SUSTech Supercomputing Team

Our Team

Yingwei Zheng

LINPACK & miniVite

CG / Compiler

Bingzhen Wang

miniVite & maintenance

CG / PL

Jixiao Zhang

NAMD & Mystery

Graph Neural Network

Tingzhen Dong

NAMD & Mystery

Security / Arch

Jia'nan Zhu

NAMD & LINPACK

Security / Arch

Junfeng Chen

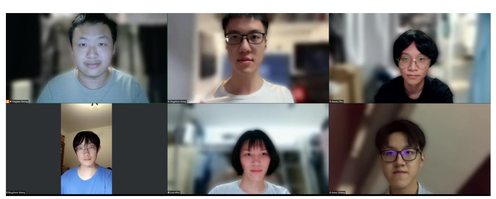
miniVite & maintenance

Full Stack

We, SUSTech Supercomputing Team (SST), are sponsored by the Center for Computational Science and Technology (CCSE) at SUSTech, advised by Dr. Fan Jing, the chief director of CCSE. The center is equipped with multipurpose supercomputing platforms, including NVIDIA Ampere, Intel Cascade Lake, Knights Mill, AMD EPYC Zen 2, and FPGAs. They are all open for our team to practice skills in real scenes virtually without any limitation.

Meanwhile, we are always seeking opportunities to improve our diversity. Team members master different skillsets about HPC, such as compiler optimization, CPU microarchitecture, large-scale software development and cloud management. Moreover, we are privileged to have Mx. Jia'nan Zhu and Miss Junfeng Chen to join us, who are both taking honors courses in the Turing Class of the Department of Computer Science and Engineering. Our team consists of 3 members with experience in supercomputing competitions and 3 members with outstanding performance in ICPC contests and CTF events.

Currently, SST has members and consultants majoring in various disciplines. Some of them come from minority Autonomous Regions, such as Guangxi Zhuang Autonomous Region and Inner Mongolia Autonomous Region. Moreover, we encourage females and non-binary to join us and strictly prohibit any form of discrimination.



SUSTech SC22 IndySCC team

Optimization

General Approaches

- Stage 0: **Profiling**
  - Locate the hotspots
- Stage 1: **System-level Tuning**
  - Override the default allocator with a faster **NUMA-aware/O(1)** allocator
  - Use different MPI libraries (Intel MPI/HPCX)
  - Tune runtime MPI & OpenMP parameters (ppn, threads per process, CPU affinity, etc.) for **best scaling**
- Stage 2: **Application-level Tuning**
  - Reduce redundant **computation/memory copy/communication**
  - **Overlap** communication and computation via non-blocking MPI calls
  - Tune the runtime parameters of applications (**decomposition/load balance**)
- Stage 3: **Architectural & Microarchitectural Tuning**
  - Tune compiler optimization parameters
  - Provide some hints to the compiler for lower **branch misprediction** & better **vectorized** code
  - Improve **cache/page performance** for some data access patterns (data alignment, AOS -> SOA/AOSOA)
  - Use some **expensive methods** (searching/ML) to generate better scheduled code
  - Run profile-guided optimization (PGO) and post-link optimizer to improve the **memory locality of code**

LINPACK

- Tune the **input parameters** (e.g., problem size) to achieve maximum performance according to the hardware configuration

NAMD

- Eliminate the unused **restart file** to reduce the communication and disk I/O
- Tune **patch-splitting** to fully utilize all CPU cores
- Tune **PME pencils** for each input and each node count to achieve best scaling
- Run **Charm++ performance tracing** to analyze the MPI performance

miniVite

- Use different MPI **communication methods** (NBSR/COLL/SR/RMA)
- Improve the **load balance** to reduce the overall communication and achieve better performance
- **Parallelize** some sequential for-loops with TODO comments

Mystery Application

- **Deploy** the application with different compilers and libraries using Spack
- **Reserve resources** for the mystery application according to the time budget/possible power limitation
- **Analyze** the scalability and computation/communication-intensity

Software Configuration

- Package Manager: Spack & Conda & vcpkg
- Workload Manager: Slurm
- Compiler:
  - Intel® oneAPI C++ Compiler
  - Intel® Implicit SPMD Program Compiler for **kernel accelerating**
  - Latest release of clang & polly & bolt
  - Some open source LLVM-based compilers for vectorization presented in **recent conference papers** (CGO, PLDI, etc.)
- Profiler:
  - Intel® VTune™ Profiler
  - Intel® Advisor
  - Intel® Trace Analyzer and Collector for MPI profiling
  - Tracy for the visualization of applications' behavior
  - Linux perf for collecting **kernel/hardware(Performance Monitoring Unit) events**
- MPI: Intel® MPI Library & HPCX
- BLAS Library: Intel® oneAPI Math Kernel Library
- Power Management: ipmitool

Preparation

We currently evaluate the benchmarks and applications using the "Tai-Yi" and "Qi-Ming" HPC clusters provided by the Center for Computational Science and Technology (CCSE) at SUSTech that roughly matches the hardware provided by IndySCC. All team members are required to do the following:

- Read the related papers and understand the performance/quality metrics
- Collect common datasets/inputs described in papers/readmes
- Deploy the benchmarks & applications to the given clusters
- Apply the optimization schemes and evaluate the results
- Prepare the out-of-the-box scripts for deploying/auto-tunning/job-submitting/visualization

The "Tai-Yi" HPC Cluster

The "Qi-Ming" HPC Cluster