**Operating System and CPU I am using:**

[hlong@snares-02] (37)$ lscpu

Architecture: x86\_64

CPU op-mode(s): 32-bit, 64-bit

Address sizes: 39 bits physical, 48 bits virtual

Byte Order: Little Endian

CPU(s): 12

On-line CPU(s) list: 0-11

Vendor ID: GenuineIntel

Model name: 11th Gen Intel(R) Core(TM) i5-11500 @ 2.70GHz

CPU family: 6

Model: 167

Thread(s) per core: 2

Core(s) per socket: 6

Socket(s): 1

Stepping: 1

CPU max MHz: 4600.0000

CPU min MHz: 800.0000

BogoMIPS: 5424.00

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\_per

fmon pebs bts rep\_good nopl xtopology nonstop\_tsc cpuid aperfmperf tsc\_known\_freq pni pclmulqdq dtes64 monitor ds\_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4\_

1 sse4\_2 x2apic movbe popcnt tsc\_deadline\_timer aes xsave avx f16c rdrand lahf\_lm abm 3dnowprefetch cpuid\_fault epb invpcid\_single ssbd ibrs ibpb stibp ibrs\_enhanced tpr\_shad

ow vnmi flexpriority ept vpid ept\_ad fsgsbase tsc\_adjust bmi1 avx2 smep bmi2 erms invpcid mpx avx512f avx512dq rdseed adx smap avx512ifma clflushopt intel\_pt avx512cd sha\_ni

avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp\_notify hwp\_act\_window hwp\_epp hwp\_pkg\_req avx512vbmi umip pku ospke avx512\_vbmi2 gfni vaes vp

clmulqdq avx512\_vnni avx512\_bitalg avx512\_vpopcntdq rdpid fsrm md\_clear flush\_l1d arch\_capabilities

Virtualization features:

Virtualization: VT-x

Caches (sum of all):

L1d: 288 KiB (6 instances)

L1i: 192 KiB (6 instances)

L2: 3 MiB (6 instances)

L3: 12 MiB (1 instance)

NUMA:

NUMA node(s): 1

NUMA node0 CPU(s): 0-11

Vulnerabilities:

Gather data sampling: Mitigation; Microcode

Itlb multihit: Not affected

L1tf: Not affected

Mds: Not affected

Meltdown: Not affected

Mmio stale data: Mitigation; Clear CPU buffers; SMT vulnerable

Retbleed: Mitigation; Enhanced IBRS

Spec rstack overflow: Not affected

Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp

Spectre v1: Mitigation; usercopy/swapgs barriers and \_\_user pointer sanitization

Spectre v2: Mitigation; Enhanced IBRS, IBPB conditional, RSB filling, PBRSB-eIBRS SW sequence

Srbds: Not affected

Tsx async abort: Not affected

**How I compile the code:**

I first cd to the ***SparseDirect/LaplacePARDISO\_0\_0*** directory and ran this:

source /s/intelcompilers/bin/iccvars.sh intel64

icc \*.cpp -qopenmp -mkl -o output.exe

./output.exe

I have 12 cores in the computer lab machine.

Part A:

**Minimum Degree Algorithm**

Code changes: iparm[1] from 3 to 0.

resulting sparsity of the **L** factor = number of non-zeros in L: 6130764

operations needed for the factorization itself = gflop for the numerical factorization: 5.930508

bandwidth achieved during factorization = gflop/s for the numerical factorization: 65.852905

run-time (in seconds) for the solve step (forward/backward substitution) = Time spent in direct solver at solve step (solve) : 0.003102 s

**"no-permutation" Permutation**

Code changes: iparm[1] stays 3. Iparm[4] from 0 to 1.

And I added this after line 27:

std::vector<MKL\_INT> perm(n);

for (MKL\_INT i = 0; i < n; i++) {

perm[i] = i;

}

resulting sparsity of the **L** factor = number of non-zeros in L: 23561947

operations needed for the factorization itself = gflop for the numerical factorization: 20.992256

bandwidth achieved during factorization = gflop/s for the numerical factorization: 2.369961

run-time (in seconds) for the solve step (forward/backward substitution) = Time spent in direct solver at solve step (solve) : 0.026622 s

Part B:

**Exploration of the impact of multiple right-hand-sides in the solve step**

Code changes: I added this before the configuration for iparm[].

using array\_t = float (&) [XDIM][YDIM][ZDIM\*nrhs];

float \*my\_x\_raw = new float [XDIM\*YDIM\*ZDIM\*nrhs];

array\_t my\_x = reinterpret\_cast<array\_t>(\*my\_x\_raw);

float \*my\_b\_raw = new float [XDIM\*YDIM\*ZDIM\*nrhs];

array\_t my\_b = reinterpret\_cast<array\_t>(\*my\_b\_raw);

for (int a = 0; a < XDIM; a++) {

for (int b = 0; b < YDIM; b++) {

for (int c = 0; c < ZDIM; c++) {

for (int d = 0; d < nrhs; d++){

my\_x[a][b][d\*ZDIM+c] = x[a][b][c];

}

}

}

}

And I changed these lines to use my\_x and my\_b:

// Back substitution and iterative refinement

phase = 33;

iparm[7] = 0; // Max numbers of iterative refinement steps

PARDISO (pt, &maxfct, &mnum, &mtype, &phase, &n,

matrix.GetValues(), matrix.GetRowOffsets(), matrix.GetColumnIndices(),

&idum, &nrhs, iparm, &msglvl, static\_cast<void\*>(&my\_b[0][0][0]), &my\_x[0][0][0], &error);

if ( error != 0 )

throw std::runtime\_error("PARDISO error during solution phase");

std::cout << "Solve completed ... " <<std::endl;

**Here are the results:**

Nrhs = 1:

Time spent in direct solver at solve step (solve) : 0.002506 s

Nrhs = 10:

Time spent in direct solver at solve step (solve) : 0.035940 s

Nrhs = 100:

Time spent in direct solver at solve step (solve) : 0.054061 s