Main steps to use the CEPBA tools (Paraver and OMPItrace)

For the EWOMP 2003 - OMPlabs, some of our tools have been installed in the IBM Regatta of the Research Center Jülich (jump.fz-juelich.de) in the lab account `kurs01`.

Before starting to work with the tools from a given account, it is necessary to set up the environment. To do that, modify the `.profile` of your home directory to add the line:

```
. /home1/zam/kurs01/cepbatools/etc/init_tools
```

OMPItrace

1. (optional) define the temporal tracefiles directory and the hardware counters

   ```
   $ setenv MPTRACE_DIR $HOME/scratch
   $ setenv MPTRACE_COUNTGROUP 60
   ```

   Use the option `-h` to obtain a list of all the environment variables available

2. (optional) Create a file specifying the name of the user functions you want to instrument. By default, only the routines that include a parallel directive are intercepted. The, set the environment variable `MPTRACE_ADDFUNCTIONS` to the filename.

3. (optional) If the duration of the application is large or it has a very fine granularity it is recommended to avoid instrumenting the whole run. The two main alternatives to limit the tracing are:
   - Use the environment variable `MPTRACE_FILE_SIZE` to limit the size of the trace. It sets a limit for the size of the temporal trace obtained for each thread, not for the size of paraver trace.
   - Modify the application sources to include calls to `ompitrace_shutdown()` and ompitrace_`restart()`. You will need to link with the OMPItrace library.

4. (optional) If you want to obtain more than one set of hardware counters in the same tracefile, you can use the API call `ompitrace_setgroup(int nb_group)`. You will need to link with the OMPItrace library.

5. Obtain the tracefile running the application under OMPItrace (normal command preceded by ompitrace, use `-counters` if you want to include hardware counters information in the trace)

   ```
   $ setenv OMP_NUM_THREADS <nb_threads>
   $ ompitrace -nosw[1] [-counters] <myprogram>
   ```

In the directory `/home1/zam/kurs01/appl` we have installed the sources of the NAS Benchmarks (OpenMP version). They can be used as example to obtain a tracefile. The BT sources have been modified to instrument only few iterations and to change the set of hardware counters.

[1] In the Jülich Regatta it is mandatory to use the -nosw option to avoid that the instrumentation try to access the switch clock

Paraver

We provide some tracefiles that can be used as examples (`/home/zam/kurs01/traces`):

- **bt.A.*omp.prv** contains a trace of the NAS BT benchmark class A (Open MP version) run with 8, 16 and 24 threads.

- **cam.2mpi*omp.prv.gz** contains a trace of the Community Atmosphere Model (CAM) run with 2 MPI tasks each one with 2 / 8 OpenMP threads.

Basic Navigation

First start Paraver (typing `paraver`) and load the selected trace. The default view displays the state of the application (running, blocking, overhead....). Click on one color and you will see which state it represents and its duration. You can see the list of defined states with the *Show Properties* option of the right mouse button menu.

To get a better look into details your right mouse button lets you zoom into a selected area of the trace. You can zoom and unzoom as needed. Using the *Copy Scale* command (right button) on a window and the *Paste X-scale* on a different one, both windows will be synchronized.

The ▸ ◂ ⏸ symbols will let you navigate through the trace. If the speed is too fast or Paraver is not very responsive, the option *Drawing speed* of the global options will let you adapt the system to your need.

If you want to measure the time between two points in one window (or between windows) the right button menu will let you do so.

Some configuration files have been prepared to easily display some interesting views ($PARAVER_HOME/etc/cfgs). To display them just select the *Configuration - Load Windows* menu.

The structure of the tree of configuration files is:

- General - cfgs applicable to both OpenMP and MPI applications. Views of user functions and user call information.
- OpenMP - cfgs specific of OpenMP runtime: parallel functions, parallel directives, worksarings, barriers...
- MPI - cfgs specific to MPI: MPI calls, Send duration, barriers, long MPI calls...
- Counters - cfgs that use the hardware counters information (IBM Power3 and Power4 processors)
- analyzer - cfgs of 2d analysis (see Basic Analysis section for further details)

For example:

- **OMPItrace/General/user_functions.cfg**: Displays the function executed by each task (timeline display). Each function is encoded by a color. Click on one color and you will see which function it represents and its duration. You can see the list of user functions using the *Show Properties* option.

- **OMPItrace/OpenMP/parallel_functions.cfg**: It displays the outlined functions from the parallel loops executed by the threads. Each outlined function is encoded by a color. Light blue background appears when the thread is outside the parallel functions. Click on one color and you will see which outlined function it represents.

- **OMPItrace/Counters/pwr4/group60/performance/MIPS.cfg**: Displays the MIPS that each process is achieving at each moment. The function is piecewise constant. For a region of constant value, the function of time represents the number of instructions executed in that interval divided by its duration. If you click on a point a what-where window will pop up showing the numerical value.

You can click on the color button and redraw each window to check that each of them just represents a function of time for each process/thread. To learn how these functions have been computed, you can open the *filter* (f) and *Semantic* (S) windows. Notice that the hardware counters are using the next event value, because the value of the counter for a given period is obtained at the end of the period. The user functions, parallels... use the last event value because the information is recorded at the beginning of the function/call.

Basic analysis

With the mechanisms described above you can look at the general behaviour of the application and obtain punctual quantitative information: value of a performance index at a given point in time and measure time between two points. For a more meaningful analysis you will need to obtain some statistics on the performance for the whole trace or at a representative section.

The 2d microscope can be easily used trough configuration files. For example:

- **What would a profiler tell me for this run?**
  - OMPItrace/analyzer/uf_profile.cfg: What you see is for each routine one column, for each process one row and in the intersection the total time spent by that process within that routine.
  - Change the statistic to *Average Burst Time* to compute the average duration of each user function.
  - Change the statistic to *#Bursts* to compute how many times was the function called.
  - To compute the average value of a hardware counter on each user routine load the OMPItrace/analyzer/counterxuf.cfg configuration file. By default the metric MIPS is used. To use a different metric from the available visualisation windows, load the corresponding configuration file that defines the window (if required), change the field *Data Window* to the counter desired and repeat the analysis.

- **Analyzing the OpenMP behaviour...**
  - OMPItrace/analyzer/omp_avgduration.cfg: Average duration of each OpenMP parallel function.
  - Change the statistic to *%Time* to compute the percentage of time spent on each parallel function.
  - Change the statistic to *Average Value* and the data window to the metric MIPS to compute the average number of MIPS obtained by each thread at each parallel function.