Analysis with Paraver - Preparation of trace

Objective

The objective of this session is to describe how to obtain from an initial trace a new trace representing a relevant region of a program run and more appropriate for detailed analysis.
The methodology can be applied to very large traces that would otherwise be impossible to handle by wxparaver.
Besides dealing with the scalability problem of trace based systems it is also useful to select a proper region for performing an analysis. For example it can be used to exclude from a trace (and thus the statistics) initialization regions or other perturbed regions.

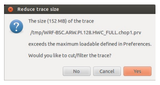
Material

The methodology explained is generally to be applied to traces of hundreds of MBs or GBs. In this case we are not distributing the original trace but we include some intermediate traces in the process and some configuration files.

- WRF-BSC.ARW.PI.128.HWC_FULL.bursts.gnuplot: Example file generated by the initial statistics generation functionality in the paraver loading module.
- ./xml/filter_useful.xml: a file needed by Paraver to specify how to filter very large traces in order to generate traces that represent the full duration of the run but contain a subset of the original information.
- WRF-BSC.ARW.PI.128.HWC_FULL.filter1.prv.gz: The result of applying the filtering process to a huge trace (not provided for space reasons). The original trace represented the execution of the WRF weather modeling application on 128 processors simulating a region of the Iberian Peninsula.
- Useful_duration.cfg: A Paraver configuration file used to indentify structure in the filtered trace.
- WRF-BSC.ARW.PI.128.HWC_FULL.chop1.prv.gz: An example trace obtained through the process described in this document.

First approach to a huge trace

When loading a trace file larger than a maximum size (limit set in the *Preferences window* -> *Global* section -> *Trace* box) wxparaver asks whether to shrink it to a filtered trace, containing only a subset of the records of the original one, but with enough information to perform initial analyses.



Reduce trace size

The size (152 MB) of the trace

/tmp/WRF-BSC.ARW.PI.128.HWC_FULL.chop1.prv

exceeds the maximum loadable defined in Preferences.

Would you like to cut/filter the trace?

Figure 1: Trace size warning window

If you answer *No* you're telling wxparaver to try to load it entirely; be careful, sometimes it's feasable, but not always. Even though you'll be able to later stop the loading and sight what's in the beginning of your trace, that probably wouldn't be useful at all. If you answer *Yes*, the *Cut & Filter* dialog (fig. 2) is shown.

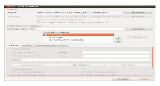


Figure 2: Cut / Filter dialog

With this window you'll be able to access and run some trace utilities useful to perform this size reduction (explained in Step 2).
The main purpose of this tutorial is to guide you through a **three step methodology** that will be appropriate for most of the analyses. Of course, variations of the general methodology can be used for specific needs or purposes. The essential observation behind this methodology is that the most important information in a trace relates to its **computation bursts**.
Before going on, cancel and close all your windows but the main one.

Step 1: Bursts Statistics

Click the gear wheel icon in the main window to open the *Run Application* dialog. Select the *Stats* application, choose the trace (the name of your trace will be fed automatically if you come from the previous *Reduce traze size* warning - fig 1. - and then you clicked the *Yes* button).
We recommend to focus on obtaining basic statistics only for computation bursts (click the *Generate bursts histogram* checkbox). Finally click *Run* button. wxparaver will scan the original trace and generate a gnuplot file shown as an html link in the log textbox (fig. 3).

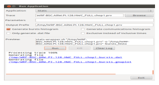


Figure 3: Parametrization of Stats in Run Application dialog

If you click the link, gnuplot's started automatically (see WRF-BSC.ARW.PI.128.HWC_FULL.bursts.gnuplot , also provided as material). Two histograms are shown on the same plot (fig. 4). The green bars are the total amount of time incurred by computation bursts of the different durations. In the figure, 45% of the total time is spent in bursts of length between 500 ms and 1 second.

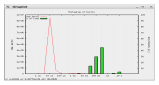


Figure 4: Histogram of computation records

The red line shows the total amount of bursts of a given duration. In the figure, more than 90% of the bursts are between 10 and 50 microseconds.
From the two histograms we can infer a duration such that the total number of computation records above such duration is small and they represent a large part of the total computation time. This number can be fed onto the .xml file used in the next step.
Very often, durations of several milliseconds or tenths of milliseconds are good choices for this parameter.

Step 2: Filtering the trace

As seen before, when trying to load a huge file choose *Yes* in *Reduce size* dialog (fig. 1) to open *Cut/Filter* dialog (fig. 2), or open it directly clicking the scissors icon in the main window. Browse for file xml/filter_useful.xml. It codes the parametrization of the cut/filter utilities. This parametrization can be modified, saved and recovered later (see *Save...* button)
Paraver will scan the original trace and generate a new one named after the original plus a ".filter#.prv" extension. As directed by filter_useful.xml, this file will essentially contain computation burst longer than $5*10^6$ ns.
The file WRF-BSC.ARW.PI.128.HWC_FULL.filter1.prv.gz provided as accompanying material is the result of applying the filtering process to a trace of 20 GB.
Load WRF-BSC.ARW.PI.128.HWC_FULL.filter1.prv.gz and configuration file useful_duration.cfg. The view represents the whole duration of the original trace (a bit over 290 seconds in this case). The pattern that appears is fairly typical of many applications, with an initialization phase and then entering into an iterative behavior. Zooming into the area on the right half of the screen (click and drag) and

fitting the semantic scale to adjust colors (right click to open the pop-up menu, *Fit Semantic Scale -> Fit Both*), we'll be able to see the iterative structure of the program (fig. 5).

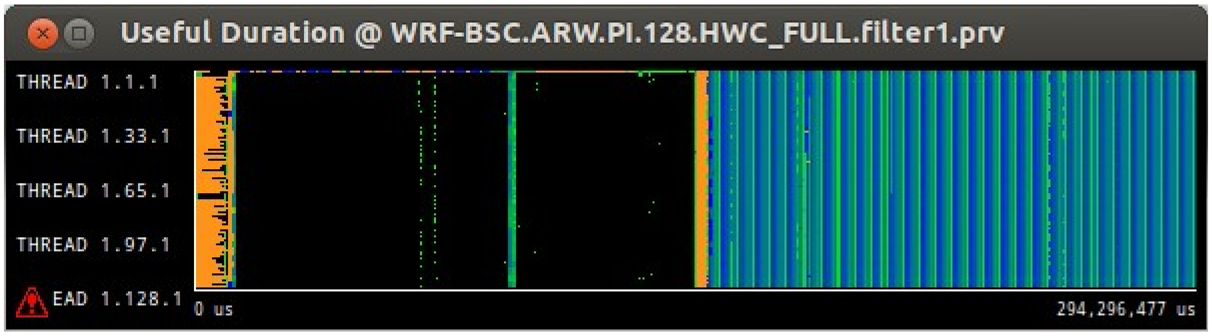


Figure 5: Useful duration view for the whole filtered trace

Recall that there are no MPI events in this trace. Loading the MPI_call.cfg configuration will not show any MPI information. There are no communication records either, so don't expect to see any communication line. Although there's no detail on MPI, it's nevertheless true it's possible to identify communication phases. These will show up as black areas in the “Useful Duration” view. A typical pattern in communication phases is several MPI calls separated by very short computation bursts. By the filtering process above described all these records are discarded. The useful duration view only reports the duration of the useful computation areas so its value is zero for all areas corresponding to MPI calls and small computations in between.
If you look at the filter_useful.xml file, you will see that besides letting through states (computation bursts) longer than 5000000 ns base of the description in this and the next section, certain event types are also passed to the generated file. If you look at the WRF-BSC.ARW.PI.128.HWC_FULL.filter1.pcf file (a copy of the pcf of the original trace) you can see the specific meaning of some of them. For example event type 40000003 identify when a process is flushing its memory trace buffer to disk. Events from 3000 to 3002 contain information about where (node, line card,...) each process is executed within the MareNostrum platform (these events may not be generated on other platforms). The standard flushing.cfg or cfgs related to process mapping cfgs can be used on the filtered trace.

Step 3: Extraction of representative subtrace

Zooming and navigating in the area that corresponds to the iterative computation structure we can actually identify clean iterative regions (fig. 6) and regions perturbed for example by preemptions (fig. 7).

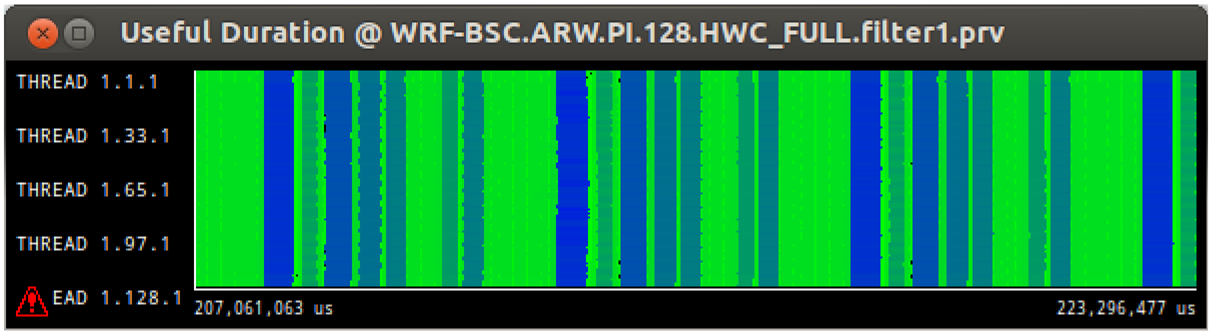


Figure 6: Iterative timeline structure. Dark blue represents a long computation phase. Light green regions with short computation bursts

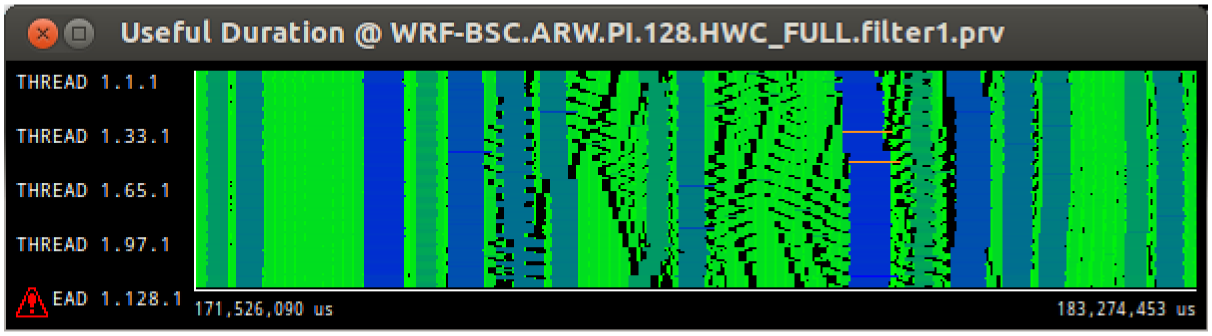


Figure 7: Region with perturbation. Communication phases (black) longer than average and their propagation

As final step we should identify a few iterations in a clean region (i.e. fig. 6) and obtain a cut of the original trace just representing the selected iterations. To do so, the easiest way is to right click on the timeline and select *Run -> Cutter*. The recommended “Trace Options” are set by default: don't use the original time, don't break states don't remove first state and remove last state. Then click on *Apply* button.
Another way is to invoke the Cut & Filter dialog with the scissors icon, browse for the input trace to cut, select "Cutter", preset the previous "Trace Options" and then click in the *Select Region...* button. You'll see the mouse icon changes to a crosshair icon. Then go back to the timeline, and drag and drop or click begin and end of the region to cut. Both times will be set as begin and end time limits.
Cutting can be done indicating the selected area on any window. Our recommended process is nevertheless to use a view of “Useful duration” and to select the lower and upper limits on the dark blue regions, clicking in the middle of the longest computation phases. The selection of lower and upper limit are made at the same point in time for all processes. By targeting long computation regions we make easy to point for all processes halfway through the same computational phase. The selected option not to “Remove First State” actually drives wxparaver to include the whole state at each process instead of cutting it. The option “Remove Last State”

drives wxaraver to discard the whole state at the right limit. In this way it is possible to extract whole iterations even if there is some skew or imbalance between processors.
The option not to “Use original time” shifts the cut thus obtained to start at time zero.
The resulting trace will include all the MPI events, communication records and hardware counter events for the selected area. Detailed analyses can thus be performed on it.