MPI Performance Analysis Tools

The performance analysis tools help identify the regions that are underperforming. Those tools can pinpoint the problem locations, allowing the program to run more efficiently. The instruments for analysis should be reliable and portable.

There are several tools available, and I chose,

1. VAMPIR
2. DEEP/MPI

VAMPIR

- The tool was developed by Pallas GmbH
- Vampirtrace is a profiling library, and it produces trace files, and the analysis is handled using trace files.
- It has an API for starting and stopping tracing and also inserts the trace file with user-defined events.
- It includes MPI I/O calls and a runtime filtering mechanism which is used to limit the amount of trace data.
- Vampitrace automatically corrects clock offset and skew, for systems without globally consistent clocks.
- The current version can display up to 512 processes and the display is clustered with cluster, node and process with a maximum of 200 objects.
- The displays include analysis of program execution, statistical analysis and dynamic calling tree.

DEEP/MPI

- Veridian-Pacific Sierra Research created the DEEP (Development Environment for Parallel Programs).
- The mpiprof DEEP profiling driver must be used to compile the program.
- It has an integrated graphics interface through which the shared and distributed memory programs can be monitored.
- It saves the data in a subfolder and shows the performance statistics after running the code, as well as the current time on the wall clock.
- One of the extra pieces of information that the user puts up is the loop performance table.
- The CPU and message balance graph shows how work is distributed and how many messages are sent between processes.
- PAPI Interface to hardware was supported and it supports the profiling.
- Supports analysis of shared memory parallelism and can be used to analyze mixed MPI and shared memory parallel programs.

Differences:

- The profiling library used for Vampir is Vampitrace and for DEEP/MPI, mpiprof is used.
- DEEP has an integrated graphics interface which fetches data from shared memory and displays data for only a small number of processes while Vampir uses a cluster mechanism and displays status of lots of processes.
- Vampir traces user-defined events while DEEP does not support this functionality.
- Vampitrace automatically corrects the clock and makes it consistent while DEEP doesn't.
- DEEP includes a call tree viewer for program structure browsing while Vampitrace uses a dynamic calling tree.