Introduction

Scope of this document:

This document will very briefly give a short introduction on how to set up the environment for the tools used in this workshop and how to get initially started with each tool. The scope of this introduction is not to get you acquainted with the tools, but only to show you how to get started.
If you require more information about a tool or how to use it, please wait for the official introduction or ask the respective expert for the tool you want to use.

Login:

For this course we offer different systems and operating systems.

Machine	Location
IBM BlueGene/L	(JSC)
SGI Altix 4700	(ZIH)
Sun Sparc T2 (Niagara 2) cluster	(RWTH)
Xeon Infiniband cluster	(RWTH)
Sun Fire V40z	(RWTH)

IBM BlueGene/L

To log onto the IBM BlueGene/L at Jülich Research Center ssh to the login-node **jump.fz-juelich.de** with the username **kurs??** and the announced password, i.e for hpclab01

```
ssh -Y -l kurs01 jump.fz-juelich.de
```

.
After successfully doing so, access the frontend by ssh-ing to **jubl** by entering

```
ssh -Y jubl
```

.
Further documentation is available at `http://www.fz-juelich.de/jsc/ibm-bgl/usage/`.

SGI Altix 4700

To log onto the SGI Altix 4700 at Technical University Dresden, ssh to the login-node **login3.zih.tu-dresden.de** with the username **hpclab??** and the announced password, i.e.

```
ssh -Y -l hpclab?? login3.zih.tu-dresden.de
```

.
After successfully doing so, access the frontend by ssh-ing to **mars** by entering

```
ssh -Y mars
```

.
Further documentation is available at `http://www.tu-dresden.de/hpcadm/hpc-ug/`.

Sun Niagara 2-Cluster

To log onto the Sun Niagara 2-Cluster at RWTH-Aachen University ssh to the node **lab1.rz.rwth-aachen.de** with your username **hpclab??** and the announced password, i.e.

```
ssh -Y -l hpclab?? lab1.rz.rwth-aachen.de
```

.
Further information is available at `http://www.rz.rwth-aachen.de/hpc/primer`.

Xeon-Infiniband-Cluster

To log onto the Xeon-Infiniband-Cluster at RWTH-Aachen University ssh to the node **lab2.rz.rwth-aachen.de** with your username **hpclab??** and the announced password , i.e.
```
ssh −Y −l hpclab?? lab2.rz.rwth−aachen.de
```
.
If you want to use the GNU-compilers instead of the default Intel-compilers, please perform the following steps after each login to switch your software environment:
```
module unload intel−64
```
```
module switch gcc gcc/4.2
```
Further information is available at `http://www.rz.rwth-aachen.de/hpc/primer`.

Sun Fire V40z

To log onto the Sun Fire V40z at RWTH-Aachen University ssh to the node **lab3.rz.rwth-aachen.de** with your username **hpclab??** and the announced password, i.e.
```
ssh −Y −l hpclab?? lab3.rz.rwth−aachen.de
```
.
Further information is available at `http://www.rz.rwth-aachen.de/hpc/primer`.

KOJAK

Setup:

RWTH-Aachen:

load modules: `module load VIHPS kojak`

Research center Jülich:

load modules: `module load kojak`

TU Dresden:

load modules: `module load kojak`

Usage:

1. To automatically instrument your application you may prepend your current compiler/linker call with **kinst** and compile and link as usual; i.e. for C instead of using `gcc ...` to compile and link, use `kinst gcc ...`.
2. The executable the can be executed as usual and will generate the analysis file ”a.elg”.
3. To analyze with a graphical user interface, call `kanal a.elg`.

SCALASCA

Setup:

RWTH-Aachen:

load modules: `module load VIHPS scalasca`

Research center Jülich:

load modules: `module load scalasca`

TU Dresden:

load modules: `module load scalasca`

Usage:

1. To automatically instrument your application you may prepend your current compiler/linker call with **skin** and compile and link as usual; i.e. for MPI and C instead of using `mpicc ...` to compile and link, use `skin mpicc ...`.

2. To perform an analysis run call `scan −t mpirun ...` to start your program. This will generate a <**epik-trace-dir**>.

3. To analyze with a graphical user interface, call `square <epik−trace−dir>`.

NOTE:

Be aware, that automatic instrumentation on Solaris systems is only functional for FORTRAN, for other languages please resort to the manual instrumentation.
For further information, please refer to the scalasca documentation.

Vampir/VampirTrace

Setup:

RWTH-Aachen:

load modules: `module load vampir`

Research center Jülich:

Not available!

TU Dresden:

load modules: `module load vampir`

Usage:

1. For full automatic instrumentation prepend your current compiler/linker call with the vampir specific wrapper and compile and link as usual.
   For C this is **vtcc**, for C++ **vtcxx** and for FORTRAN **vtf90**. For an example one would instrument/compile a C program by calling
   `vtcc −vt:cc mpicc foo.c −o foo.exe`.
   Please note, that the **-vt:cc** <**COMPILER**> specifies the compiler to be used for compilation. If you plan to use a different compiler please specify the compiler after the -vt: option.

2. The executable can then be executed as usual and will generate the analysis file `∗.otf`.

3. Start analyzing the generated trace data by running `vampir *.otf`

NOTE:

Vampir also provides other ways of instrumentation and program analysis. Please refer to the documentation for further information.

MARMOT

Setup:

RWTH-Aachen:

load modules: `module load VIHPS marmot`

Research center Jülich:

TU Dresden:

load modules: `module load marmot`

Usage:

1. For full automatic instrumentation replace your current compiler/linker call with the langage specific wrapper.
   For C this is `marmotcc`, for FORTAN and C++ with `marmotcxx` and `marmotf77` respectively, and compile and link as usual.
   You may have to add −lrt to your linker command line.

2. Run as usual, but add an additional process required by marmot to serve as a debug server, i.e. one needs (n+1) instead of n processes for `mpirun`.

3. Start analyzing reported errors with `cube MarmotLog?.cube`.

Note:

You can control the type of analysis output generated by specifying the environment variable `MARMOT_LOGFILE_TYPE` with

1. ”0” for ASCII format,

2. ”1” for HTML format and

3. ”2” for CUBE format.

OMPP

Setup:

RWTH-Aachen:

load modules: `module load VIHPS ompp`

Research center Jülich:

Not available.

TU Dresden:

load modules: `module load ompp`

Usage:

1. To automatically instrument your application you may prepend your current compiler/linker call with **kinst-ompp** and compile and link as usual.
   For C one would have to change `gcc ...` to `kinst-ompp gcc ...`.

2. The executable the can be executed as usual and will generate the analysis file **executable.*.txt**.

3. To analyze read the reportfile **executable.*.txt** with a suitable editor.