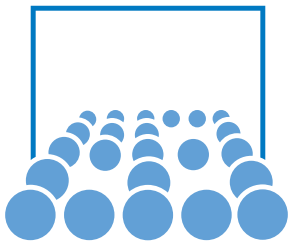
Lab: Scientific Computing – Tsunami-Simulation

Session 4: Linux-Cluster, VTune, OpenMP

A. Breuer, S. Rettenberger, M. Bader

27th May, 2013



Outline

Presentations: Assignment 3

Linux-Cluster

VTune

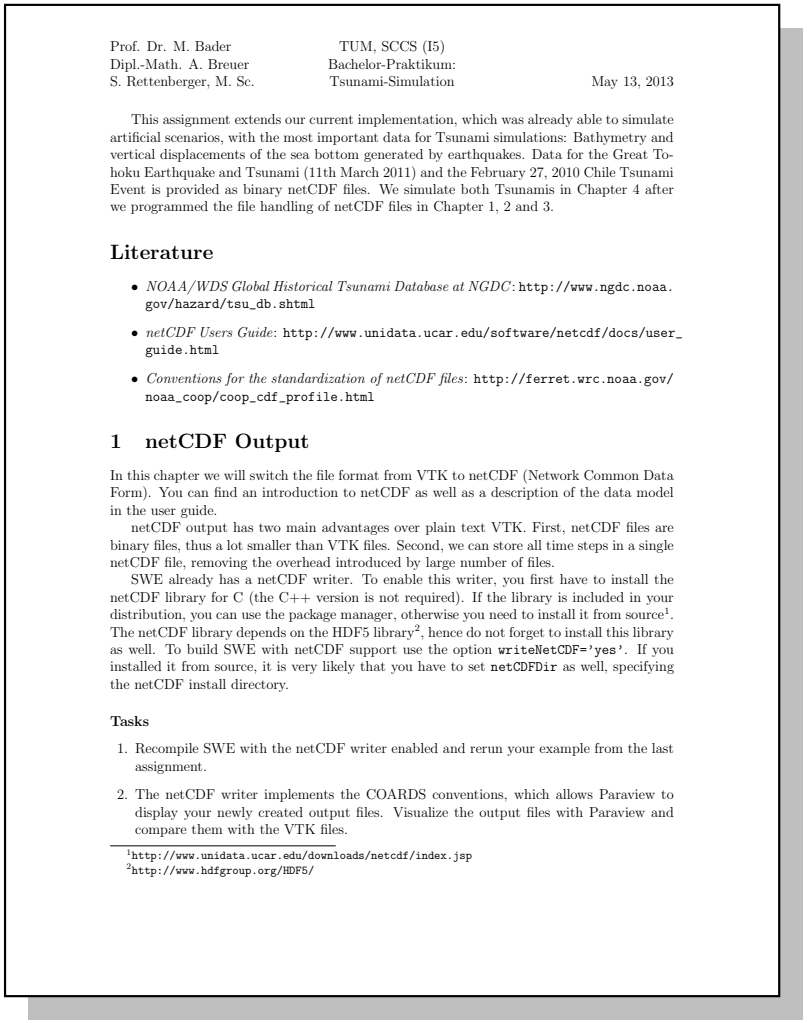
OpenMP

Preparation: Assignment 4

Schedule (big meetings)

Date	Schedule
15.04.2013	Kickoff
29.04.2013	Presentation 1
13.05.2013	Presentation 2
27.05.2013	Presentation 3
10.06.2013	Presentation 4
24.06.2013	Report: Project Phase
01.07.2013	Presentation Project

Presentations: Assignment 3



Prof. Dr. M. Bader — TUM, SCCS (I5)
Dipl.-Math. A. Breuer — Bachelor-Praktikum:
S. Rettenberger, M. Sc. — Tsunami-Simulation — May 13, 2013

This assignment extends our current implementation, which was already able to simulate artificial scenarios, with the most important data for Tsunami simulations: Bathymetry and vertical displacements of the sea bottom generated by earthquakes. Data for the Great Tohoku Earthquake and Tsunami (11th March 2011) and the February 27, 2010 Chile Tsunami Event is provided as binary netCDF files. We simulate both Tsunamis in Chapter 4 after we programmed the file handling of netCDF files in Chapter 1, 2 and 3.

Literature

- *NOAA/WDS Global Historical Tsunami Database at NGDC*: `http://www.ngdc.noaa.gov/hazard/tsu_db.shtml`
- *netCDF Users Guide*: `http://www.unidata.ucar.edu/software/netcdf/docs/user_guide.html`
- *Conventions for the standardization of netCDF files*: `http://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html`

1 netCDF Output

In this chapter we will switch the file format from VTK to netCDF (Network Common Data Form). You can find an introduction to netCDF as well as a description of the data model in the user guide.

netCDF output has two main advantages over plain text VTK. First, netCDF files are binary files, thus a lot smaller than VTK files. Second, we can store all time steps in a single netCDF file, removing the overhead introduced by large number of files.

SWE already has a netCDF writer. To enable this writer, you first have to install the netCDF library for C (the C++ version is not required). If the library is included in your distribution, you can use the package manager, otherwise you need to install it from source[1]. The netCDF library depends on the HDF5 library[2], hence do not forget to install this library as well. To build SWE with netCDF support use the option `writeNetCDF='yes'`. If you installed it from source, it is very likely that you have to set `netCDFDir` as well, specifying the netCDF install directory.

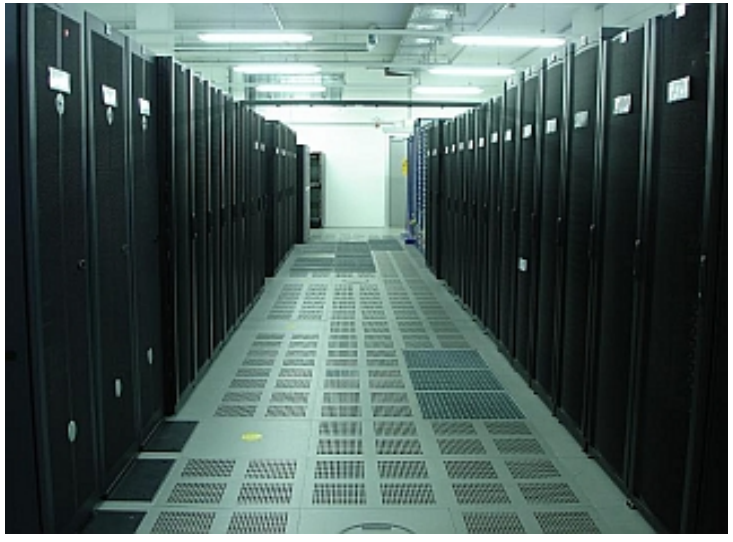
Tasks

1. Recompile SWE with the netCDF writer enabled and rerun your example from the last assignment.
2. The netCDF writer implements the COARDS conventions, which allows Paraview to display your newly created output files. Visualize the output files with Paraview and compare them with the VTK files.

[1] `http://www.unidata.ucar.edu/downloads/netcdf/index.jsp`
[2] `http://www.hdfgroup.org/HDF5/`

Linux-Cluster

1999



2012

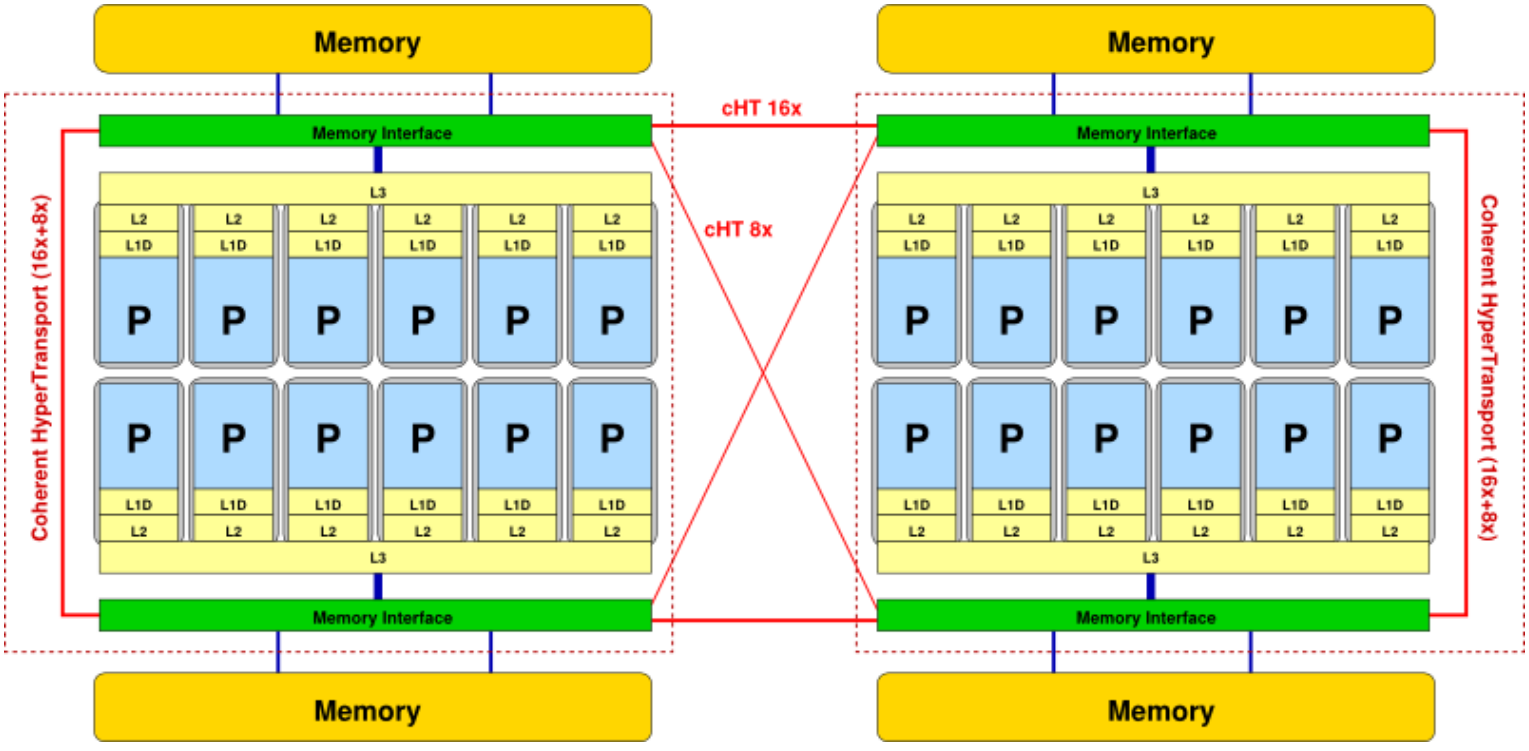
https://www.lrz.de/services/compute/linux-cluster/lx_timeline/

Hardware

Segment	Number of nodes	Number of processor cores	aggregate peak performance (TFlop/s)	aggregate memory (TByte)
ICE	64	512	5.2	1.5
MPP	178	2848	22.7	2.8
UV	2	2080	20.0	6.0

`https://www.lrz.de/services/compute/linux-cluster/overview/`

AMD Opteron “Magny Cours”



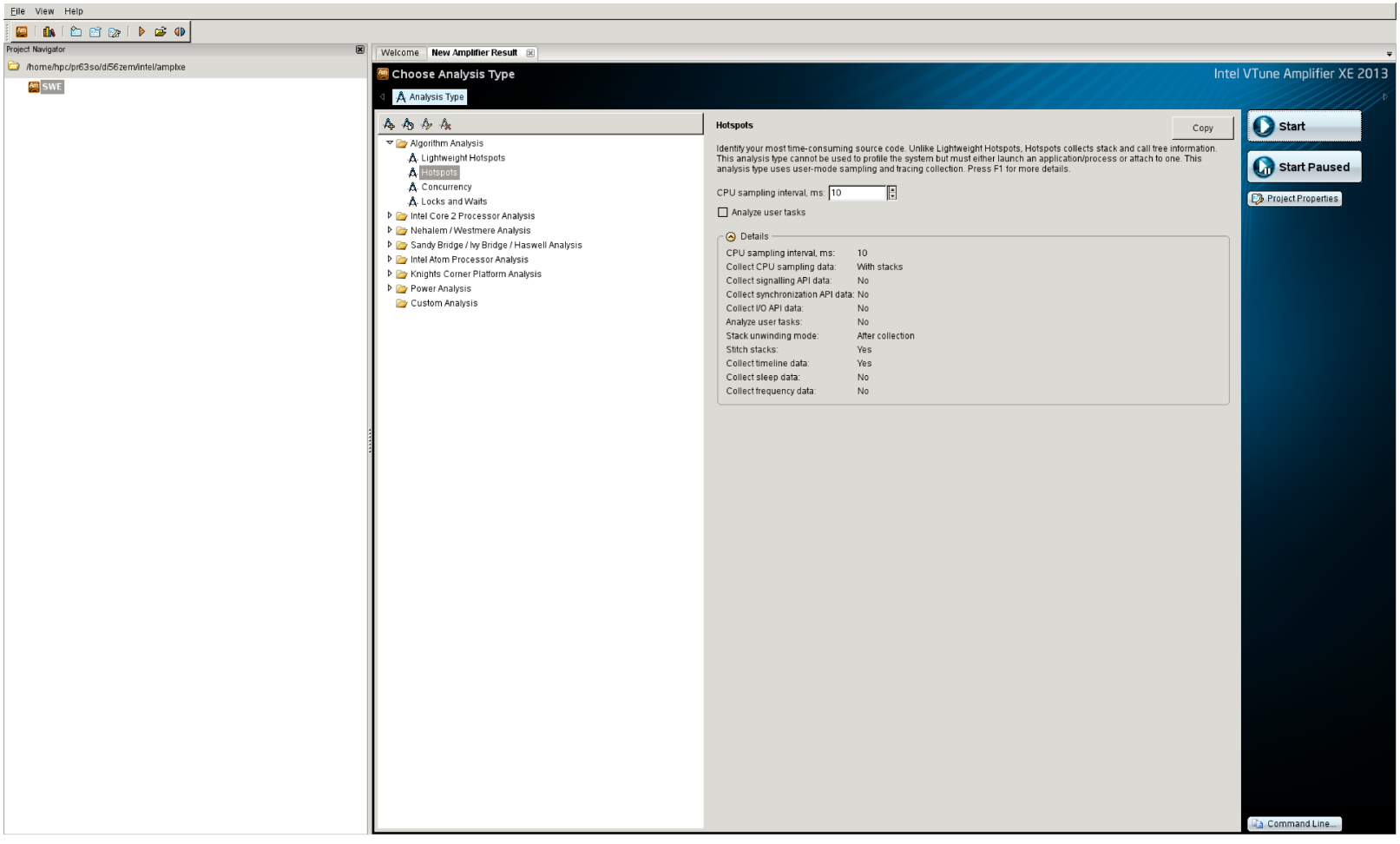
Dual socket, 12 core
(Linux-Cluster: Dual socket, octa core)

http://www.rrze.uni-erlangen.de/dienste/arbeiten-rechnen/hpc/systeme/memoryhog.shtml

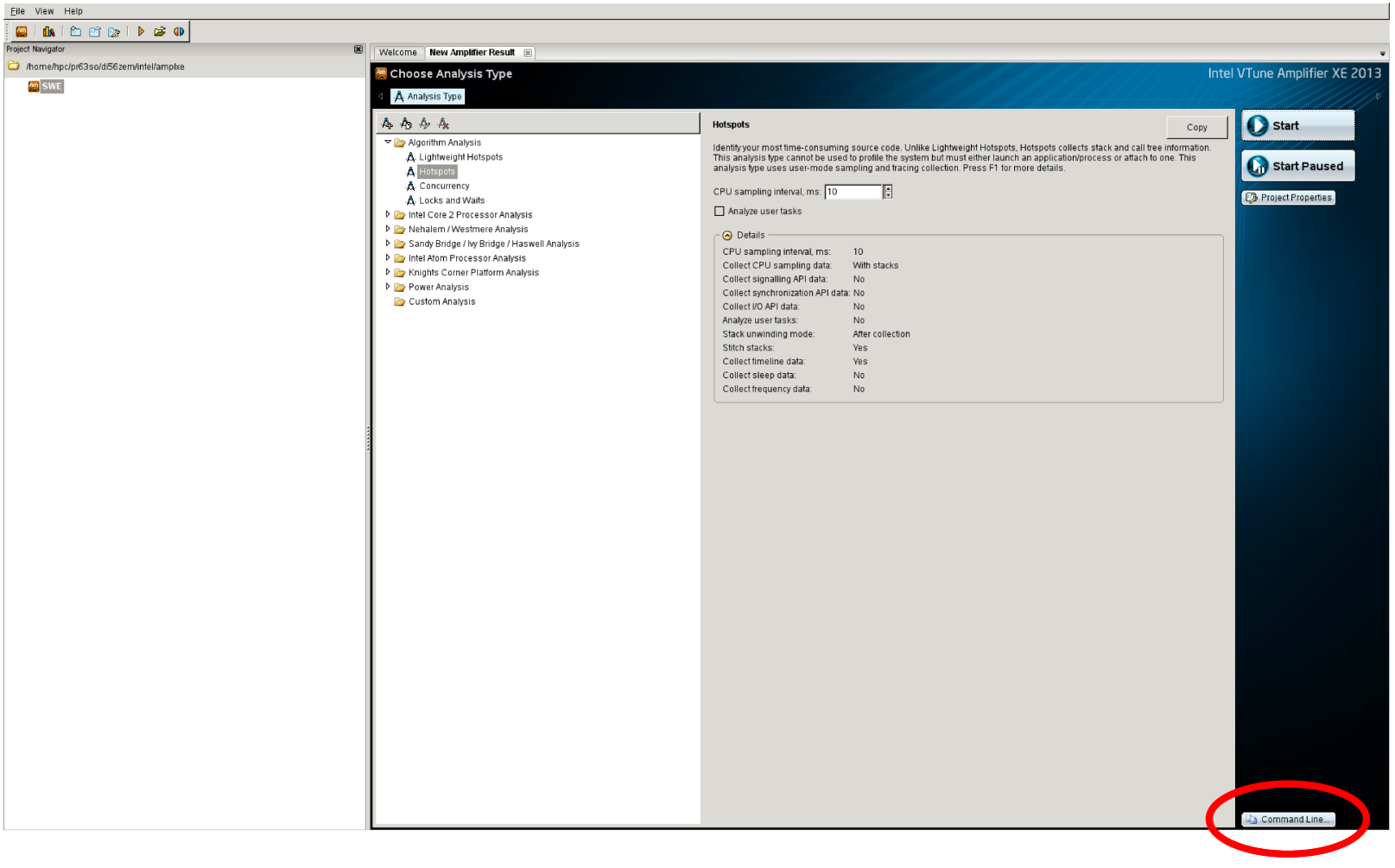
Rules

1. Do not use login nodes to run your application.
2. Do not start more than one interactive job.
3. Do not put large input/output files in your home directory.

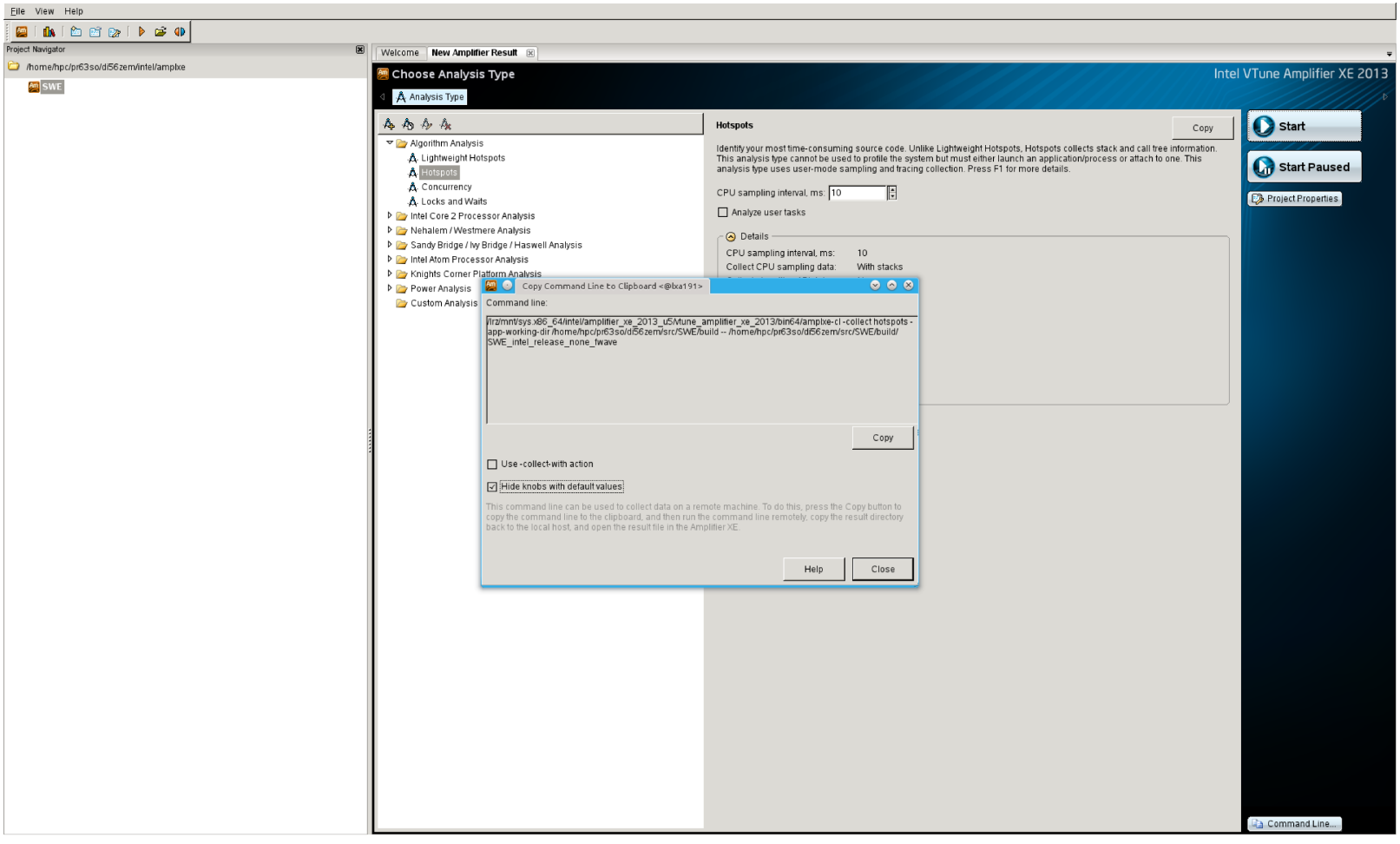
VTune


File View Help
Project Navigator
/home/hpc/pr63so/di56zem/intel/amplxe
SWE
Welcome
New Amplifier Result
Choose Analysis Type
Intel VTune Amplifier XE 2013
Analysis Type
Algorithm Analysis
Lightweight Hotspots
Hotspots
Concurrency
Locks and Waits
Intel Core 2 Processor Analysis
Nehalem / Westmere Analysis
Sandy Bridge / Ivy Bridge / Haswell Analysis
Intel Atom Processor Analysis
Knights Corner Platform Analysis
Power Analysis
Custom Analysis
Hotspots
Copy
Identify your most time-consuming source code. Unlike Lightweight Hotspots, Hotspots collects stack and call tree information. This analysis type cannot be used to profile the system but must either launch an application/process or attach to one. This analysis type uses user-mode sampling and tracing collection. Press F1 for more details.
CPU sampling interval, ms: 10
Analyze user tasks
Details
CPU sampling interval, ms: 10
Collect CPU sampling data: With stacks
Collect signalling API data: No
Collect synchronization API data: No
Collect I/O API data: No
Analyze user tasks: No
Stack unwinding mode: After collection
Stitch stacks: Yes
Collect timeline data: Yes
Collect sleep data: No
Collect frequency data: No
Start
Start Paused
Project Properties
Command Line...

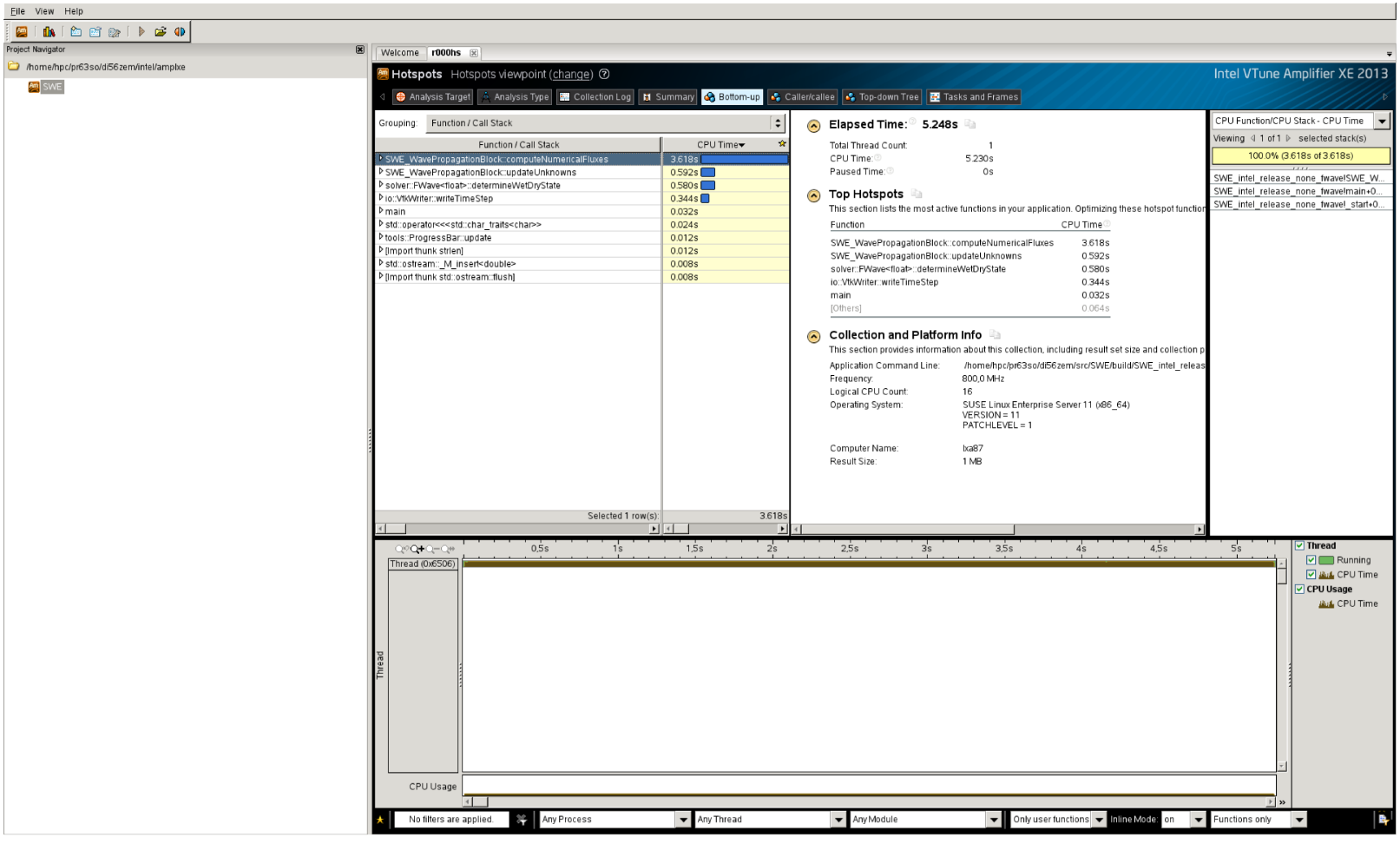
VTune – Start Analysis



VTune – Start Analysis

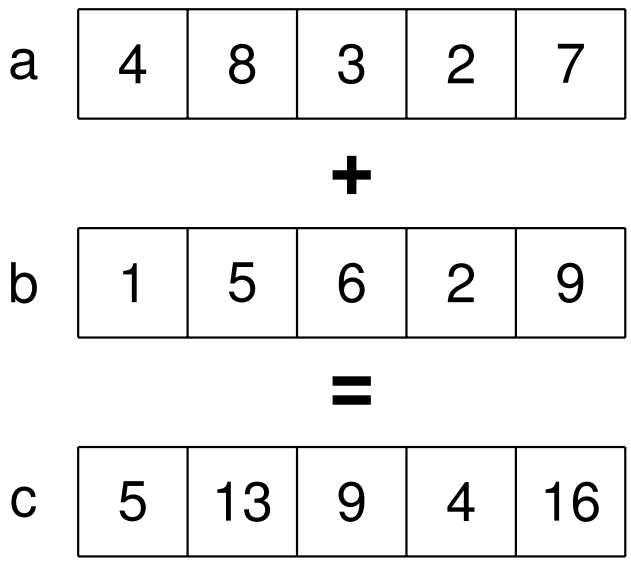


VTune – Results



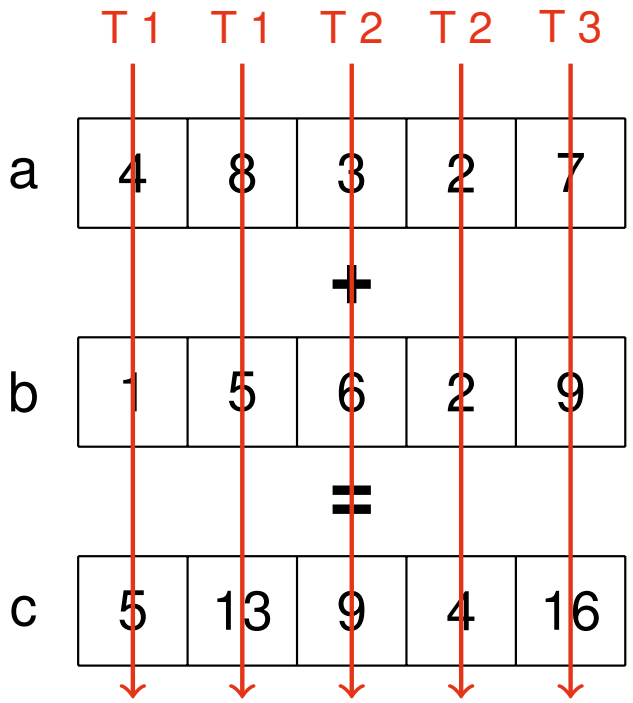
Serial Code

```
for (int i = 0; i < n; i++) {
  c[i] = a[i] + b[i];
}
```



OpenMP

```
#pragma omp parallel for
for (int i = 0; i < n; i++) {
  c[i] = a[i] + b[i];
}
```



Reductions

```
#pragma omp parallel for
for (int i = 0; i < n; i++) {
  c = c + a[i] + b[i];
}
```

Reductions

```
#pragma omp parallel for
for (int i = 0; i < n; i++) {
  c = c + a[i] + b[i];
}
```

What happens if 2 threads write to `c` at the same time?

OpenMP – Critical

```
int sum;

#pragma omp parallel for private(sum)
for (int i = 0; i < n; i++) {
   sum = a[i] + b[i];
   #pragma omp critical
   {
      c += sum;
   }
}
```

OpenMP – Reduction

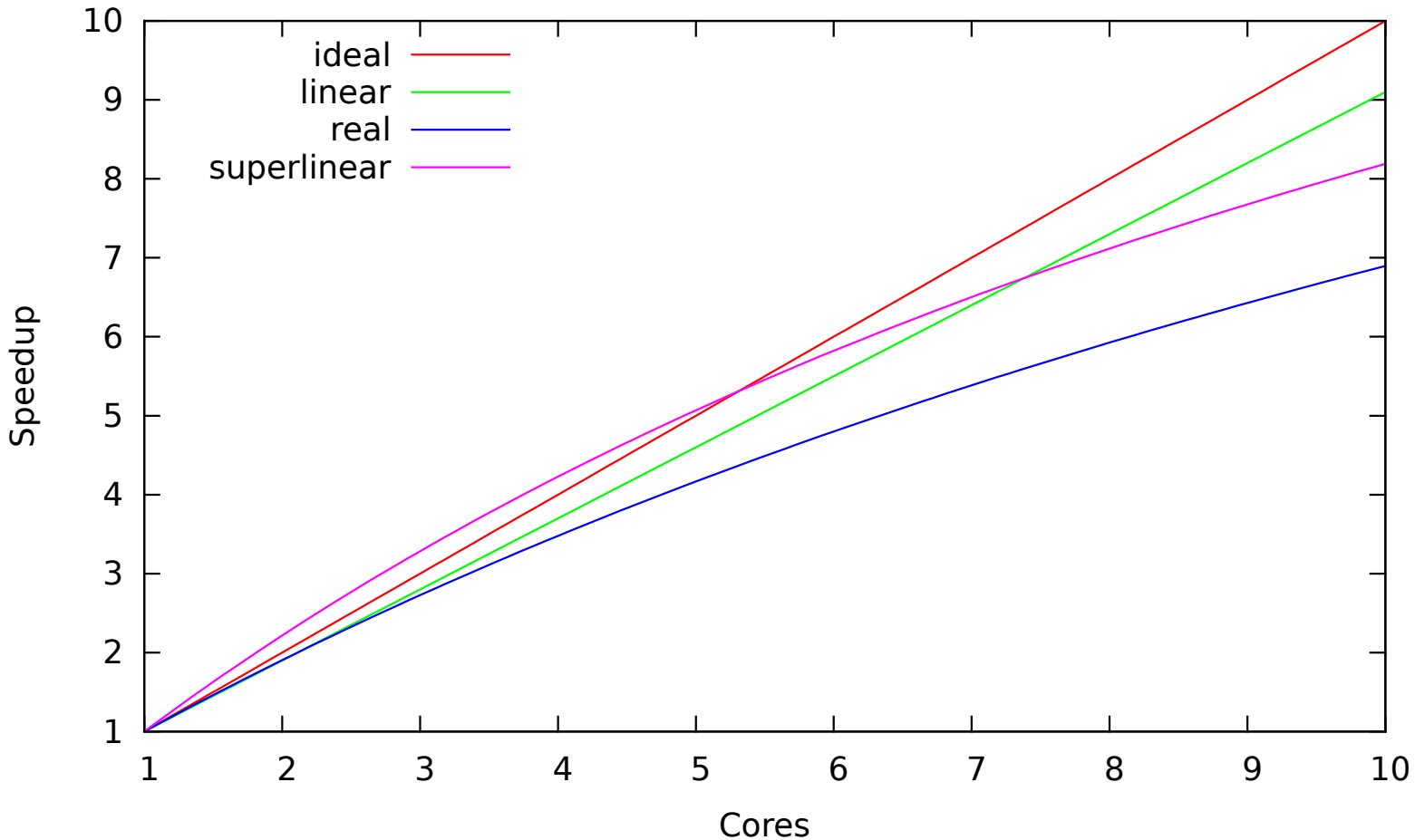
```
#pragma omp parallel for reduction(+: c)
for (int i = 0; i < n; i++) {
  c = c + a[i] + b[i];
}
```

Speedup

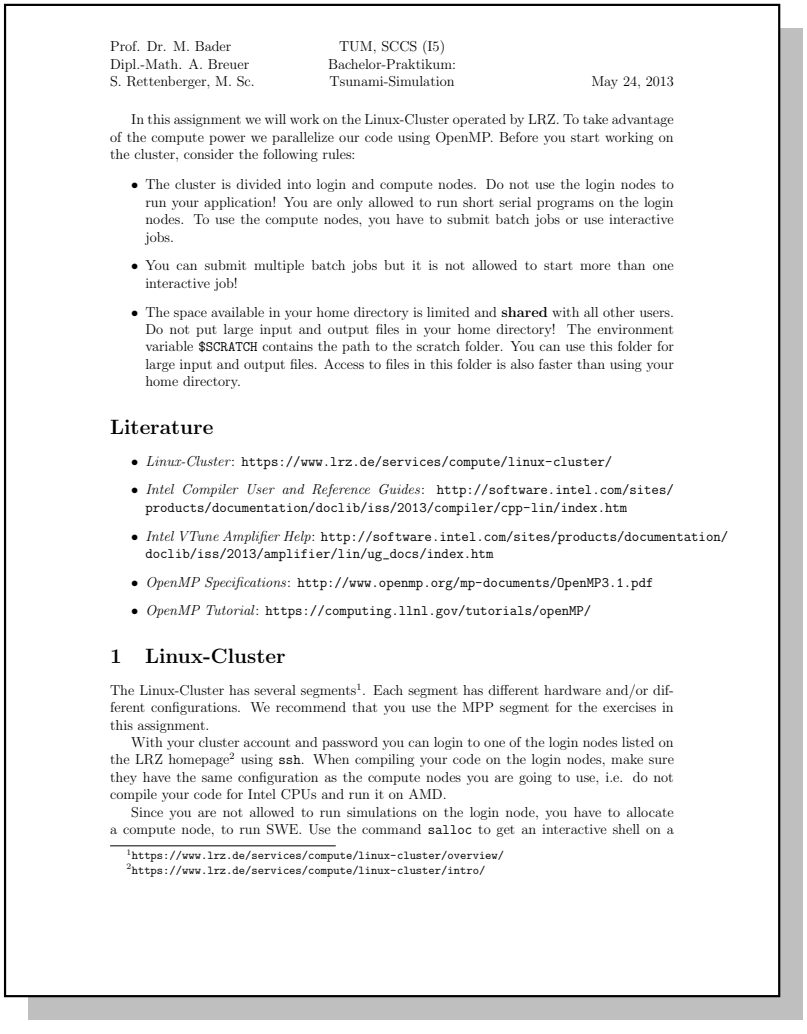
$$S_p = \frac{T_1}{T_p}$$

S_p = speedup for p threads
T_p = run time with p threads

Speedup



Preparation: Assignment 4



Prof. Dr. M. Bader — TUM, SCCS (I5)
Dipl.-Math. A. Breuer — Bachelor-Praktikum:
S. Rettenberger, M. Sc. — Tsunami-Simulation — May 24, 2013

In this assignment we will work on the Linux-Cluster operated by LRZ. To take advantage of the compute power we parallelize our code using OpenMP. Before you start working on the cluster, consider the following rules:

- The cluster is divided into login and compute nodes. Do not use the login nodes to run your application! You are only allowed to run short serial programs on the login nodes. To use the compute nodes, you have to submit batch jobs or use interactive jobs.
- You can submit multiple batch jobs but it is not allowed to start more than one interactive job!
- The space available in your home directory is limited and **shared** with all other users. Do not put large input and output files in your home directory! The environment variable `$SCRATCH` contains the path to the scratch folder. You can use this folder for large input and output files. Access to files in this folder is also faster than using your home directory.

Literature

- *Linux-Cluster*: `https://www.lrz.de/services/compute/linux-cluster/`
- *Intel Compiler User and Reference Guides*: `http://software.intel.com/sites/products/documentation/doclib/iss/2013/compiler/cpp-lin/index.htm`
- *Intel VTune Amplifier Help*: `http://software.intel.com/sites/products/documentation/doclib/iss/2013/amplifier/lin/ug_docs/index.htm`
- *OpenMP Specifications*: `http://www.openmp.org/mp-documents/OpenMP3.1.pdf`
- *OpenMP Tutorial*: `https://computing.llnl.gov/tutorials/openMP/`

1 Linux-Cluster

The Linux-Cluster has several segments[1]. Each segment has different hardware and/or different configurations. We recommend that you use the MPP segment for the exercises in this assignment.

With your cluster account and password you can login to one of the login nodes listed on the LRZ homepage[2] using `ssh`. When compiling your code on the login nodes, make sure they have the same configuration as the compute nodes you are going to use, i.e. do not compile your code for Intel CPUs and run it on AMD.

Since you are not allowed to run simulations on the login node, you have to allocate a compute node, to run SWE. Use the command `salloc` to get an interactive shell on a

[1] `https://www.lrz.de/services/compute/linux-cluster/overview/`
[2] `https://www.lrz.de/services/compute/linux-cluster/intro/`