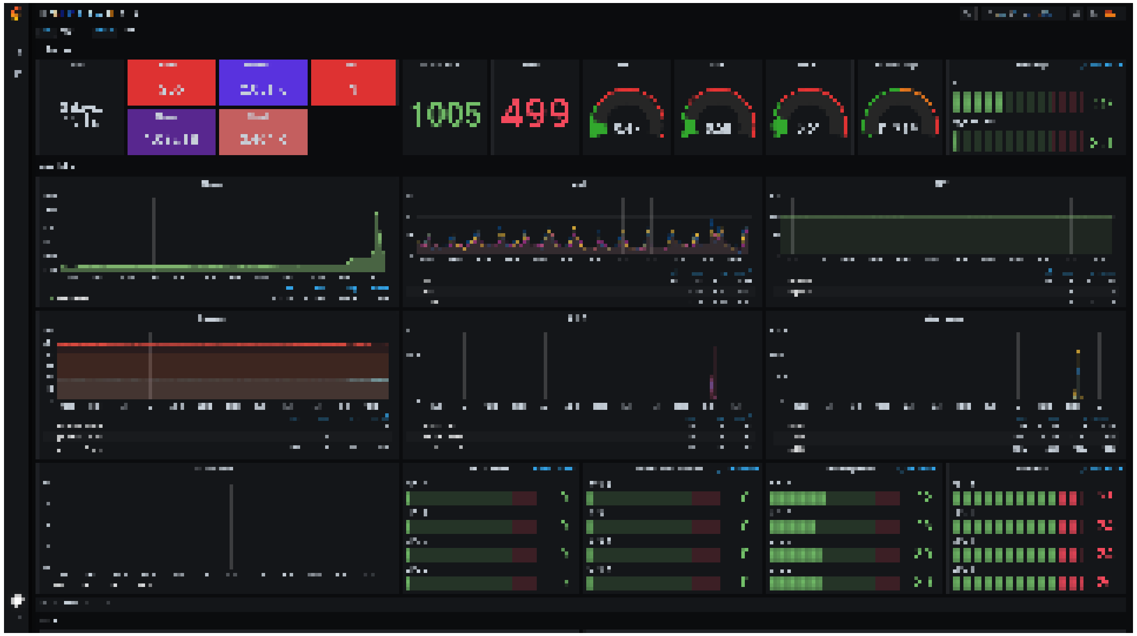




Username Please input username

Password Please input password

Login



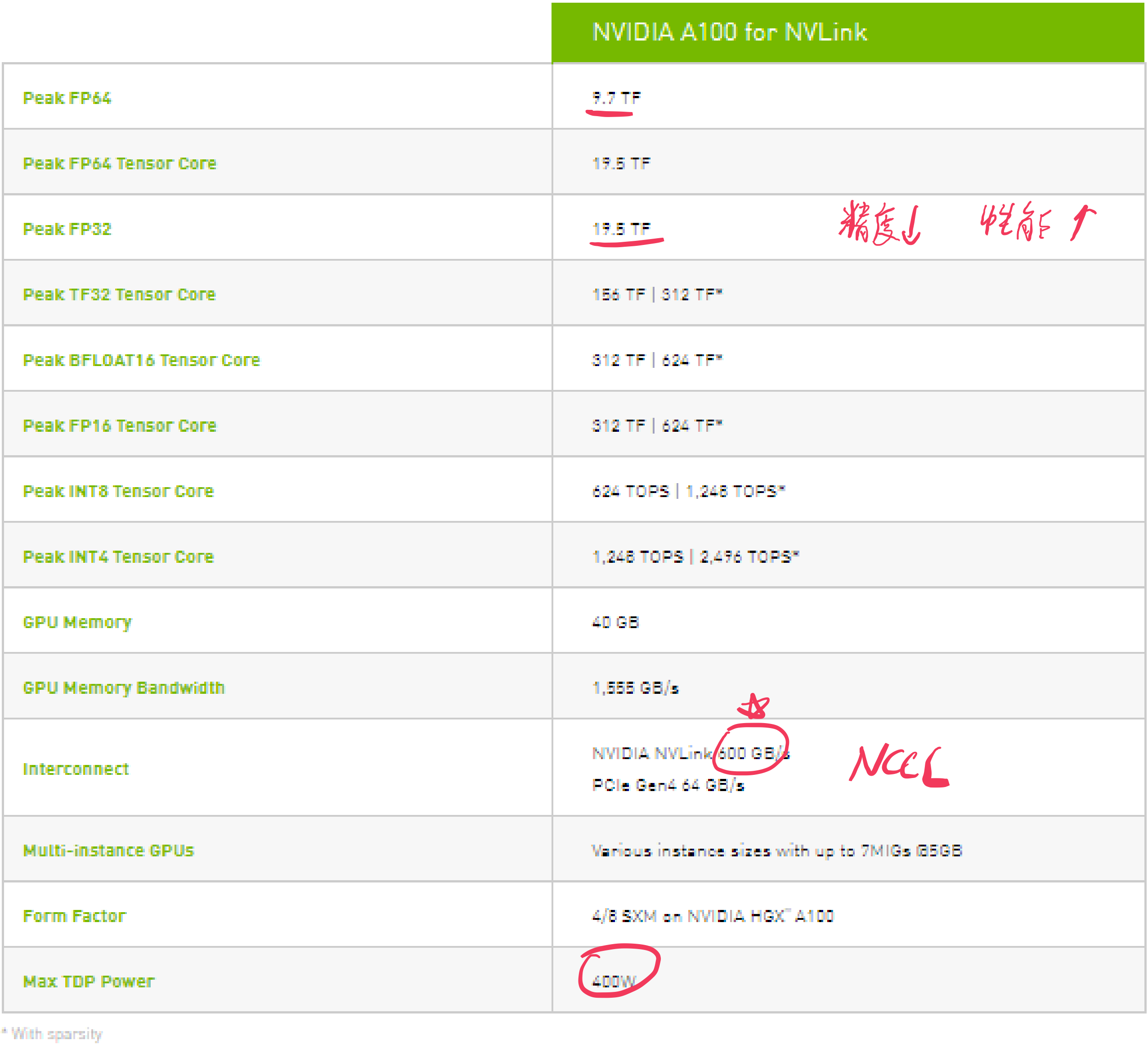
Our HPC Composition

numa

- 计算
  - CPU：双路 Intel / ~~AMD~~ software 中高级服务器处理器，用于所有计算用途
  - GPU：NVIDIA V106, 用于大规模并行浮点计算 (包括 benchmark) A100
- 存储设备
  - RAM：DDR4 高频率 (>2933 Mhz) ECC RDIMM 效验
  - SAS/SATA SSD：用于系统安装、日常文件存储
  - NVMe SSD：高性能、高功耗, 用于高吞吐量程序、IO benchmark 等
  - NVRam：外存？内存？
- 通信设备
  - Ethernet NIC：低功耗、稳定，用于管理 InfiniBand(200Gb/s ucx open sm
  - (IB) NIC：高带宽、低延迟，用于应用通信

no hdmi

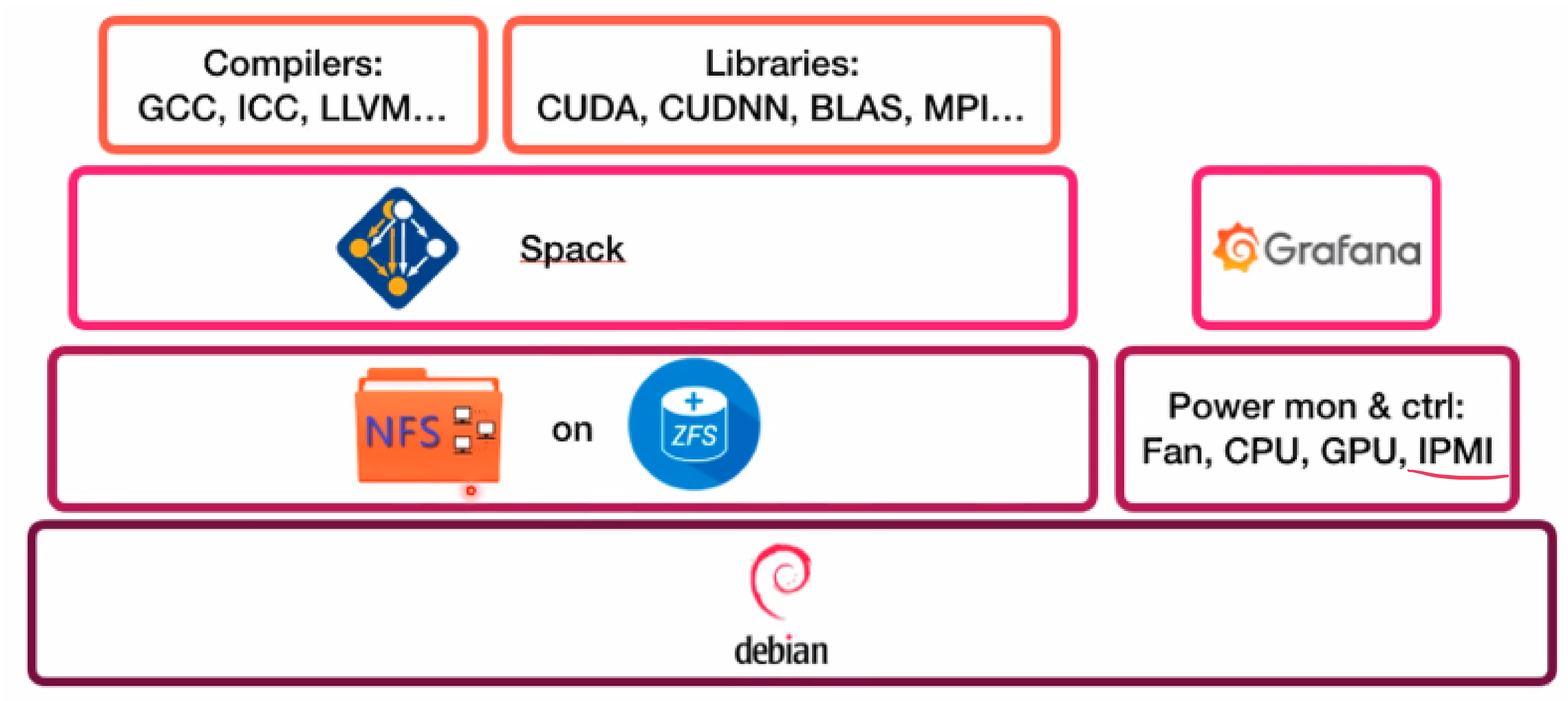
Our HPC Composition



	NVIDIA A100 for NVLink
Peak FP64	9.7 TF
Peak FP64 Tensor Core	19.5 TF
Peak FP32	19.5 TF 精度↓ 性能↑
Peak TF32 Tensor Core	156 TF | 312 TF*
Peak BFLOAT16 Tensor Core	312 TF | 624 TF*
Peak FP16 Tensor Core	312 TF | 624 TF*
Peak INT8 Tensor Core	624 TOPS | 1,248 TOPS*
Peak INT4 Tensor Core	1,248 TOPS | 2,496 TOPS*
GPU Memory	40 GB
GPU Memory Bandwidth	1,555 GB/s
Interconnect	NVIDIA NVLink 600 GB/s PCIe Gen4 64 GB/s NCCL
Multi-instance GPUs	Various instance sizes with up to 7MIGs @5GB
Form Factor	4/8 SXM on NVIDIA HGX™ A100
Max TDP Power	400W

* With sparsity

HPC DevOps Stack



Compilers:
GCC, ICC, LLVM...

Libraries:
CUDA, CUDNN, BLAS, MPI...

Spack

Grafana

NFS on ZFS

Power mon & ctrl:
Fan, CPU, GPU, IPMI

debian

HPC DevOps Stack

系统与软件管理

- ① CentOS Linux 操作系统，使用 SSH 连接集群 。使用 clustershell 进行统一控制
- ② 通常需要各种各样的工具和库
    - ① 编译器: GCC / ICC / Clang / PGI
    - ② MPI: OpenMPI / Intel MPI / Mellanox HPC-X
    - ③ 通信方式: Ethernet / IPoIB / UCX
    - ④ 数学库：CuBLAS / MKL / OpenBLAS, FFTW / CuFFTW
    - ⑤ 编译选项：是否启用 AVX512 指令集 / 是否开启 o3 优化 使用 Spack 统一管理各种软件的不同版本
- ① 动管理软件包依赖
    - ① Spack一键配置 / 清理所需环境

HPC DevOps Stack

① 体力活
  ① 反复装卸搬运各类硬件设备以供测试赛前后组装、拆卸集群，整理线纺、布置机框

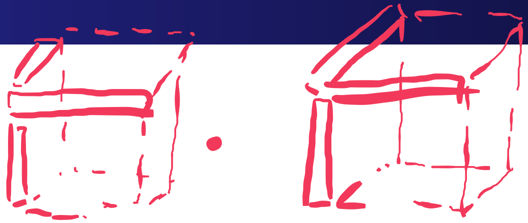
② 脑力活
  ① 安装维护系统、修复问题
  ② 搭建监控系统，实时监测功耗、风扇等关键信息
  ③ 配置网络、存储等基础设施

③ 玄学活
  ① 在集群装好之后施法以提高散热效率

CPU & (GP)GPU Fintune

Sgemm

How to optimize a gemm Software?



```
for i
    for j
        for k
            Cki = aij + bjk
```

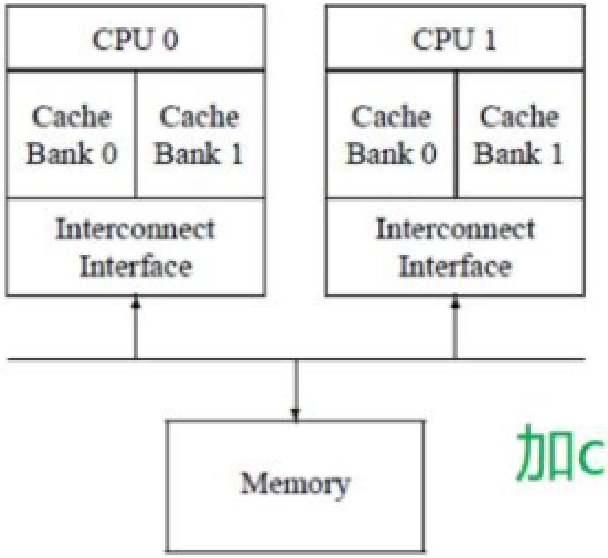
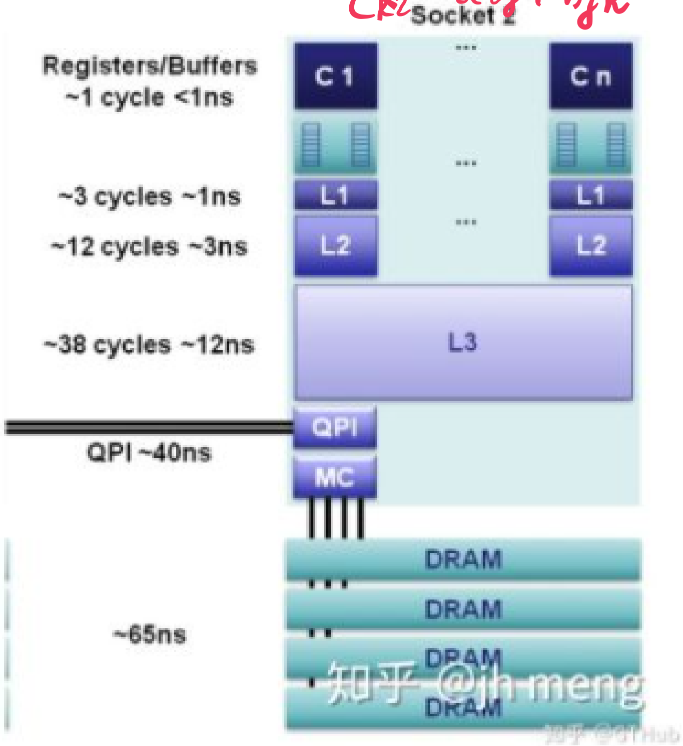
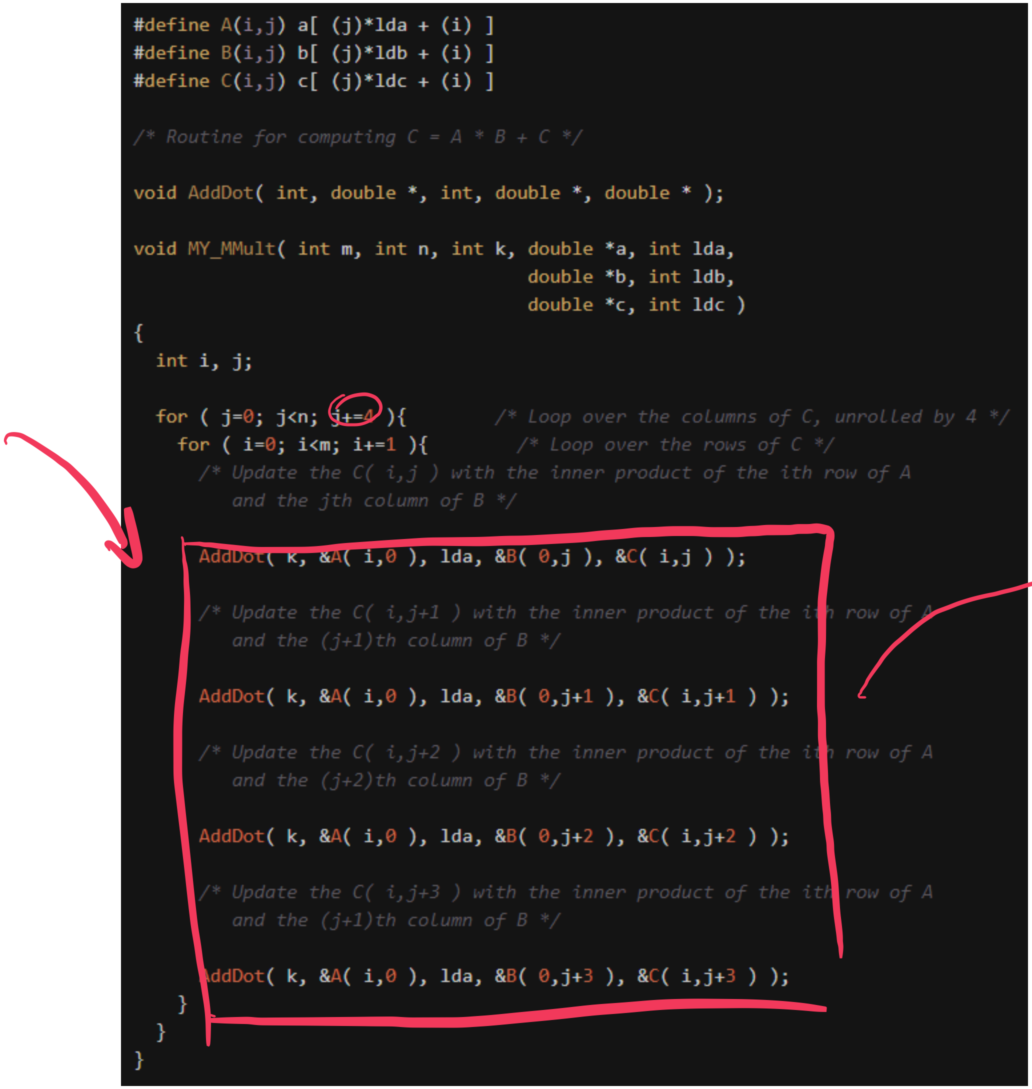


Figure 2: Hardware Parallelism

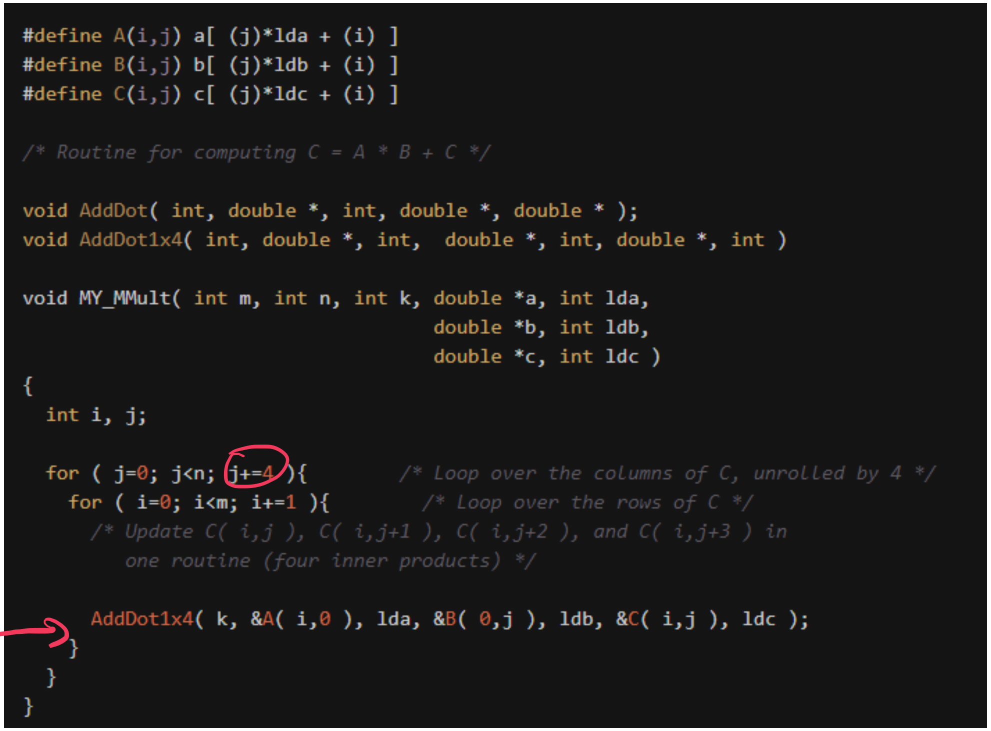
加cache



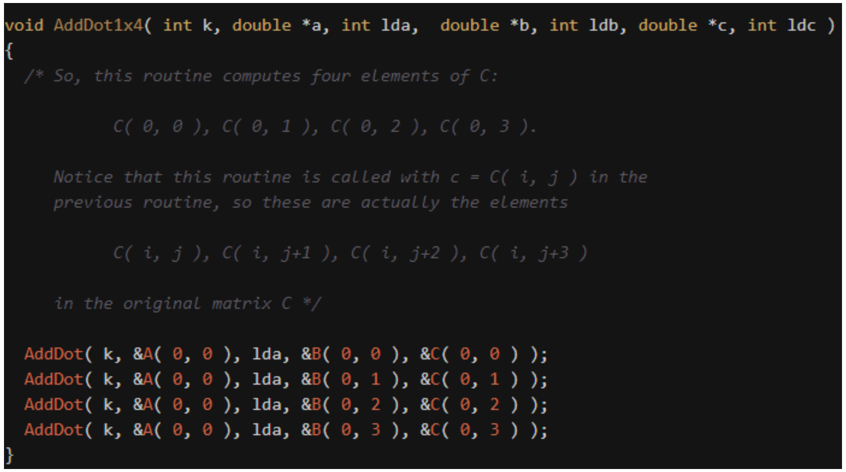
How to optimize a gemm Software?



```c
#define A(i,j) a[ (j)*lda + (i) ]
#define B(i,j) b[ (j)*ldb + (i) ]
#define C(i,j) c[ (j)*ldc + (i) ]

/* Routine for computing C = A * B + C */

void AddDot( int, double *, int, double *, double * );

void MY_MMult( int m, int n, int k, double *a, int lda,
                                    double *b, int ldb,
                                    double *c, int ldc )
{
  int i, j;

  for ( j=0; j<n; j+=4 ){        /* Loop over the columns of C, unrolled by 4 */
    for ( i=0; i<m; i+=1 ){        /* Loop over the rows of C */
      /* Update the C( i,j ) with the inner product of the ith row of A
         and the jth column of B */

      AddDot( k, &A( i,0 ), lda, &B( 0,j ), &C( i,j ) );

      /* Update the C( i,j+1 ) with the inner product of the ith row of A
         and the (j+1)th column of B */

      AddDot( k, &A( i,0 ), lda, &B( 0,j+1 ), &C( i,j+1 ) );

      /* Update the C( i,j+2 ) with the inner product of the ith row of A
         and the (j+2)th column of B */

      AddDot( k, &A( i,0 ), lda, &B( 0,j+2 ), &C( i,j+2 ) );

      /* Update the C( i,j+3 ) with the inner product of the ith row of A
         and the (j+1)th column of B */

      AddDot( k, &A( i,0 ), lda, &B( 0,j+3 ), &C( i,j+3 ) );
    }
  }
}
```

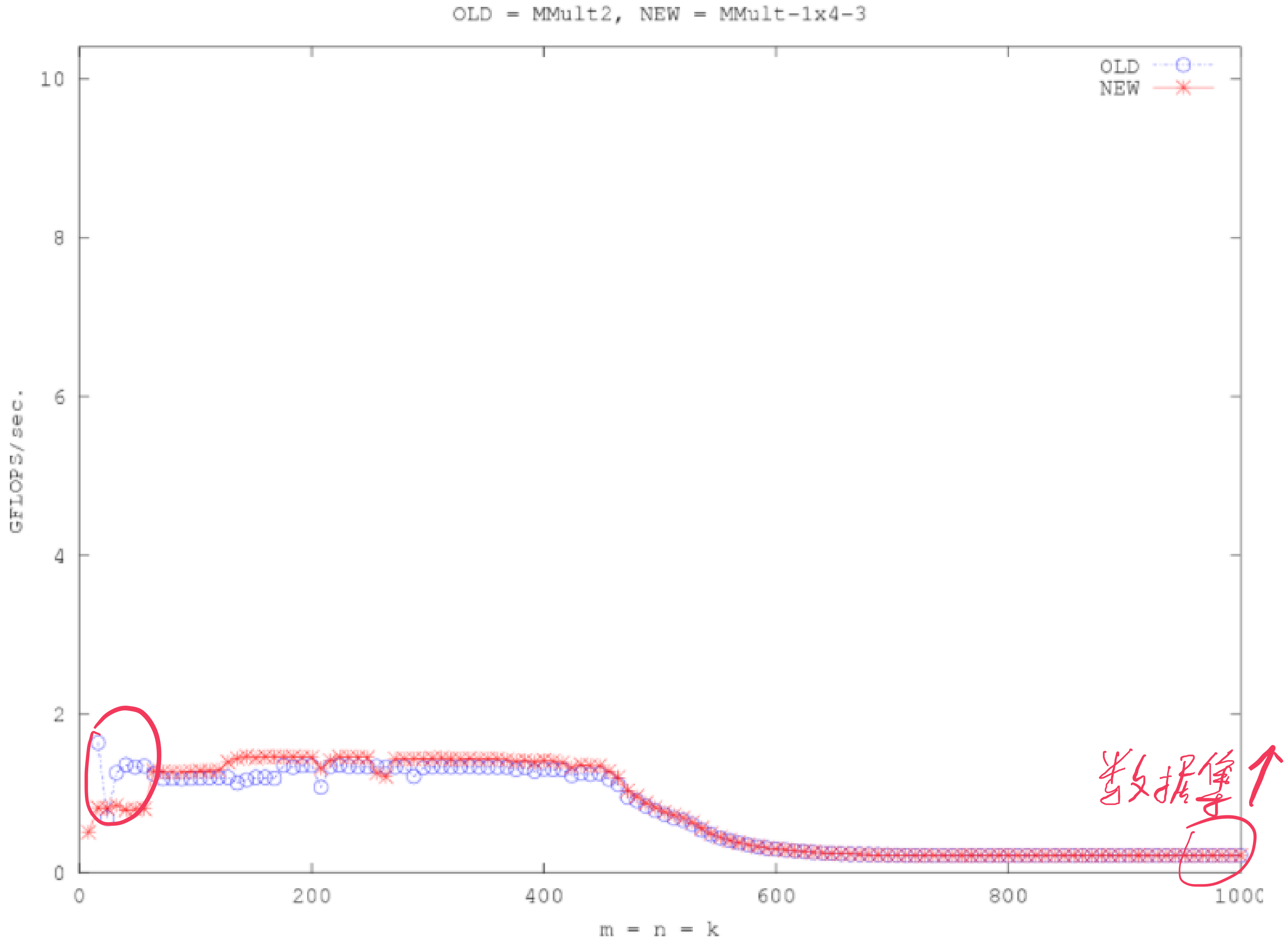


```c
#define A(i,j) a[ (j)*lda + (i) ]
#define B(i,j) b[ (j)*ldb + (i) ]
#define C(i,j) c[ (j)*ldc + (i) ]

/* Routine for computing C = A * B + C */

void AddDot( int, double *, int, double *, double * );
void AddDot1x4( int, double *, int,  double *, int, double *, int )

void MY_MMult( int m, int n, int k, double *a, int lda,
                                    double *b, int ldb,
                                    double *c, int ldc )
{
  int i, j;

  for ( j=0; j<n; j+=4 ){        /* Loop over the columns of C, unrolled by 4 */
    for ( i=0; i<m; i+=1 ){        /* Loop over the rows of C */
      /* Update C( i,j ), C( i,j+1 ), C( i,j+2 ), and C( i,j+3 ) in
         one routine (four inner products) */

      AddDot1x4( k, &A( i,0 ), lda, &B( 0,j ), ldb, &C( i,j ), ldc );
    }
  }
}
```

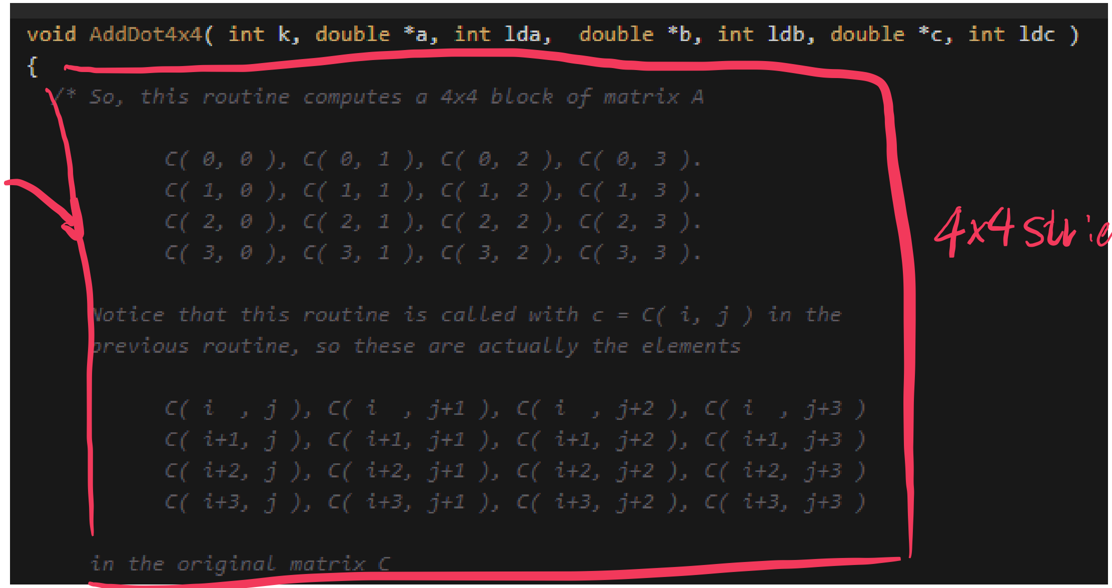


```c
void AddDot1x4( int k, double *a, int lda,  double *b, int ldb, double *c, int ldc )
{
  /* So, this routine computes four elements of C:

           C( 0, 0 ), C( 0, 1 ), C( 0, 2 ), C( 0, 3 ).

     Notice that this routine is called with c = C( i, j ) in the
     previous routine, so these are actually the elements

           C( i, j ), C( i, j+1 ), C( i, j+2 ), C( i, j+3 )

     in the original matrix C */

  AddDot( k, &A( 0, 0 ), lda, &B( 0, 0 ), &C( 0, 0 ) );
  AddDot( k, &A( 0, 0 ), lda, &B( 0, 1 ), &C( 0, 1 ) );
  AddDot( k, &A( 0, 0 ), lda, &B( 0, 2 ), &C( 0, 2 ) );
  AddDot( k, &A( 0, 0 ), lda, &B( 0, 3 ), &C( 0, 3 ) );
}
```

How to optimize a gemm Software?

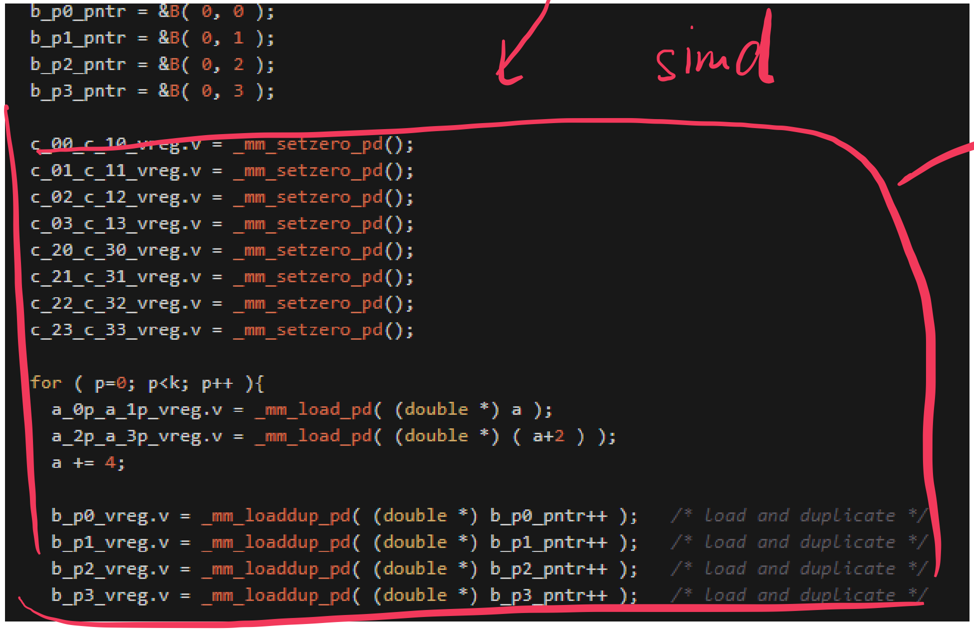


How to optimize a gemm Software?



```
void AddDot4x4( int k, double *a, int lda,  double *b, int ldb, double *c, int ldc )
{
  /* So, this routine computes a 4x4 block of matrix A

           C( 0, 0 ), C( 0, 1 ), C( 0, 2 ), C( 0, 3 ).
           C( 1, 0 ), C( 1, 1 ), C( 1, 2 ), C( 1, 3 ).
           C( 2, 0 ), C( 2, 1 ), C( 2, 2 ), C( 2, 3 ).
           C( 3, 0 ), C( 3, 1 ), C( 3, 2 ), C( 3, 3 ).

     Notice that this routine is called with c = C( i, j ) in the
     previous routine, so these are actually the elements

           C( i  , j ), C( i  , j+1 ), C( i  , j+2 ), C( i  , j+3 )
           C( i+1, j ), C( i+1, j+1 ), C( i+1, j+2 ), C( i+1, j+3 )
           C( i+2, j ), C( i+2, j+1 ), C( i+2, j+2 ), C( i+2, j+3 )
           C( i+3, j ), C( i+3, j+1 ), C( i+3, j+2 ), C( i+3, j+3 )

     in the original matrix C
```

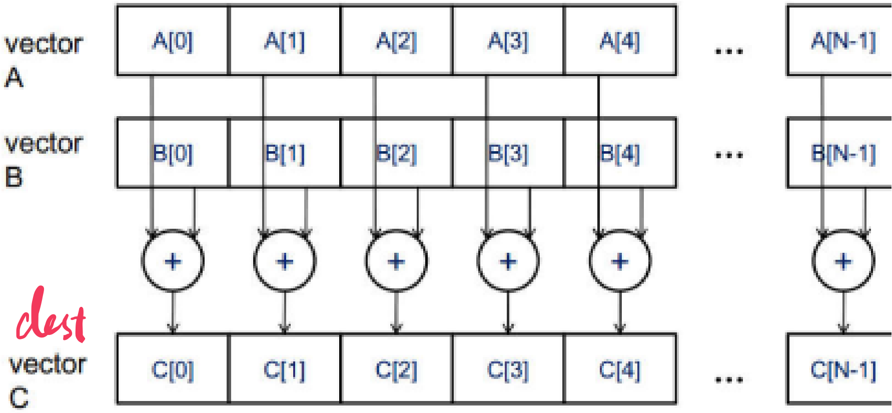
4x4 stride?



```
  b_p0_pntr = &B( 0, 0 );
  b_p1_pntr = &B( 0, 1 );
  b_p2_pntr = &B( 0, 2 );
  b_p3_pntr = &B( 0, 3 );

  c_00_c_10_vreg.v = _mm_setzero_pd();
  c_01_c_11_vreg.v = _mm_setzero_pd();
  c_02_c_12_vreg.v = _mm_setzero_pd();
  c_03_c_13_vreg.v = _mm_setzero_pd();
  c_20_c_30_vreg.v = _mm_setzero_pd();
  c_21_c_31_vreg.v = _mm_setzero_pd();
  c_22_c_32_vreg.v = _mm_setzero_pd();
  c_23_c_33_vreg.v = _mm_setzero_pd();

  for ( p=0; p<k; p++ ){
    a_0p_a_1p_vreg.v = _mm_load_pd( (double *) a );
    a_2p_a_3p_vreg.v = _mm_load_pd( (double *) ( a+2 ) );
    a += 4;

    b_p0_vreg.v = _mm_loaddup_pd( (double *) b_p0_pntr++ );   /* load and duplicate */
    b_p1_vreg.v = _mm_loaddup_pd( (double *) b_p1_pntr++ );   /* load and duplicate */
    b_p2_vreg.v = _mm_loaddup_pd( (double *) b_p2_pntr++ );   /* load and duplicate */
    b_p3_vreg.v = _mm_loaddup_pd( (double *) b_p3_pntr++ );   /* load and duplicate */
```

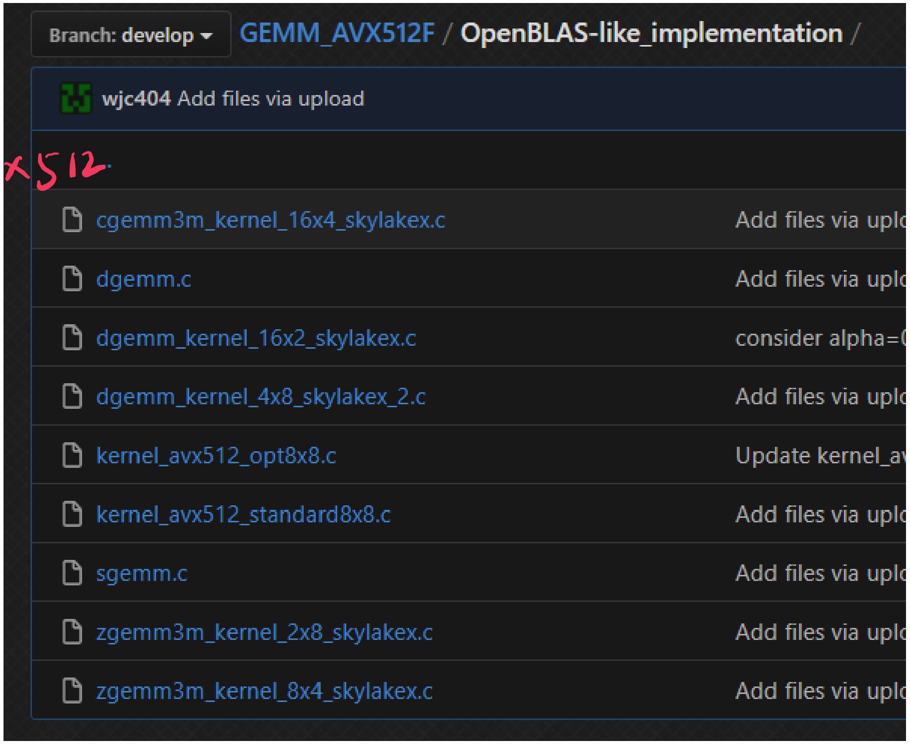
simd

xmm



dest

avx512.



Branch: develop ▾ GEMM_AVX512F / OpenBLAS-like_implementation /

wjc404 Add files via upload

cgemm3m_kernel_16x4_skylakex.c	Add files via upl
dgemm.c	Add files via upl
dgemm_kernel_16x2_skylakex.c	consider alpha=(
dgemm_kernel_4x8_skylakex_2.c	Add files via upl
kernel_avx512_opt8x8.c	Update kernel_av
kernel_avx512_standard8x8.c	Add files via upl
sgemm.c	Add files via upl
zgemm3m_kernel_2x8_skylakex.c	Add files via upl
zgemm3m_kernel_8x4_skylakex.c	Add files via upl

Our HPC Composition

MPI

。MIMD 模型，多进程（多机）

。进程/线程绑定：numactl

。使用 UCX 框架基于 IB 进行通信（环状）

OpenMP

。SMT 模型，多线程（单机）

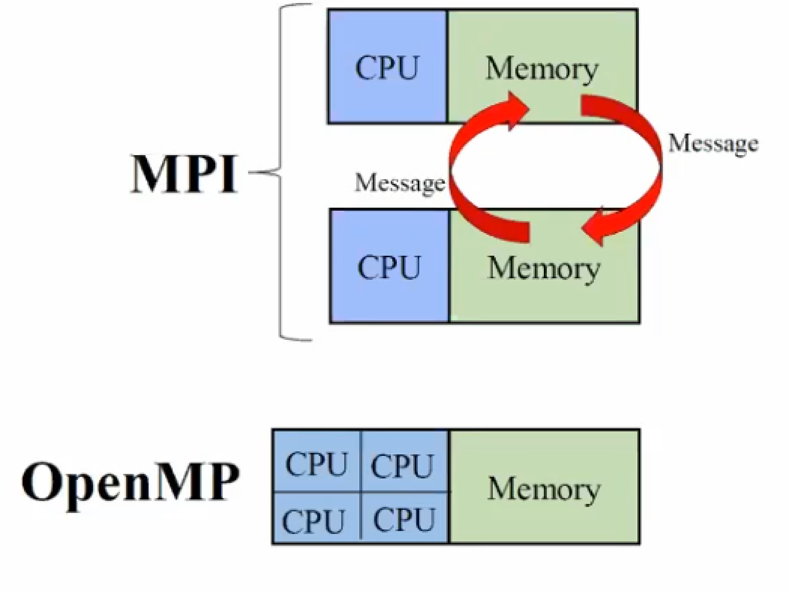
。线程绑定：OMP_AFFINITY

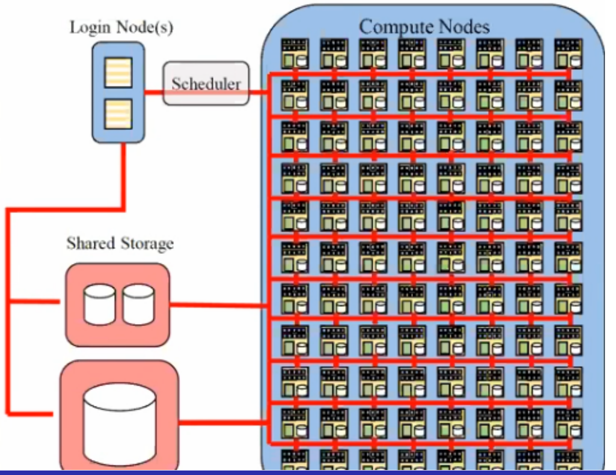
* pthread

CUDA

。SIMT 模型，在 GPU 上进行 (简单的) 大规模并行

。可与 MPI / OpenMP 结合：CUDA-aware MPI NCCL

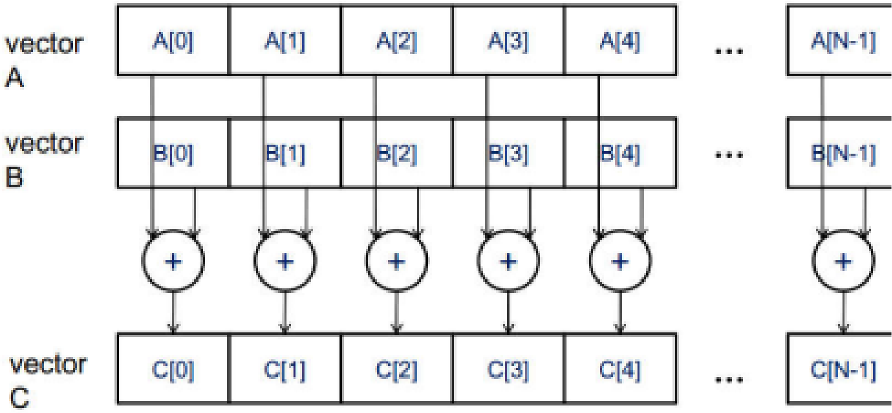




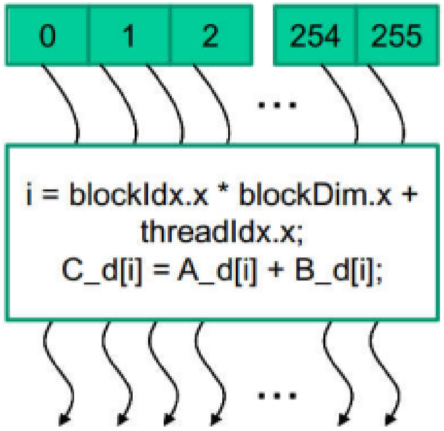
Cuda Basics

- Sequential program iterates through the elements.

```
void vecAdd(float* A, float* B, float* C, int n)
{
  for (i = 0, i < n, i++)
    C[i] = A[i] + B[i];
}
```

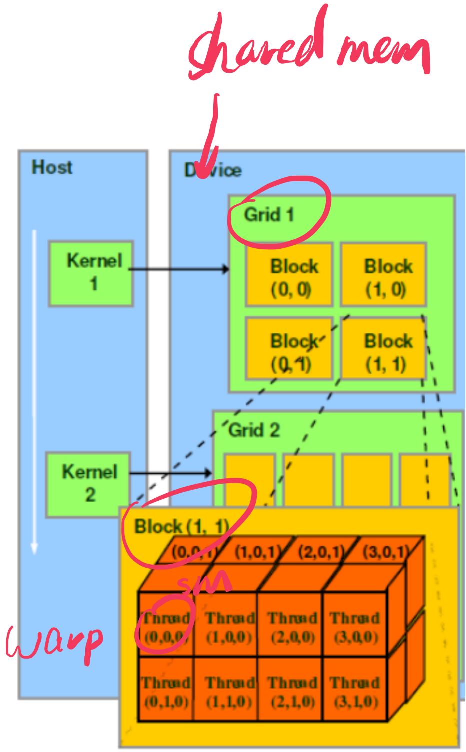


- GPU kernel launches many threads, one for each vector element.
  - □ Potentially millions of threads.
  - □ Hardware ensures low (almost zero) overhead thread management.

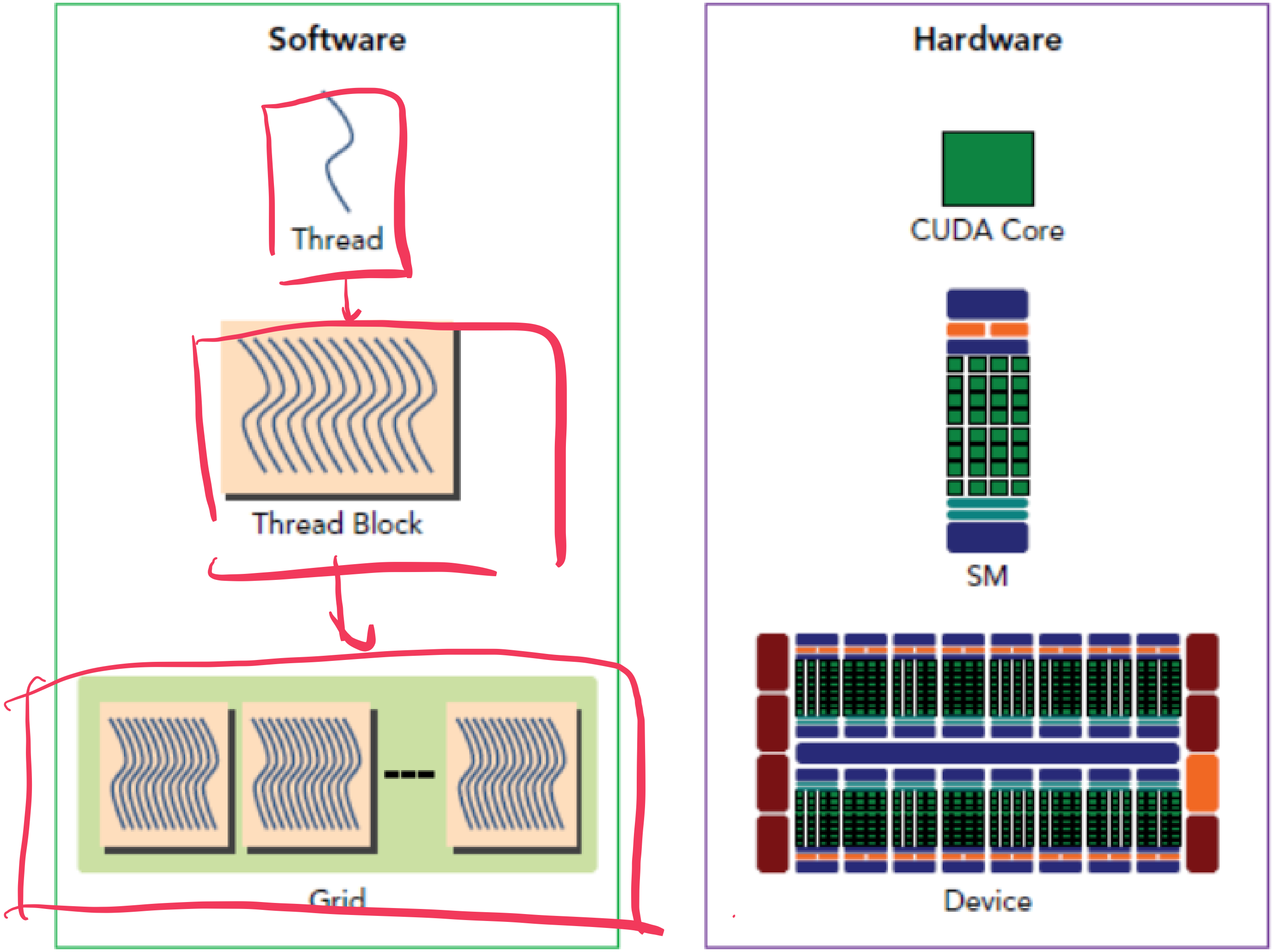


```
i = blockIdx.x * blockDim.x +
threadIdx.x;
C_d[i] = A_d[i] + B_d[i];
```

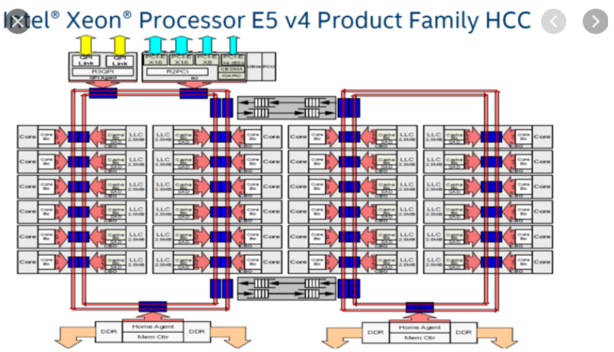
```
__global__
void vecAddKernel(float* A_d, float* B_d, float* C_d, int n)
{
    int i = threadIdx.x + blockDim.x * blockIdx.x;
    if(i<n) C_d[i] = A_d[i] + B_d[i];
}
```



Cuda Basics (With thread)



Core Affinity



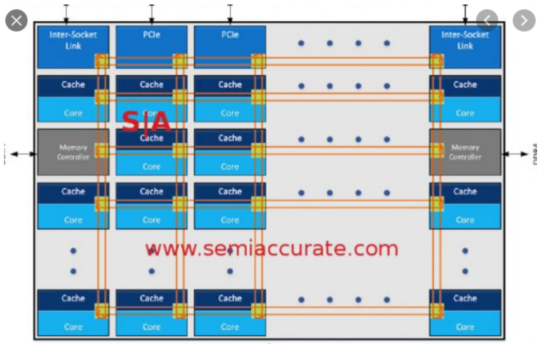
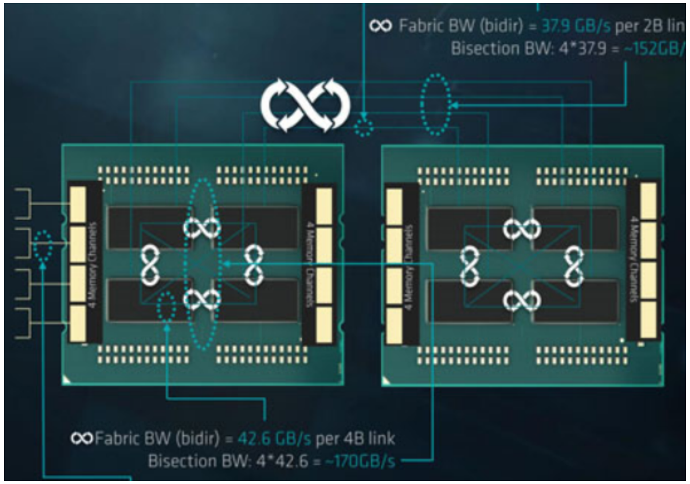
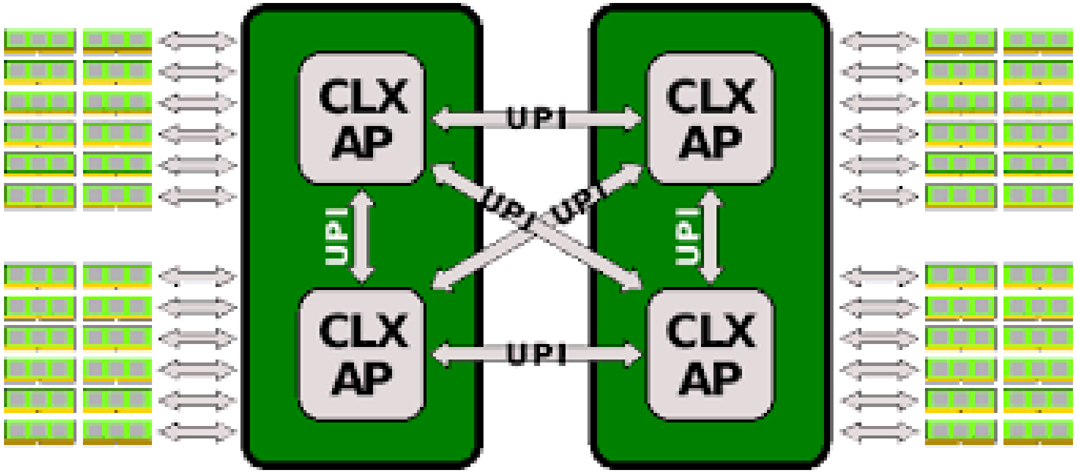


Figure 1: Mesh architecture conceptual representation


∞ Fabric BW (bidir) = 37.9 GB/s per 2B lin
Bisection BW: 4*37.9 = ~152GB/
4 Memory Channels
4 Memory Channels
4 Memory Channels
4 Memory Channels
∞Fabric BW (bidir) = 42.6 GB/s per 4B link
Bisection BW: 4*42.6 = ~170GB/s



```
taskset [options] mask command [argument...]
taskset [options] -p [mask] pid
```

Misc

- **CPU** Power:
  - Tune the best # of cores (Disable some when necessary)
  - BIOS settings
    a. Hyper-threading ?
    b. Turbo Boost ?
- **GPU** Power:
  - `nvidia-smi`
  - Persistence Mode ?
- **Fans** Power: 风扇速度 ~ 性能比 Tradeoff
- Power **Monitoring**: IMPORTANT
  i. Power Meter
  ii. Intelligent Platform Management Interface (*IPMI*)
  iii. 有 GUI: `Grafana`

> e.g. 特定应用 Power-curve 形状规律分析, 能提前预知下一次 Power 高峰并作出应对.

Optimize OS layer

Bottleneck by OS in HPC

numctl

Work Scheduler: Core isolation - prevent 降频恢复overhead

API syscall

Kernel Bypass - I/O read()/write() no internal lock

Cache invalidation / page fault - memory hiearachy

Zero-copy / shared memory - false sharing

OS

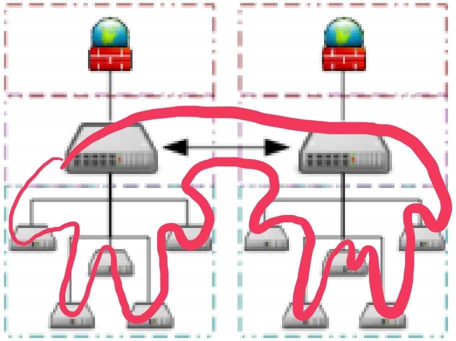
Avoid thread lock / busy spin -Modify your code

loop 并行

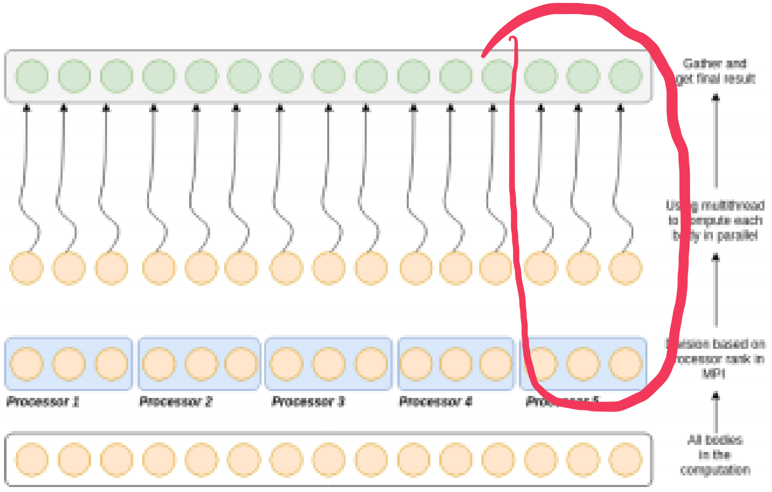
Non-blocking (context switching) - process binding

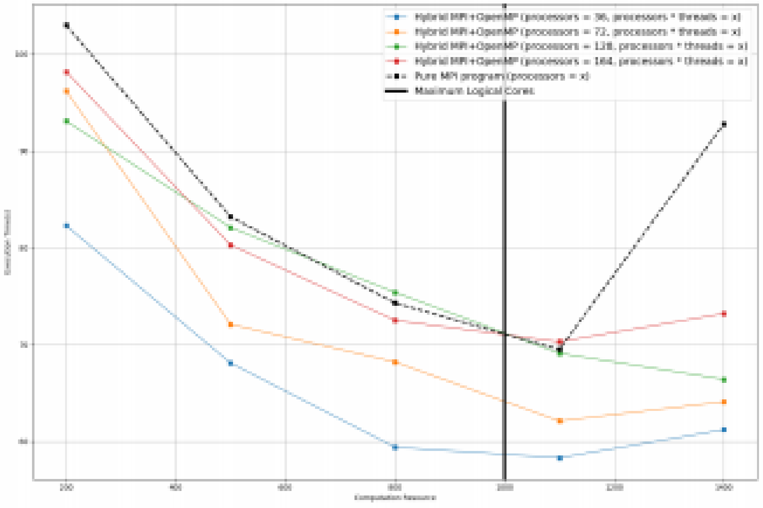
What's OS

- Opt 3 - Hybrid Process and Thread
  - Environment: multi-node cluster



  - Ex. N-body simulation on Sunway Taihulight Supercomputer


Gather and get final result
Using multithread to compute each body in parallel
Division based on processor rank in MPI
All bodies in the computation
Processor 1
Processor 2
Processor 3
Processor 4
Processor 5



本科生培养

Some food for thoughts

https://github.com/ntuhpc/training-ay1819

https://github.com/Kobzol/hardware-effects-gpu

https://github.com/kobzol/hardware-effects

https://wiki.geekpie.club/hpc

rcore/ucore xv6

Some Books uploaded in the qq group

Computer Archietecture by onur

Compiler by Stanford

Operating System by jyy nju

....

vitowu.cn

enigmahuang.me

....

Some Tools recommendation

Vim ？？Cheat Sheat / 脚本+plugin>>Vscode

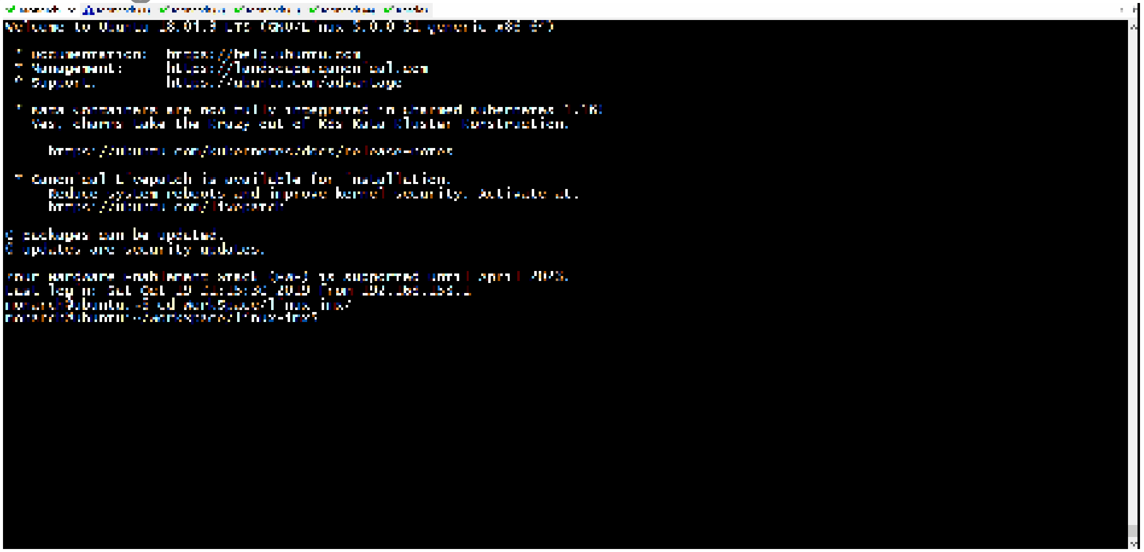
Arch - Autogen Make CMake/ Hackintosh - UEFI ACPI

Jetson nano - DevOps + SLAM + GPU TVM

Vtune Profiler - RL 自适应优化

常用炼丹工具的掉包与调参，有一定的看论文比如LSTM、RNN、（预训练）BERT、Transformer功底。

Learning by doing



```
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Kata Containers are now fully integrated in Charmed Kubernetes 1.16!
   Yes, charms take the Krazy out of K8s Kata Kluster Konstruction.

     https://ubuntu.com/kubernetes/docs/release-notes

 * Canonical Livepatch is available for installation.
     Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Sat Oct 19 21:15:30 2019 from 192.168.153.1
[illegible]@ubuntu:~$ cd workspace/linux-lab/
[illegible]@ubuntu:~/workspace/linux-lab$
```

Some food for thoughts

https://github.com/ntuhpc/training-ay1819

https://github.com/Kobzol/hardware-effects-gpu

https://github.com/kobzol/hardware-effects

https://wiki.geekpie.club/hpc

rcore/ucore xv6

Some Books uploaded in the qq group

Computer Archietecture by onur

Compiler by Stanford

Operating System by jyy nju

………………..

vitowu.cn

enigmahuang.me

……………….

Some Tools recommendation

Vim ？？Cheat Sheat / 脚本+plugin>>Vscode

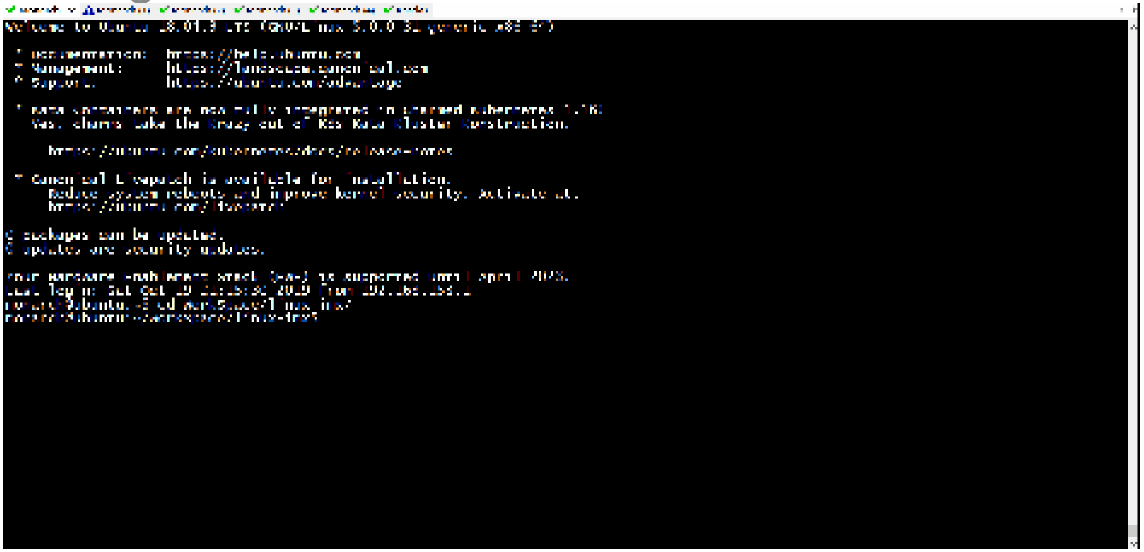
Arch - Autogen Make CMake/ Hackintosh - UEFI ACPI

Jetson nano - DevOps + SLAM + GPU TVM

Vtune Profiler - RL 自适应优化

常用炼丹工具的掉包与调参，有一定的看论文比如LSTM、RNN、（预训练）BERT、Transformer功底。

Learning by doing



```
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Kata Containers are now fully integrated in Charmed Kubernetes 1.16!
   Yes, charms take the Krazy out of K8s Kata Kluster Konstruction.

     https://ubuntu.com/kubernetes/docs/release-notes

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Sat Oct 19 [illegible] 2019 from 192.168.[illegible]
[illegible]@ubuntu:~$ cd workspace/linux_inx/
[illegible]@ubuntu:~/workspace/linux-inx$
```

Some Tools recommendation

Vim ？？Cheat Sheat / 脚本+plugin>>Vscode

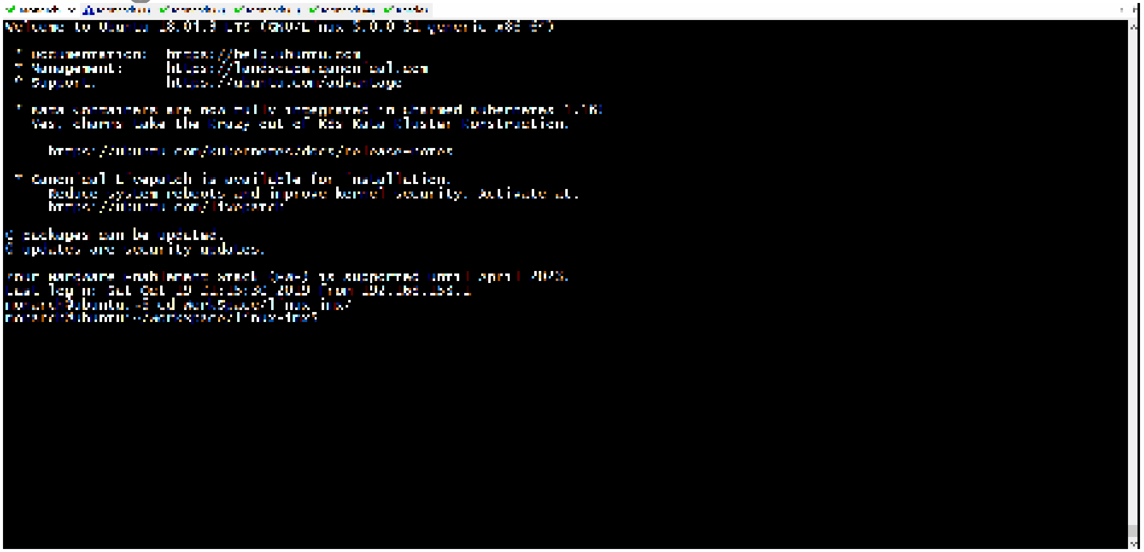
Arch - Autogen Make CMake/ Hackintosh - UEFI ACPI

Jetson nano - DevOps + SLAM + GPU TVM

Vtune Profiler - RL 自适应优化

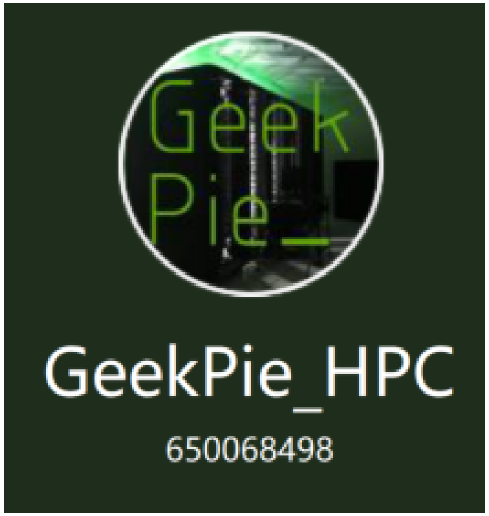
常用炼丹工具的掉包与调参，有一定的看论文比如LSTM、RNN、（预训练）BERT、Transformer功底。

Learning by doing



```
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Kata Containers are now fully integrated in Charmed Kubernetes 1.16!
   Yes, charms take the Krazy out of K8s Kata Kluster Konstruction.

     https://ubuntu.com/kubernetes/docs/release-notes

 * Canonical Livepatch is available for installation.
     Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Sat Oct 19 [illegible] 2019 from 192.168.153.1
[illegible]@ubuntu:~$ cd workspace/linux_img/
[illegible]@ubuntu:~/workspace/linux-img$
```

Thanks!



GeekPie_HPC

650068498