Compile and Run HPC code on Sapelo2

Georgia Advanced Computing Resource Center (GACRC)
Enterprise Information Technology Services (EITS)
The University of Georgia
Outline

• GACRC Sapelo2 Cluster

• HPC Conceptual Framework

• Software Modules for HPC Programming

• Multi-threaded Parallel - An example of OpenMP

• Distributed Parallel - An example of OpenMPI
Note: You need to connect to the UGA network using VPN when accessing from outside of the UGA main campus.
UGA VPN: https://eits.uga.edu/access_and_security/infosec/tools/vpn/
HPC Conceptual Framework

• Serial single-core job
  1. Run computational task in a single/main thread, using one CPU core on a single node

• Threaded memory-sharing parallel job
  1. Run task in multi-threads, using more than one CPU core on a single node
  2. Programming library: OpenMP (Open Multi-Processing)
  3. Important environment variables: OMP_NUM_THREADS, OMP_PROC_BIND (close or spread)
HPC Conceptual Framework

• MPI parallel job
  1. Run task in multiple MPI processes; using one or more CPU cores on a single or multiple nodes
  2. Programming library: OpenMPI, MVAPICH2, Intel MPI Library

• GPU job
  1. Run task in a single thread or multi-threads, using one or more CPU cores on a single host
  2. Use one or more GPU devices
  3. Programming library: OpenACC or CUDA
COMPILE AND RUN HPC CODE ON SAPELO2

7

MPI

Single core

node1

core

core

core

host memory

node2

GPU

OpenMP

node3

OpenACC

host memory

node4

GPU

node5

GPU

node6

GPU

host memory

core

core

host memory

GPU memory

GPU memory

GPU memory

GPU memory

GPU memory

GPU memory
Software Modules for HPC Programming – Compilers

https://wiki.gacrc.uga.edu/wiki/Code_compilation_on_sapelo2

<table>
<thead>
<tr>
<th>GNU gcc/g++/gfortran:</th>
<th>Intel icc/icpc/ifort:</th>
<th>PGI pgcc/pgc++/pgfortran:</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCC/6.4.0-2.28</td>
<td>iccifort/2013_sp1.0.080</td>
<td>PGI/17.9</td>
</tr>
<tr>
<td>GCC/7.3.0-2.30</td>
<td>iccifort/2015.2.164-GCC-4.8.5</td>
<td>PGI/19.10-GCC-8.3.0-2.32</td>
</tr>
<tr>
<td>GCC/8.3.0</td>
<td>iccifort/2018.1.163-GCC-6.4.0-2.28</td>
<td></td>
</tr>
<tr>
<td>GCC/9.2.0</td>
<td>iccifort/2019.5.281</td>
<td></td>
</tr>
<tr>
<td>GCC/9.3.0</td>
<td>iccifort/2020.4.304</td>
<td></td>
</tr>
<tr>
<td>GCC/10.2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCC/10.3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCC/11.2.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Activate OpenMP multi-threaded code at compile time:

GNU : -fopenmp

Intel : -qopenmp

PGI : -mp
Software Modules for HPC Programming [https://wiki.gacrc.uga.edu/wiki/MPI](https://wiki.gacrc.uga.edu/wiki/MPI)

OpenMPI libs:

OpenMPI/3.1.4-GCC-8.3.0

- foss/2019b (OpenBLAS/0.3.7, ScaLAPACK/2.0.2, FFTW/3.3.8)
- gompi/2019b (no math libs)

OpenMPI/3.1.4-gcccuda-2019b

- fosscuda/2019b (math libs of foss/2019b + CUDA/10.1.243)
- gompic/2019b (no math libs, CUDA/10.1.243)

OpenMPI/4.0.5-GCC-10.2.0

- foss/2020b (OpenBLAS/0.3.12, ScaLAPACK/2.1.0, FFTW/3.3.8)
- gompi/2020b (no math libs)

OpenMPI/4.0.5-gcccuda-2020b

- fosscuda/2020b (math libs of foss/2020b + CUDA/11.1.1)
- gompic/2020b (no math libs, CUDA/11.1.1)
# Software Modules for HPC Programming

[https://wiki.gacrc.uga.edu/wiki/MPI](https://wiki.gacrc.uga.edu/wiki/MPI)

## Intel MPI libs:

<table>
<thead>
<tr>
<th>Version</th>
<th>Compiler</th>
<th>IMPI Version</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>impi/2021.2.0</code>-intel-compilers-2021.2.0</td>
<td><code>intel/2021a</code> (imkl/2021.2.0)</td>
<td></td>
<td><code>iimpi/2021a</code> (no math libs)</td>
</tr>
</tbody>
</table>
Software Modules for HPC Programming [https://wiki.gacrc.uga.edu/wiki/MPI](https://wiki.gacrc.uga.edu/wiki/MPI)

**MVAPICH2 MPI libs:**

MVAPICH2/2.3.4-GCC-8.3.0  
- gmvolf/2019b (OpenBLAS/0.3.7, FFTW/3.3.8, and ScaLAPACK/2.0.2)  
- gmvapich2/2019b (no math libs)

MVAPICH2/2.3.6-GCC-9.3.0  
- gmvolf/2020a (OpenBLAS/0.3.9, FFTW/3.3.8, and ScaLAPACK/2.0.2)  
- gmvapich2/2020a (no math libs)
Software Modules for HPC Programming – Tips

Tip1: Some MPI lib modules don’t work on old AMD Opteron nodes (lack of AVX2 to run recent OpenBLAS). Use EDR node feature (#SBATCH --constraint=EDR) to run a MPI program built with foss/2020b, fosscuda/2020b, gmvolf/2020a, or gmvapich2/2020a.

Tip2: Both srun and mpirun can launch a MPI program. Which one is better? srun has tighter Slurm integration than mpirun. In some cases mpirun can encounter inter-rank communication problems.

Tip3: When using srun to lunch a MPI program built with MVAPICH2, you need to use --mpi=pmi2 option, for example, srun --mpi=pmi2 –n 400 program.x
Software Modules for HPC Programming – Tips

Tip4: Different MPI lib modules are included in the corresponding foss (2019b, 2020b), intel (2019b, 2020b, 2021a,) and gmvolf (2019b, 2020a) modules. If you use foss, intel or gmvolf modules, you don’t need to load the already included MPI lib modules.

E.g. 1: ml OpenMPI/4.0.5-GCC-10.2.0 ➔ don’t need
       ml foss/2020b

E.g. 2: ml intel/2020b
       ml impi/2019.9.304-iccifort-2020.4.304 ➔ don’t need

Tip5: When you load more than one software modules, toolchain compatibility is the most important thing you need to pay attention to. If you load more than one module and some toolchains are incompatible, you will end up with failing dependencies or Lmod errors. GACRC Kaltura Video: https://kaltura.uga.edu/media/t/1_6vek2zt5/176125031
Multi-threaded Parallel - An example of OpenMP

• Numerical approximation of the integral of De Jong function:

\[ f(x, y) = \left(0.002 + \sum_{i=-2}^{2} \sum_{j=-2}^{2} \frac{1}{5(i + 2) + (j + 3) + (x - 16j)^6 + (y - 16i)^6}\right)^{-1} \]

\[ x \in [0,15], y \in [0,15], \Delta x = \Delta y = 0.001 \]

• Demo folder on Sapelo2: /usr/local/training/CSP_seminar_09062022/demo_omp/
  1. integral_dejong.cpp : sample code
  2. sub_omp.sh : batch job submission script (sbatch sub_omp.sh)
  3. util : folder storing memu.sh
  4. cmd_list : list of commands
  5. makefile : compile code in an interactive session:
     ▶ interact -p batch -c 32 --mem 8gb --constraint EDR
     ▶ copy demo folder to your working folder
     ▶ In your demo_omp folder, run ml GCC/8.3.0 && make
Multi-threaded Parallel - An example of OpenMP

- **System:**
  - a5-10 AMD Milan: AMD EPYC 7713P 64-Core, 128GB RAM (--constraint Milan)
  - a5-2 AMD Rome: AMD EPYC 7702P 64-Core, 128GB RAM (--constraint Rome)
  - d2-3 AMD Naples: AMD EPYC 7551P 32-Core, 128GB RAM (--constraint Naples)
  - c4-6 Intel Skylake: Intel(R) Xeon(R) Gold 6130 32-Core, 192GB RAM (--constraint Skylake)

- **Task Monitoring:**
  - Time:
    - export OMP_DISPLAY_ENV=verbose
    - export OMP_PLACES=cores
    - export OMP_PROC_BIND=close
    - export OMP_NUM_THREADS=<1,4,8,16,24,32>
    - time ./integral_dejong.x &
Multi-threaded Parallel - An example of OpenMP
Multi-threaded Parallel - An example of OpenMP

• Task Monitoring:

  ➢ Memory usage:

  1. `top -bcn1 -u <username>`

  2. `ps aux | awk 'NR==1 || $2==<pid> {print $0}'`

  3. `pmap -p <pid>`

  4. `./util/memu.sh pid=<pid>  (./util/memu.sh help; read /proc/<pid>/smaps)`

  5. All above commands are included in `cmd_list`
Multi-threaded Parallel - An example of OpenMP

- What are VIRT/VSZ and RES/RSS reported by top and ps?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>description</th>
<th>note</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIRT</td>
<td>VSZ</td>
<td>Total accessible address space of a process <em>(virtual memory)</em></td>
<td>May not be resident in RAM (i.e., not be used by a proc), e.g., heap allocated by malloc/new</td>
</tr>
<tr>
<td>RES</td>
<td>RSS</td>
<td>Total memory actually held in physical RAM for a process <em>(resident set size)</em></td>
<td>Could be overestimated when using a shared lib that is shared by multi procs</td>
</tr>
</tbody>
</table>

- What are Unique(Uss), Rss, and Pss reported by memu.sh?

  Unique = Private_Clean + Private_Dirty
  Shared = Shared_Clean + Shared_Dirty
  Rss (Resident set size) = Unique + Shared  
  Pss (Proportional set size) = Unique + Shared / Number of processes sharing the same memory
Multi-threaded Parallel - An example of OpenMPI

• Numerical summation using Kahan algorithm
  https://en.wikipedia.org/wiki/Kahan_summation_algorithm

• Demo folder on Sapelo2: /usr/local/training/CSP_seminar_09062022/demo_mpi/
  1. kahan_sum.cpp: sample code
  2. sub_mpi.sh: batch job submission script (sbatch sub_mpi.sh)
  3. util: folder storing memu.sh and nodes_run_job.sh
  4. cmd_list: list of commands
  5. makefile: compile code in an interactive session:
     - interact -p batch -c 1 --mem 4gb --constraint EDR
     - copy demo folder to your working folder
     - In your demo_mpi folder, run ml Boost/1.71.0-gompi-2019b && make
Multi-threaded Parallel - An example of OpenMPI

- System:
  - AMD Rome EDR nodes:
    #SBATCH --partition=batch
    #SBATCH --ntasks=200
    #SBATCH --constraint=Rome

- Task Monitoring:
  - Time: Elapsed time reported by sacct-gacrc -X

<table>
<thead>
<tr>
<th>MPI Ranks</th>
<th>Nodes</th>
<th>Complete Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>5</td>
<td>01:29:18 (5358 secs)</td>
</tr>
<tr>
<td>400</td>
<td>13</td>
<td>00:45:05 (2705 secs)</td>
</tr>
<tr>
<td>600</td>
<td>20</td>
<td>00:35:47 (2147 secs)</td>
</tr>
<tr>
<td>800</td>
<td>21</td>
<td>00:27:03 (1623 secs)</td>
</tr>
<tr>
<td>1200</td>
<td>30</td>
<td>00:18:15 (1095 secs)</td>
</tr>
<tr>
<td>1600</td>
<td>47</td>
<td>00:03:14 (194 secs)</td>
</tr>
</tbody>
</table>
Multi-threaded Parallel - An example of OpenMPI

- Check MPI procs running on node:
  1. `.util/nodes_run_job.sh <partition> <username>`
     `.util/nodes_run_job.sh batch zhuofei`
  2. `ssh <node> top -bcn1 -u <username>`
     `ssh d4-9 top -bcn1 -u zhuofei`
     `ssh d4-9 top -bcn1 -u zhuofei grep kahan_sum wc -l`
  3. `ssh <node> ps aux awk 'NR==1||$11~/<pattern>/ {print $0}'`
     `ssh d4-9 ps aux awk 'NR==1||$11~/kahan_sum/ {print $0}'`
     `ssh d4-9 ps aux awk '$11~/kahan_sum/ {print $0}' wc -l`
  4. `ssh <node> /full/path/to/memu.sh cmdfull`
     `ssh d4-9 /usr/local/training/CSP_seminar_09062022/demo_mpi/util/memu.sh cmdfull`
Multi-threaded Parallel - An example of OpenMPI

• Three sections in sub_mpi.sh:

```bash
# === foss ===
ml Boost/1.71.0-gompi-2019b
make clean
make
srun -n 200 ./kahan_sum.x 2400

# === MVAPICH2 ===
# ml Boost/1.74.0-gmvapich2-2019b
# make clean
# make
# srun --mpi=pmi2 -n 200 ./kahan_sum.x 2400

# === intel ===
# ml Boost/1.71.0-iimpi-2019b
# make clean
# make
# srun -n 200 ./kahan_sum.x 2400
```

You are welcome to uncomment intel and MVAPICH2 sections for test!
Thank You!

Georgia Advanced Computing Resource Center
101-108 Computing Services building
University of Georgia
Athens, GA 30602
GACRC Help: http://help.gacrc.uga.edu