MIMICA code performance analysis report

Wei Zhang

NSC, SNIC

1 Project Description

This report aims to evaluate MIMICA code performance, pinpoint performance bottleneck, and propose performance-enhancement suggestions.

2 Time Measurements

Experiment platform: Triolith cluster at NSC, Intel Fortran compiler 15.0.1, Intel MPI library impi/5.0.2.044, optimization flag: -O2, two problem sizes:

- P1: $dt = 2.0$, $nstart = 1$, $nstop = 1000$, $dx = dy = 62.5$, $dz = 25.0$
- P2: $dt = 2.0$, $nstart = 1$, $nstop = 1000$, $dx = dy = 31.25$, $dz = 50.0$

The time measurements (provided by mpprun) are listed in Table 1.

Table 1: P1, P2 time measurements (seconds)

Cores	Time (P1)	$\text{Time}_{8-core}/\text{Time}$	Time (P2)	$\text{Time}_{8-core}/\text{Time}$
8 (1 node)	1412	1.00	4653	1.00
16 (1 node)	1239	1.13	4233	1.09
32 (2 nodes)	916	1.54	2818	1.65
64 (4 nodes)	677	2.08	2057	2.26
128 (8 nodes)	611	2.31	1758	2.64

3 Performance Analysis

Table 1 shows MIMICA has poor scalability. Profilers Allinea and Scalasca are used to analyse the problem.

3.1 Allinea

The Allinea profiling results of 128 cores are shown in Figures 1 and 2. All results from 8 cores to 128 cores are plotted in Figure 3.

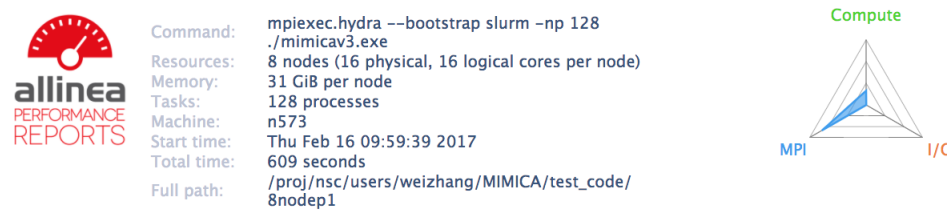

Figure 1: P1 time breakdown provided by Allinea (128 cores)

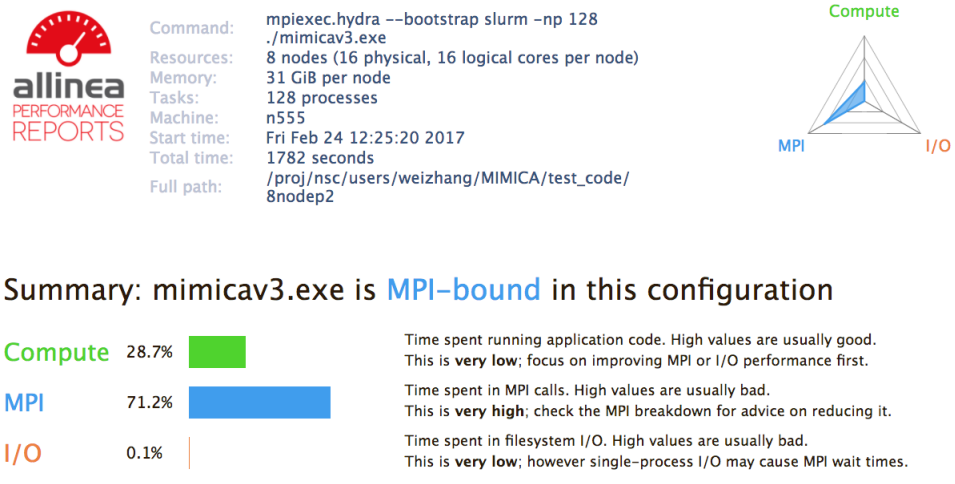

Figure 2: P2 time breakdown provided by Allinea (128 cores)

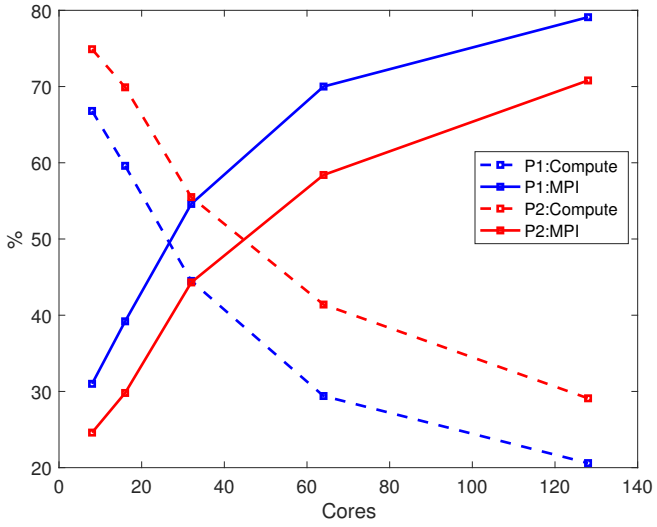

Figure 3: P1, P2 time percentage of compute and MPI communication.

The portion of MPI communication increases dramatically, which leads to poor scalability.

3.2 Scalasca

Figures 4 and 5 show Scalasca profiling results.

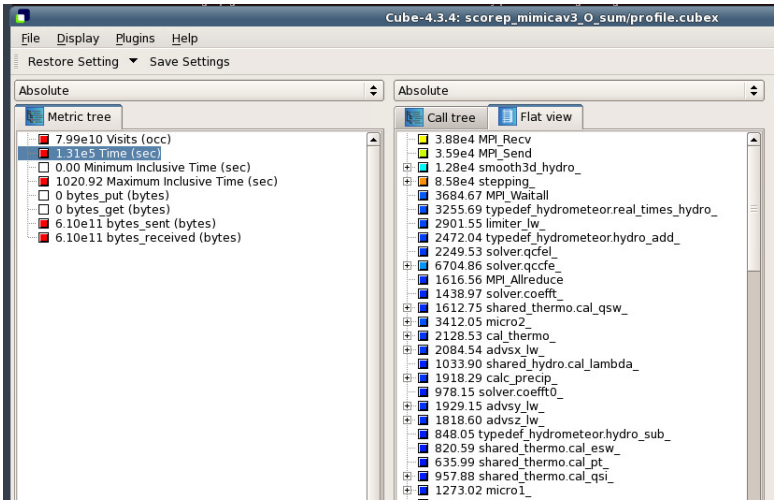

Figure 4: P1 time breakdown provided by Scalasca (128 cores)

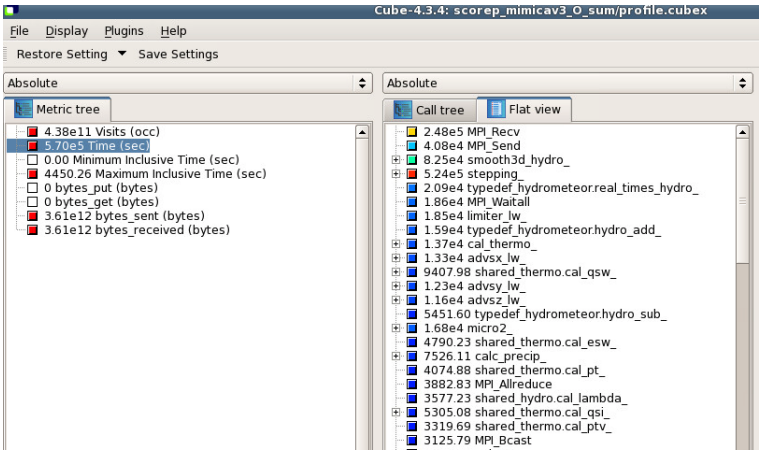

Figure 5: P2 time breakdown provided by Scalasca (128 cores)

They show MPI point-to-point communication is the main time-consuming part, which can be found in e.g., subroutine collect_i1d_sp:

```
if ( mypid .ne. 0 ) then
      ...
          call MPI_SEND (tmp(1,1,1), tot, REALTYPE, 0, mypid, MPI_COMM_WORLD, ierr )
 else
      ...
          do i = 0, nprocx-1
           do j = 0, nprocy-1
             if (i+j /= 0) &
              call MPI_IRECV (rtmp(1,1,1,i*nprocy+j),tot, REALTYPE,  &
                       i*nprocy+j, i*nprocy+j, MPI_COMM_WORLD, req(i*nprocy+j), ierr )
```

Similar inefficient point-to-point communication is found in subroutine, e.g. distribute_i1d_sp.

4 Conclusion and Suggestion

The main problem of MIMICA is scalability, caused by inefficient MPI communication. One suggestion for performance-enhancement:

- Replacing MPI point-to-point communication by MPI_GATHER. This is a simple and quick modification. However, the performance needs further investigation.