Lab 2-0
QCT SERVER

Parallel Programming
2024/09/26

System Spec

Slurm partition : qct-cpu

96 cores per node

2 threads per core

Maximum allocate node per job : 1

MPI module : openmpi/4.1.6, mpi/latest

Slurm sbatch

```
#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --output=my_job_%j.out
#SBATCH --error=my_job_%j.err
#SBATCH --time=00:05:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=1
```

Lab 2-1
VScode SSH & SSH-key

Parallel Programming
2024/09/26

Generate SSH key

On your computer:

- ssh-keygen -t rsa -b 4096
- cat ~/.ssh/id_rsa.pub

copy it

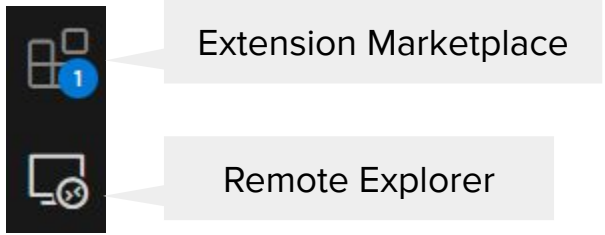
On the remote server:

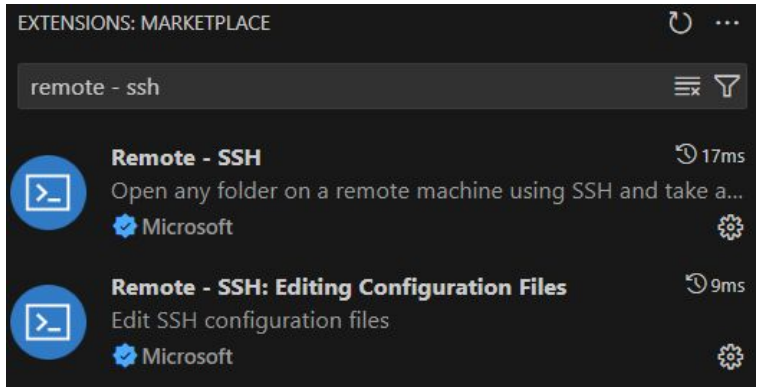
- mkdir ~/.ssh
- vim ~/.ssh/authorized_keys

paste the key into it

VScode SSH-Config

1. Go to extension marketplace
2. Install both of them
3. Go to remote explorer
4. Open SSH Config File





VScode SSH-Config

Host : An alias use to refer to this server

HostName : IP address of the target server

User : Username

ProxyJump : Tells SSH to use the another host as an jump server

IdentityFile : The private key file to use for authentication

Port : Port to connect the target server

```
Host pp24-qctserver
  HostName 192.168.176.61
  User pp24s000
  ProxyJump pp24-jump
  #IdentityFile ~/.ssh/id_rsa

Host pp24-jump
  HostName 60.250.52.226
  User pp24s000
  Port 51983
  #IdentityFile ~/.ssh/id_rsa
```

After setting up, Ctrl+S and refresh 

VScode SSH-Config

如果你是用前面簡報的方式設定,

請將你本地的public key複製到跳板機以及QCT server。

如果你是先連線到跳板機再連線到QCT server,

跳板機上要放本地的public key以外，要再生成一個key,

並將這個key放到QCT server。

或是你也可以打密碼就好

Lab 2-2
Pthread & OpenMP

Parallel Programming
2024/09/26

Linking Pthread & Openmp

Linking pthread

gcc your_program.c -o your_program -pthread

Linking openmp

gcc your_program.c -o your_program -fopenmp

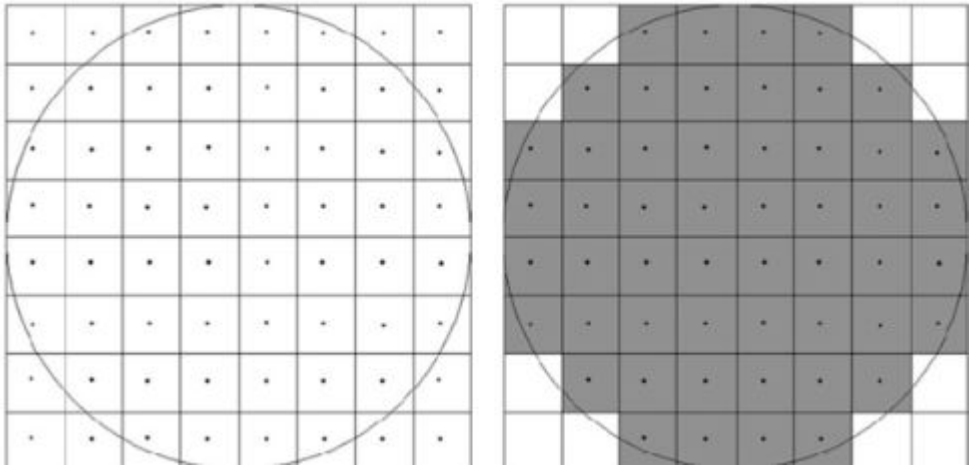
Linking both

gcc your_program.c -o your_program -pthread -fopenmp

Approximate pixels

Copy source file to your home directory : cp -r /home/share/lab2 ~/

lab2 judges : lab2_pthread-judge lab2_omp-judge lab2_hybrid-judge



Approximate pixels using pthread and OpenMP

- Modify the sequential code lab2_pthread.cc with pthread
  - g++ lab2_pthread.cc -o lab2_pthread -pthread -lm
  - srun -c4 -n1 ./lab2_pthread ? ?
- Modify the sequential code lab2_openmp.cc with openmp
  - g++ lab2_openmp.cc -o lab2_openmp -fopenmp -lm
  - srun -c4 -n1 ./lab2_openmp ? ?

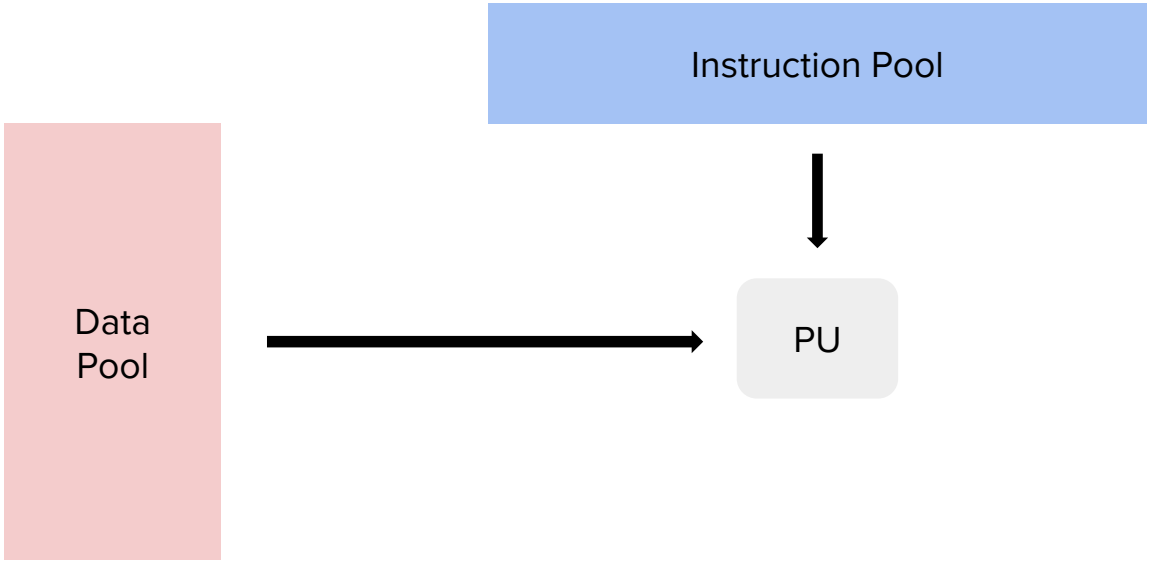
Approximate pixels using Hybrid MPI with OpenMP

- Modify the sequential code lab2_hybrid.cc with MPI and OpenMP
  - mpicxx lab2_hybrid.cc -o lab2_hybrid -fopenmp -lm
  - srun -n6 -c4 ./lab2_hybrid ? ?

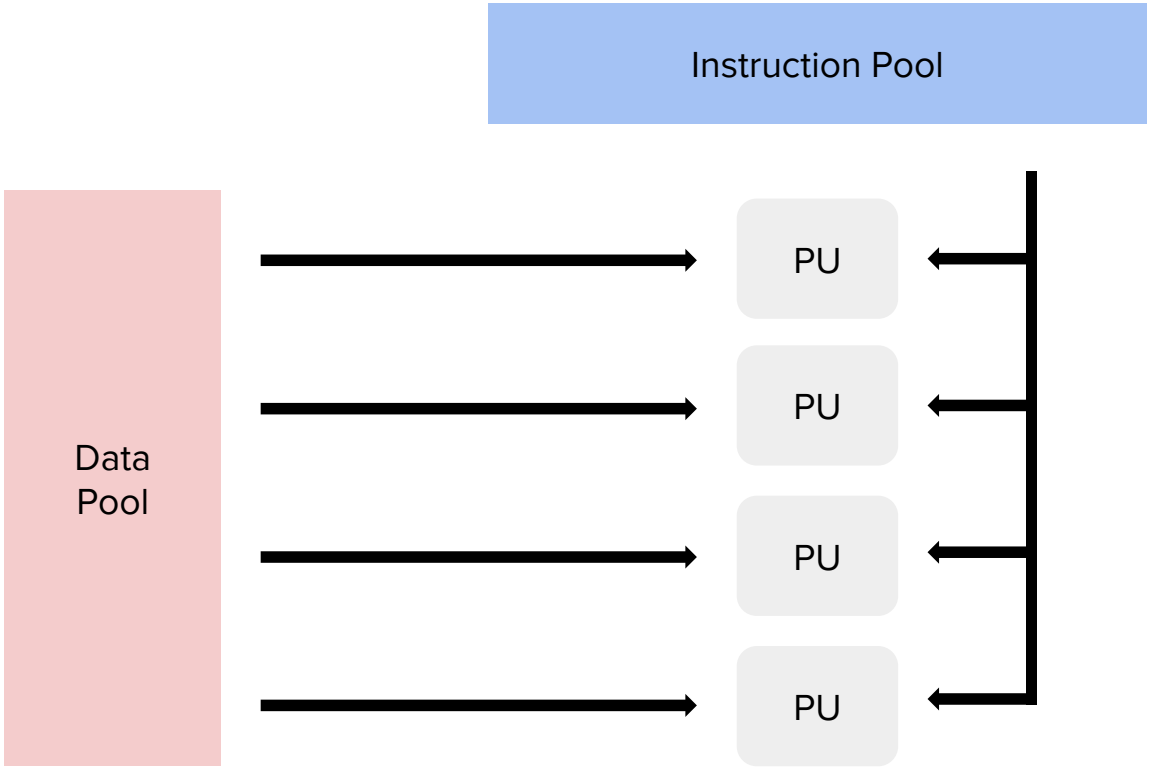
Lab 2-3
Vectorization

Parallel Programming
2024/09/26

SISD



SIMD



Vector Instruction Set

lscpu can be used to display the vector instruction sets supported by the CPU

On the QCT server, sse/sse2/avx/avx2/avx512 and more are available.

```
Flags:                           fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb
rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sd
bg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l3 cat_l2 cdp_l3 invp
cid_single cdp_l2 ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid cqm rdt_a avx512f avx5
12dq rdseed adx smap avx512ifma clflushopt clwb intel_pt avx512cd sha_ni avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local avx_v
nni avx512_bf16 wbnoinvd dtherm ida arat pln pts hfi avx512vbmi umip pku ospke waitpkg avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni avx512_bitalg tme avx512_vpopcntdq la57 rdpi
d bus_lock_detect cldemote movdiri movdir64b enqcmd fsrm md_clear serialize tsxldtrk pconfig arch_lbr ibt amx_bf16 avx512_fp16 amx_tile amx_int8 flush_l1d arch_capabilities
```

Vector Instruction Set

SSE（Streaming SIMD Extensions）

- 128-bit registers, doubling the width of the 64-bit MMX registers
- SSE only supports 32-bit floating point
- SSE2 adds double, long long, int, char

Vector Instruction Set

AVX（Advanced Vector Extensions）

- Expanded SIMD registers from 128 bits (in SSE) to 256 bits
- Supports both single-precision and double-precision floating-point operations
- AVX512 expands SIMD registers to 512 bits

Vector Instruction Set

1. Codes have to be executed many times, will probably benefit from vectorization

2. If there are no data dependency, it will be easier to vectorize

Data dependency means the value of one data elements depends on another

Automatic Vectorization

GCC Vectorization

- -ftree-vectorize : enabled vectorization
- -O3 : enabled vectorization by default
- -march=native : use instructions supported by the local CPU
- -fopt-info-vec-all : print vectorization log
- #pragma GCC ivdep : tells compiler there is no data dependency in the following loop

Intel Intrinsics

Allows developers to use advanced instruction sets of processors directly in C/C++

Procedure:

1. Load data from memory to the special registers
2. Perform vector instructions
3. Save data from the special registers to memory

Intel Intrinsics

Compile sample code:

g++ -o vectorize_example vectorize_example.cc -march=native

```
void multiple_and_add(float *a, float *b, float *c, float *d, int size){
    for(int i = 0; i < size; i++){
        a[i] = b[i] * c[i] + d[i];
    }
}
```

Original

Intel Intrinsics

```
void vec_multiple_and_add(float *a, float *b, float *c, float *d, int size){

    int i;

    for(i = 0; i < size - 15; i += 16){

        // load data to special registers
        __m512 b_vec = _mm512_loadu_ps(&b[i]);
        __m512 c_vec = _mm512_loadu_ps(&c[i]);
        __m512 d_vec = _mm512_loadu_ps(&d[i]);

        // _mm512_fmadd_ps finish the multiplae and add operation
        // _mm512_storeu_ps store the result to a array
        _mm512_storeu_ps(&a[i], _mm512_fmadd_ps(b_vec, c_vec, d_vec));
    }

    // remaining elements
    for(; i < size; i++){
        a[i] = b[i] * c[i] + d[i];
    }
}
```

Lab 2-4
Profiling

Parallel Programming
2024/09/26

Intel Vtune

module load intel-vtune/2024

Vtune command line syntax introduction : link

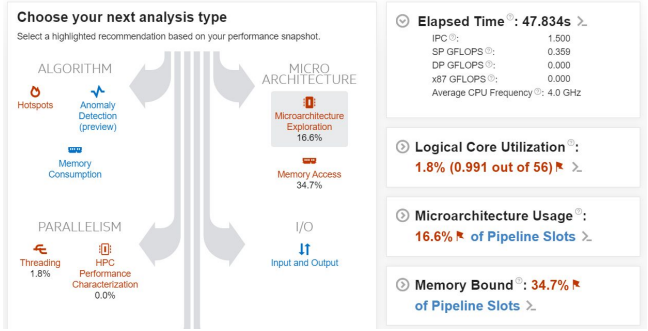
- vtune -h
- vtune -collect hotspots -result-dir $HOME/vtune-result ./vectorize_example
- mpirun -n 2 vtune -collect hotspots -result-dir $HOME/vtune-result ./vectorize_example

Intel Vtune

Install it on your computer : link

Find this :

Choose your .vtune file and open it



performance-snapshot

hotsots

HW2

Parallel Programming
2024/09/26

SPEC