HPC SLURM System Navigation and Execution Documentation

Table of Contents

Overview

This documentation provides comprehensive guidance for navigating and executing various parallel computing, distributed systems, and GPU computing programs in a SLURM-based HPC environment. The system includes implementations for OpenMP, MPI, shared memory programming, distributed deep learning, job scheduling, and GPU offloading.

System Prerequisites

Required Modules

Before executing any programs, load the necessary modules:

```bash
# Load basic modules
module load gcc/latest
module load openmpi/latest
module load cuda/latest
module load python/3.9
module load pytorch/latest
module load tensorflow/latest

# For OpenACC
module load pgi/latest
# or
module load nvhpc/latest

# Check loaded modules
module list
```

Environment Setup

```bash
# Set environment variables
export OMP_NUM_THREADS=8
export CUDA_VISIBLE_DEVICES=0,1,2,3
export NCCL_DEBUG=INFO
export PYTHONPATH=$PYTHONPATH:/path/to/your/project
```

Directory Structure

```
Project Root/
├── Demonstration_Mnist_Dist/   # MNIST distributed training demo
├── Dist_DL/                    # Distributed deep learning examples
├── Dist_Tensorflow/            # TensorFlow distributed computing
├── GPU_Off/                    # GPU offloading (CUDA & OpenACC)
├── job_sch_new/                # Advanced job scheduling
├── Job_Sch_Res_Man/            # Job scheduling & resource management
├── MPI/                        # MPI parallel programming
├── openmp_prog/                # OpenMP parallel programming
└── SMP/                        # Shared memory programming
```

Compilation Commands

1. C Programs (Standard)

```bash
# Basic C compilation
gcc -o program_name program_name.c

# With optimization
gcc -O3 -o program_name program_name.c

# With debugging
gcc -g -o program_name program_name.c
```

2. OpenMP Programs

```bash
```

```
# Compile OpenMP programs
gcc -fopenmp -o program_name program_name.c

# With optimization
gcc -fopenmp -O3 -o program_name program_name.c

# Example for openmp_prog directory
cd openmp_prog/plr
gcc -fopenmp -o plr plr.c

cd ../rsp
gcc -fopenmp -o rsp rsp.c

cd ../tms
gcc -fopenmp -o tmc tms.c
```

3. MPI Programs

```bash
# Compile MPI programs
mpicc -o program_name program_name.c

# With optimization
mpicc -O3 -o program_name program_name.c

# Examples for MPI directory
cd MPI/BSG
mpicc -o BSG BSG.c

cd ../COS
mpicc -o COS COS.c

cd ../P2PC
mpicc -o P2PC P2PC.c

cd ../TMS
mpicc -o TMS TMS.c
```

4. CUDA Programs (NVCC)

```bash
```

```bash
# Basic CUDA compilation
nvcc -o program_name program_name.cu

# With optimization and architecture specification
nvcc -O3 -arch=sm_70 -o program_name program_name.cu

# Examples for GPU_Off/CUDA_NVC
cd GPU_Off/CUDA_NVC/CGK
nvcc -o CGK CGK.cu

cd ../MGC
nvcc -o MGC MGC.cu
```

5. OpenACC Programs

```bash
# Using PGI/NVHPC compiler
pgcc -acc -ta=tesla -o program_name program_name.c

# Or using GCC with OpenACC
gcc -fopenacc -o program_name program_name.c

# Examples for GPU_Off/OpenACC
cd GPU_Off/OpenACC/DGP
pgcc -acc -ta=tesla -o DGP_gpu DGP.c
gcc -o DGP_cpu DGP.c  # CPU version

cd ../SLT
pgcc -acc -ta=tesla -o SLT SLT.c
```

6. Pthread Programs

```bash
# Compile with pthread library
gcc -pthread -o program_name program_name.c

# Example for SMP/Pthr
cd SMP/Pthr
gcc -pthread -o pth pth.c
```

SLURM Job Submission

Basic SLURM Script Template

```bash
#!/bin/bash
#SBATCH --job-name=job_name
#SBATCH --output=output_%j.out
#SBATCH --error=error_%j.err
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=8
#SBATCH --time=01:00:00
#SBATCH --partition=compute

# Load modules
module load gcc/latest

# Execute program
./program_name
```

Submit Jobs

```bash
# Submit job
sbatch job_script.slurm

# Check job status
squeue -u $USER

# Check job details
scontrol show job JOBID

# Cancel job
scancel JOBID
```

Module-Specific Instructions

1. OpenMP Programs (openmp_prog/)

Navigation and Execution:

```bash
```

```
cd openmp_prog/

# Parallel Loop (plr)
cd plr/
gcc -fopenmp -o plr plr.c
export OMP_NUM_THREADS=4
./plr

# Race Condition Prevention (rsp)
cd ../rsp/
gcc -fopenmp -o rsp rsp.c
./rsp

# Thread Management System (tms)
cd ../tms/
gcc -fopenmp -o tmc tms.c
./tmc
```

SLURM Script for OpenMP:

```bash
#!/bin/bash
#SBATCH --job-name=openmp_job
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --time=00:30:00

export OMP_NUM_THREADS=8
./program_name
```

2. MPI Programs (MPI/)

Navigation and Execution:

```bash
```

```
cd MPI/

# Basic Send-Get (BSG)
cd BSG/
mpicc -o BSG BSG.c
mpirun -np 4 ./BSG

# Collective Operations (COS)
cd ../COS/
mpicc -o COS COS.c
mpirun -np 4 ./COS

# Point-to-Point Communication (P2PC)
cd ../P2PC/
mpicc -o P2PC P2PC.c
mpirun -np 2 ./P2PC
```

SLURM Script for MPI:

```bash
#!/bin/bash
#SBATCH --job-name=mpi_job
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --time=00:30:00

mpirun ./program_name
```

3. GPU Offloading (GPU_Off/)

CUDA Programs:

```bash
```

```
cd GPU_Off/CUDA_NVC/

# CUDA Gaussian Kernel (CGK)
cd CGK/
nvcc -arch=sm_70 -o CGK CGK.cu
./CGK

# Multi-GPU Computing (MGC)
cd ../MGC/
nvcc -arch=sm_70 -o MGC MGC.cu
./MGC
```

OpenACC Programs:

```bash
cd GPU_Off/OpenACC/

# Data GPU Processing (DGP)
cd DGP/
pgcc -acc -ta=tesla -o DGP_gpu DGP.c
gcc -o DGP_cpu DGP.c
./DGP_gpu  # GPU version
./DGP_cpu  # CPU version

# Stencil Loop Tiling (SLT)
cd ../SLT/
pgcc -acc -ta=tesla -o SLT SLT.c
./SLT
```

SLURM Script for GPU:

```bash
#!/bin/bash
#SBATCH --job-name=gpu_job
#SBATCH --nodes=1
#SBATCH --gres=gpu:2
#SBATCH --time=01:00:00

export CUDA_VISIBLE_DEVICES=0,1
./program_name
```

4. Distributed Deep Learning (Dist_DL/)

Model Sharding:

```bash
cd Dist_DL/model_shard/
sbatch run_nccl.slurm
# Check outputs: nccl_*.out and nccl_*.err files
```

Multi-GPU and Multi-Node:

```bash
cd Dist_DL/multi_gpu_and_multi_node/

# NCCL backend
sbatch run_nccl.slurm

# Gloo backend
sbatch run_gloo.slurm
```

Synchronous vs Asynchronous:

```bash
cd Dist_DL/sync_vs_async/
sbatch sy_vs_asy.slurm
```

5. Job Scheduling and Resource Management (Job_Sch_Res_Man/)

Basic SLURM Usage:

```bash
cd Job_Sch_Res_Man/bas_slu_usa/

# Single node job
cd req_cpu_gpu_mem/sing_nod/
sbatch single_node.slurm

# Multiple node job
cd ../mult_nod/
sbatch multi_node.slurm

# Shell command execution
cd ../../run_shel/
sbatch shell_comm.slurm
```

Python Script Submission:

```bash
cd Job_Sch_Res_Man/run_py_scr/sub_py_train_sbatch/
sbatch pytorch_sbatch.slurm
```

6. Shared Memory Programming (SMP/)

Pthread Programming:

```bash
cd SMP/Pthr/
gcc -pthread -o pth pth.c
./pth
```

Python Multiprocessing:

```bash
cd SMP/py_mul/
python multi_proc.py
```

7. MNIST Distributed Demonstration

```bash
cd Demonstration_Mnist_Dist/
sbatch py_train.slurm
# Monitor outputs: nccl_*.out and nccl_*.err files
```

8. TensorFlow Distributed Computing

```bash
cd Dist_Tensorflow/
sbatch py_train.slurm
# or
sbatch train.slurm
```

Advanced Job Scheduling (job_sch_new/)

```bash
```

```bash
cd job_sch_new/

# Submit advanced scheduling job
sbatch job_sched1.slurm

# Fine-tuning job
sbatch slurm_ft.slurm

# Run local script
./run_local.sh

# Communication job
./slurm_comm.sh
```

Monitoring and Debugging

Check Job Status:

```bash
# View queue
squeue

# View your jobs
squeue -u $USER

# Detailed job info
scontrol show job JOBID

# Job accounting info
sacct -j JOBID --format=JobID,JobName,MaxRSS,Elapsed
```

View Output Files:

```bash
# Real-time monitoring
tail -f output_file.out
tail -f error_file.err

# View completed job outputs
cat slurm-JOBID.out
cat JOBNAME_JOBID.err
```

Resource Usage:

```bash
# Check node information
sinfo

# Check available partitions
sinfo -s

# Check node details
scontrol show nodes
```

Troubleshooting

Common Issues and Solutions:

1. **Module Not Found Error:**

```bash
module avail  # Check available modules
module load module_name
```

2. **Compilation Errors:**

```bash
# Check if all dependencies are loaded
which gcc
which mpicc
which nvcc
```

3. **CUDA Out of Memory:**

```bash
# Check GPU status
nvidia-smi
# Reduce batch size or use gradient accumulation
```

4. **MPI Communication Errors:**

```bash
# Check network connectivity
mpirun -np 2 hostname
```

5. **SLURM Job Failures:**

```bash
```

```
# Check job details
scontrol show job JOBID
# View error logs
cat error_JOBID.err
```

Best Practices

1. Resource Management:

- Always specify resource requirements accurately
- Use appropriate partition for your job type
- Set reasonable time limits
- Clean up temporary files after job completion

2. Code Organization:

- Keep source code and executables separate
- Use meaningful job names and output file names
- Implement proper error handling in your programs
- Comment your SLURM scripts for clarity

3. Performance Optimization:

- Profile your applications before scaling up
- Use appropriate compiler optimization flags
- Balance CPU, memory, and GPU resources
- Monitor resource utilization during execution

4. Data Management:

- Use appropriate data storage locations (scratch space vs home directory)
- Implement checkpointing for long-running jobs
- Clean up intermediate data files
- Use efficient I/O patterns

5. Security and Collaboration:

- Set appropriate file permissions
- Use version control for your code
- Document your experiments and results
- Share resources responsibly in multi-user environment

Quick Reference Commands

```bash
# Compilation Quick Reference
gcc -fopenmp -O3 -o program program.c        # OpenMP
mpicc -O3 -o program program.c               # MPI
nvcc -arch=sm_70 -O3 -o program program.cu   # CUDA
pgcc -acc -ta=tesla -o program program.c     # OpenACC
gcc -pthread -o program program.c            # Pthread

# SLURM Quick Reference
sbatch script.slurm    # Submit job
squeue -u $USER        # Check your jobs
scancel JOBID          # Cancel job
sinfo                  # Check nodes
sacct -j JOBID         # Job accounting
```

This documentation provides a comprehensive guide for navigating and executing programs in your HPC SLURM environment. Adjust the specific parameters (node counts, time limits, etc.) based on your system's configuration and job requirements.