

APAC26

We are SUSTech Scout Regiment, a passionate team from Southern University of Science and Technology,

Our diverse team brings together students from various academic years and disciplines, combining our strengths to tackle high-performance optimization and learn from real supercomputing challenges.

In this competition, we used two clusters.

Our CPU experiments were conducted on the Qiming 2.0 cluster, equipped with Intel® Xeon® Gold 6138 CPUs and a Lustre parallel file system. To take the hardware as close as the Gadi and NSCC platform We conducted our NWChem workloads on up to 4 nodes (160 CPU cores) to evaluate scalability and optimize performance.

For our GPU Deepseek-R1 workload, Our performance is achieved on the organizers' H200 reference cluster. We're thankful for the access to this infrastructure, which allows our software optimizations to truly shine.

NWChem

Now, let's first introduce our NWChem Optimizations

"NWChem is a scalable, open-source computational chemistry package optimized for supercomputers. It's widely used for quantum chemistry and molecular dynamics." SCF (Self-Consistent Field) workloads solve the electronic wavefunction, DFT (Density Functional Theory) workloads extend this with grid-based numerical integrations .

Both are communication-intensive and rely heavily on MPI and Global Arrays for distributed computation.

Workload

For workload, NWChem holds a Global array. During computation in SCF and DFT, it will fetch piece of array to compute, then write back to array, finishing 1 iterations. The arrays is built on Global array, running on MPI functions stacks

Here we run the baseline of nwchem using intel mpi and openblas. We use wall time as our evaluation metrics . Results shows the correct outputs with an observations that: they didn't scale well for 4 nodes.

Here we use VTune to explain why: the result lies in 2 part: Communication & computation.

So for 28% the time, in communication and they are caused by mpi barrier or MPI_Iprobe. And we guess that this is because load imbalance.

The second thing is computation. There are many operators implemented in NWChem. They are memory bound and we can only optimize them with compiler and I/O improvement.

The third things, obviously is from BLAS function. Like Double General Matrix mulpl, which may optimize by BLAS.

And we also do IPM profiling for MPI, we can see that most of the processes are doing MPI_Iprobe, it's asking that whether a data has come. So we can see most of the time the processes are waiting for data.

Due to these observations, we proposed 5 optimization to optim the bottleneck. For I/O, Compile and BLAS, we change BLAS & COmpiler. We also tried OpenMP+MPI

For com barrier cause by load imbalance, we tried to select best MPI, and select best strategy like MPI-PR. Also choosing the best mem

Deepseek

parallelism

NCCL

Backend

Spedcl recodafsd