Traces Preprocessing Tool

Abdellah Rahmani

January 28th, 2022

Polytechnique Montreal

DORSAL Laboratory

Agenda

- Trace Compass scalability
- Traces preprocessing tool
- Demo
- What’s next ?
- Conclusion

Traces preprocessing tool - Context of this work

- Trace compass scalability: analyze very large traces, cover new uses cases such as HPC clusters (very large number of nodes).

- Other work in this context:
  . Distributed Architecture of trace compass
  . Distributed computation of the critical path
  . Partial history tree

Traces preprocessing tool - What ? Why ?

- Non-interactive generation indexes and intermediate analysis files
- After preprocessing, opening and navigating the traces is very quick
- Precomputing several analyses with one command
- Saves resources usage since it’s CLI-based

Traces preprocessing tool - How it works ?

- Find the traces (metadata files) reached from the input folder
- Posts traces, creates an experiment and starts the indexing
- Launches analyses preprocessing
- Interacts with the trace-server with **TSP** using **libcurl** library (HTTP requests)
- Parses Trace server output (Json) using **libjsoncpp** library to check requests status.

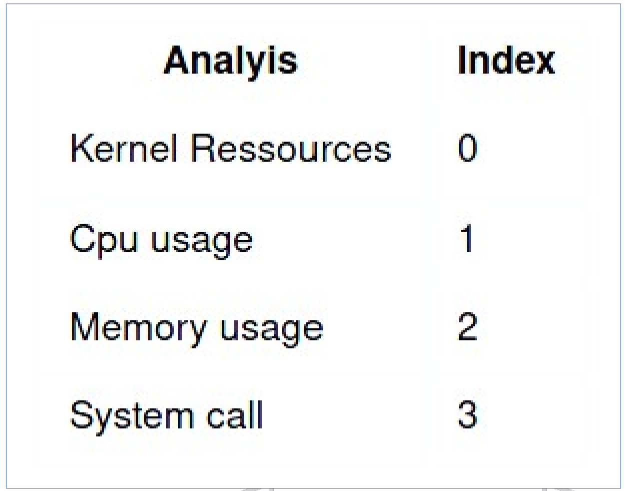
Traces preprocessing tool - How to use it ?

The Precomputing part:

- Download the trace server and install the dependencies
- Build the script with a simple *make* command
- Start the trace server and launch the script :

./preprocessor /Path_to_the_traces_directory /Path_to_the_trace-server_workspace [0,1,2]

- [0,1,2] is an array that contains the analyses the user wants

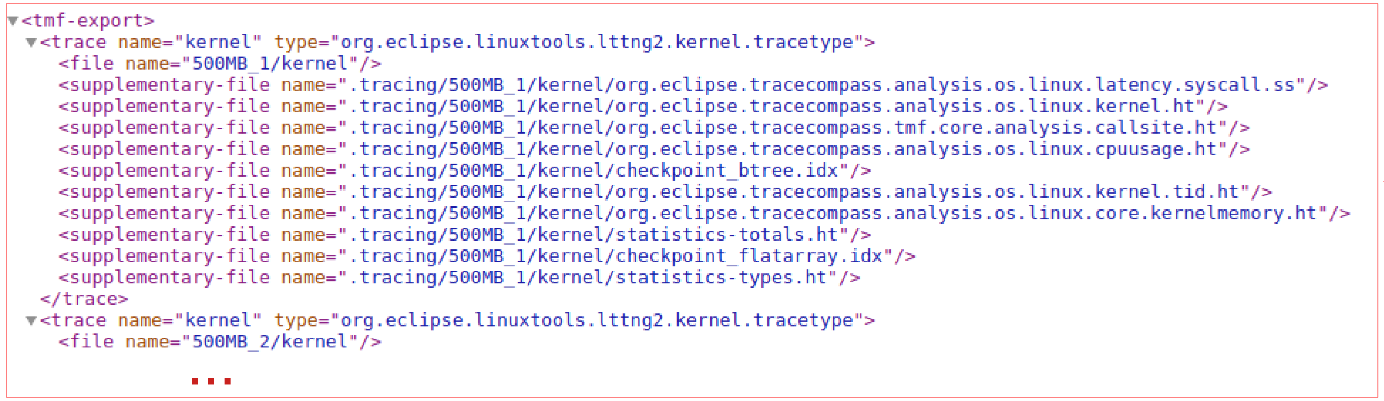


Analyis	Index
Kernel Ressources	0
Cpu usage	1
Memory usage	2
System call	3

Traces preprocessing tool - How to use it ?

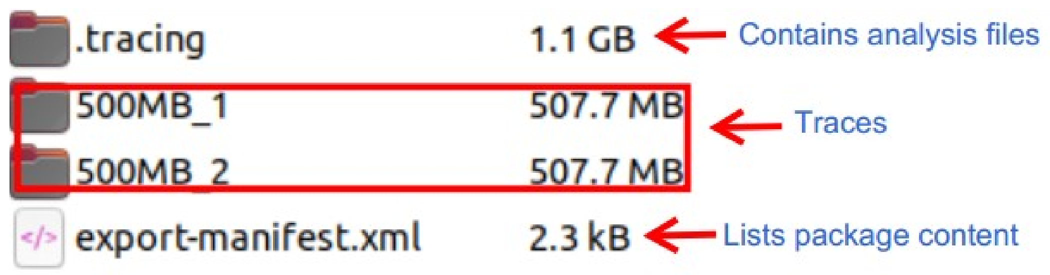
The Precomputing part:

- Possibility of importing a trace package into Trace Compass workspace



```
▼<tmf-export>
  ▼<trace name="kernel" type="org.eclipse.linuxtools.lttng2.kernel.tracetype">
     <file name="500MB_1/kernel"/>
     <supplementary-file name=".tracing/500MB_1/kernel/org.eclipse.tracecompass.analysis.os.linux.latency.syscall.ss"/>
     <supplementary-file name=".tracing/500MB_1/kernel/org.eclipse.tracecompass.analysis.os.linux.kernel.ht"/>
     <supplementary-file name=".tracing/500MB_1/kernel/org.eclipse.tracecompass.tmf.core.analysis.callsite.ht"/>
     <supplementary-file name=".tracing/500MB_1/kernel/org.eclipse.tracecompass.analysis.os.linux.cpuusage.ht"/>
     <supplementary-file name=".tracing/500MB_1/kernel/checkpoint_btree.idx"/>
     <supplementary-file name=".tracing/500MB_1/kernel/org.eclipse.tracecompass.analysis.os.linux.kernel.tid.ht"/>
     <supplementary-file name=".tracing/500MB_1/kernel/org.eclipse.tracecompass.analysis.os.linux.core.kernelmemory.ht"/>
     <supplementary-file name=".tracing/500MB_1/kernel/statistics-totals.ht"/>
     <supplementary-file name=".tracing/500MB_1/kernel/checkpoint_flatarray.idx"/>
     <supplementary-file name=".tracing/500MB_1/kernel/statistics-types.ht"/>
   </trace>
  ▼<trace name="kernel" type="org.eclipse.linuxtools.lttng2.kernel.tracetype">
     <file name="500MB_2/kernel"/>
          ...
```

Trace package structure



.tracing	1.1 GB	Contains analysis files
500MB_1	507.7 MB	Traces
500MB_2	507.7 MB	
export-manifest.xml	2.3 kB	Lists package content

- Adding ***-p*** or ***--package*** will create a trace package with the script:
  ./preprocessor /Path_to_the_traces_directory /Path_to_the_trace-server_workspace [0,1,2] --package
- To be used ideally with small traces (zipping / unzipping time)

Traces preprocessing tool - How to use it ?

The results visualization part:

The following options can be used:

- Open the Trace compass server workspace with trace compass
- Open the views with Theia Trace Viewer using a browser
- Import the trace package to trace compass if you have created one.

Traces preprocessing tool - Demo

Demo

Trace Compass Scalability - What next ?

Preprocessing tool :

- Add an estimation of computing time
- Put the script in a docker to make it cross-platform
- Use Mpi to deploy the preprocessing on several nodes

Trace Compass scalability :

- Efficient preprocessing on individual nodes in the traced cluster
- Minimize the disk footprint of the indexes and analysis (Partial State History)
- Minimize the disk footprint of the traces (different streams, filtering)
- Parallel processing of traces from huge clusters for interactive querying and viewing

Traces preprocessing tool - Conclusion

- Option of batch generation of indexes and analysis files
- Thereafter, opening and navigating a Trace is quick even if > 100GB
- Study the performance of Trace Compass on these huge traces and further optimize
- Keep Trace Compass the best tool for huge traces!

Q&A

Source code repository: https://github.com/dorsal-lab/Trace-preprocessing-script-

Email: abdellah.rahmani@polymtl.ca