Performance and Code Analysis Tools for HPC

- Intel Performance Tools
- Information on Hardware and Topology
- Timing and Profiling
- Hardware Performance Counters
- MPI, OpenMP, Parallelization, Vectorization, SIMD Analysis
- Memory Leaks
- Optimization for Energy

For optimization and tuning strategies see:

- Tuning and Optimization for HPC

Intel Performance Tools

Tools for applications that are compiled with the Intel compilers and/or linked with Intel MPI are available:

- Intel Advisor: A tool to guide parallelization
- Intel Inspector: Assuring Correctness of scientific codes
- Intel Tracing Tools: Profiling and Correctness checking of MPI programs
- Intel VTune Amplifier and Performance Snapshots

Information on Hardware and Topology

HWLoc

The hardware locality toolset provides command line tools as well as a programming interface for identifying and controlling resources and resource mappings for threaded execution.

likwid-topology

Modern computers get more and more complicated. They consist of multiple cores and each core can support multiple hardware threads. Because cores share caches and main memory access it is important to pin threads to dedicated cores. To decide this it is important to know a machines topology. likwid-topology extracts this information from the cpuid instruction.

Timing and Profiling

Timing commands and Timing functions

Timers can be used to measure the total run time of an application. Different implementations are available on the UNIX and Linux systems. Some subroutines are also available to be called within your code to measure specific sections.

gprof

gprof calculates the amount of time spent in each routine. The effect of called routines is incorporated in the profile of each caller. The profile data is taken from the call graph profile file which is produced by compiling/linking the executable with -pg.

Profile Guided Optimization

Main purpose of profile guided optimization is to re-order instructions in an optimal way. The instrumented executable is run one or more times with different typical data sets. The dynamic profiling information is merged, and the combined information is used to generate a profile-optimized executable.

Hardware Performance Counters

Note: Switch off the LRZ PerSyst monitoring system to avoid conflict with other tools that measure performance. Call persyst_off before doing the measurements.

Intel VTune Amplifier
The Intel Amplifier (formerly VTune) analyzer collects, analyzes, and displays hardware performance data from the system-wide view down to a specific function, module, or instruction.

**LIKWID**

Likwid (Like I knew what I am doing) provides easy to use command line tools for Linux to support programmers in developing high performance multi threaded programs.

**PAPI**

PAPI (Performance Application Programming Interface) aims to provide the tool designer and application engineer with a consistent interface and methodology for use of the performance counter hardware found in most major microprocessors. PAPI enables software engineers to see, in near real time, the relation between software performance and processor events.

**PerSyst**

Performance properties are collected by the PerSyst Monitoring system at LRZ. Using the web API or and the console command line tool, user friendly Interfaces are presented. In this case there is no need to use the persyst_off command.

### MPI, OpenMP, Parallelization, Vectorization, SIMD Analysis

**Intel Application Performance Snapshot**

Intel Application Performance Snapshot provides a quick look at your application's performance like MPI parallelism, OpenMP* parallelism, Memory access, FPU Utilization, I/O efficiency

**Intel Tracing Tools**

The Intel Tracing Tools support the development and tuning of programs parallelized using MPI. By using these tools you are able to investigate the communication structure of your parallel program, and hence to isolate incorrect and/or inefficient MPI programming. The

- Trace Collector is a set of MPI tracing libraries, and the
- Trace Analyzer provides a GUI for analysis of the tracing data.

**Intel Inspector**

Inspector allows you to perform correctness checking on multi-threaded applications (running in shared memory).

**Intel Amplifier**

Intel Amplifier (formerly VTune) allows you to perform performance analysis on multi-threaded applications (running in shared memory). The analyzer also collects, analyzes, and displays hardware performance data from the system-wide view down to a specific function, module, or instruction.

**Intel Advisor**

Advisor allows you to identify optimization potential in your code (both multi-threading and SIMD vectorization)

**Vampir NG**

Vampir from TU Dresden is the State-Of-The-Art Tool for tracing parallel programs based on MPI, OpenMP or CUDA, and serial programs. It is designed to provide accurate trace information of MPI and user function calls. The user interface and parallel processing of tracing data makes Vampir NG the most powerful tool for tracing. It includes the capability for performance-counter analysis based on PAPI.

**Scalasca**

Scalasca is an open-source project developed by the Jülich Supercomputing Centre which focuses on analyzing OpenMP, MPI and hybrid OpenMP/MPI parallel applications. Scalasca can be used to help identify bottlenecks by providing a number of important features: profiling and tracing of highly parallel programs; automated trace analysis that localizes and quantifies communication and synchronization inefficiencies; flexibility and integration with

**Marmot**

Marmot is a MPI correct checker. It automatically checks the correct usage of MPI functions and their arguments. It can identify deadlocks, wrong ordering of messages, wrong MPI types, etc.

**Guideview**

GuideView is a tool that displays the performance details of an OpenMP program's parallel execution.

### Memory Leaks

**MemoryScape**

This tool provides a subset of Totalview functionality to detect memory leaks.

**Valgrind**

Valgrind
For finding memory leaks, measuring memory consumption as well as identifying performance bottlenecks.

**Optimization for Energy**