General information about the used benchmark

The used benchmark for this training is NPB3.3-MZ-MPI:
www.vi-hps.org/upload/material/general/NPB3.3-MZ-MPI.tar.gz

For the compilation, first modify or specify the compiler and compilation flags in config/make.def. Use the following line to compile the application:

```
make benchmark CLASS=class NPROCS=number_of_procs [VERSION=VEC]
```

The used benchmark for this training is NPB3.3-MZ-MPI:

```
make bt-mz CLASS=W NPROCS=4
```

Exercise 1: Detecting the focus of analysis (FOA)

a) Instrument the application with *Score-P*. The compiler to use is specified in config/make.def in the variable `MPIF77`.

b) Generate a profile for the program by executing the instrumented binary with the desired number of processes and/or threads.

c) Use the *Cube* GUI to open the generated profile and investigate the results. Identify regions which consume a lot of execution time and which are worth further analysis (FOA).

d) Find out the share of the execution time for calculation, MPI operations and OpenMP synchronization.

e) Identify functions which consume the most exclusive execution time (hotspot functions).

Exercise 2: MPI Communication and OpenMP Synchronization

Some time in the application is spent for communication between processes and for synchronization of threads.

a) Identify how much time is spent for which MPI operations and OpenMP constructs. Is this time equally distributed among processes/threads?

Exercise 3: PAPI counters

a) Identify which PAPI counter are available on your system. If PAPI is already installed, you can see the description of each PAPI counter with `papi_avail` or `papi_native_avail`.

b) For better understanding of computation time load balance among processes/threads, you should observe the distribution of instructions executed by processes/threads. Measure in the next profile run the number of executed instructions and spent cycles by each processes/threads.

c) Calculate the load balance of the instructions among the processes/threads.

d) Calculate the number of instructions per cycle (IPC) for FOA and for the hotspot functions.

e) Measure the number of cache accesses and cache misses. Calculate the cache miss ratio for the FOA and for the hotspot functions.

Exercise 4: Functions filtering and creating of a Trace

For most applications the trace files become very big. Usually most small functions, which are called often, but do not consume a lot of time, need to be filtered out.

a) Identify which functions are called very often with `scorep-score -r profile.cubex`. Create a filter file to filter these functions out.

b) Generate a trace for your application and use the specifications of the filter file. Use the *Vampir* GUI to open the generated trace.

Exercise 5: Efficiency metrics and Load Balance

The efficiency metrics (Serialization Efficiency (SE), Transfer Efficiency (TE), Communication Efficiency (ComE) and Parallel Efficiency (PE)) and Load Balance (LB) can be calculated manually or by using a help script created by JSC.

```
SE = max comp time / max total time on ideal network
TE = max total time on ideal network / max total time
ComE = SE + TE
LB = sum comp time / avg comp time
PE = LB * ComE
```

a) List the efficiency metrics and load balance in a table.

	FOA	Full Application
Parallel Efficiency		
Load Balance		
Communication Efficiency		
Serialization Efficiency		
Transfer Efficiency		

Table 1: Time efficiencies observed in the FOA