nsys2prv: translate Nsight System traces to Paraver

Marc Clascà

Best Practices for Performance and Programmability

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tools overview

Extrae and Paraver

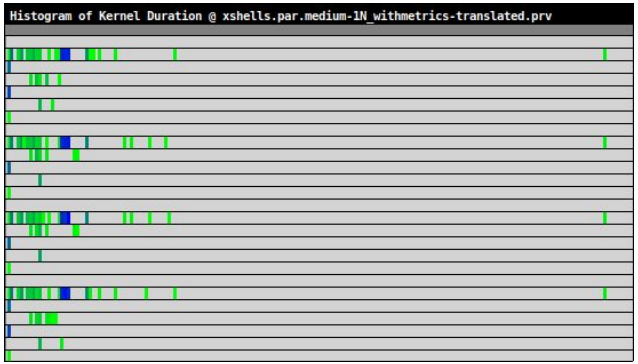
- Parallel performance analysis
- Configurable visualization
- Trace information -> Semantics
- Missing kernel names

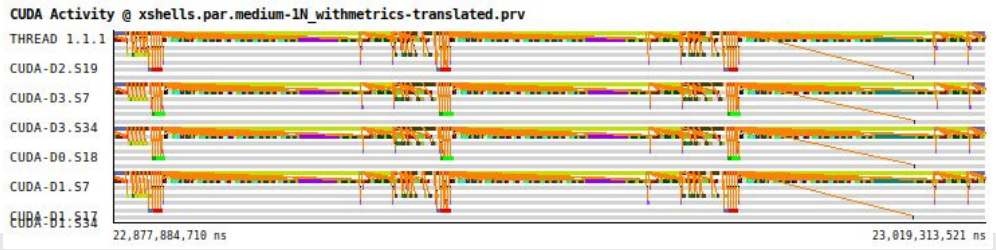
NVIDIA Nsight Systems/Compute

- NVIDIA libraries -> More information (for the moment)
- NOT a parallel profiler
  - Not good visualization for a lot of processes
  - Not all communications information
  - Different report for each process for multi-node
- … but outputs all information in CSV format :)

nsys2prv tool: translate nsys traces to Paraver format

- Python script
- Reads data in CSV format from `nsys stats` command and SQLite DB
  - CUDA API calls
  - Kernels: execution, Block/Grid, name, launch->execute relation, regs/thread
  - Memory transfers: from/to relationship, size
  - MPI Calls
  - NVTX regions
  - GPU metrics
  - OpenACC
- Custom CFGs, more to come





Benefits and limitations

Benefits:

- Profit from all nsight information + adapting it to paraver semantics
- Easy data treatment with python
- Supports analyst work: fast tool feature integration -> first script, then to extrae

Main enabler for performance analysis: reduce gaps of different tools

Limitations:

- Nsight Systems csv output format not documented
- SQLite schema briefly documented
- Only from 1 report: WIP for multi-node
- Still it’s a lot of information, manual translation

Usage

From an Nsight Systems profiling…

```
$> nsys profile --gpu-metrics-device=all -t cuda,nvtx,mpi… -o ./llm_all python
TestLLAMA.py
```

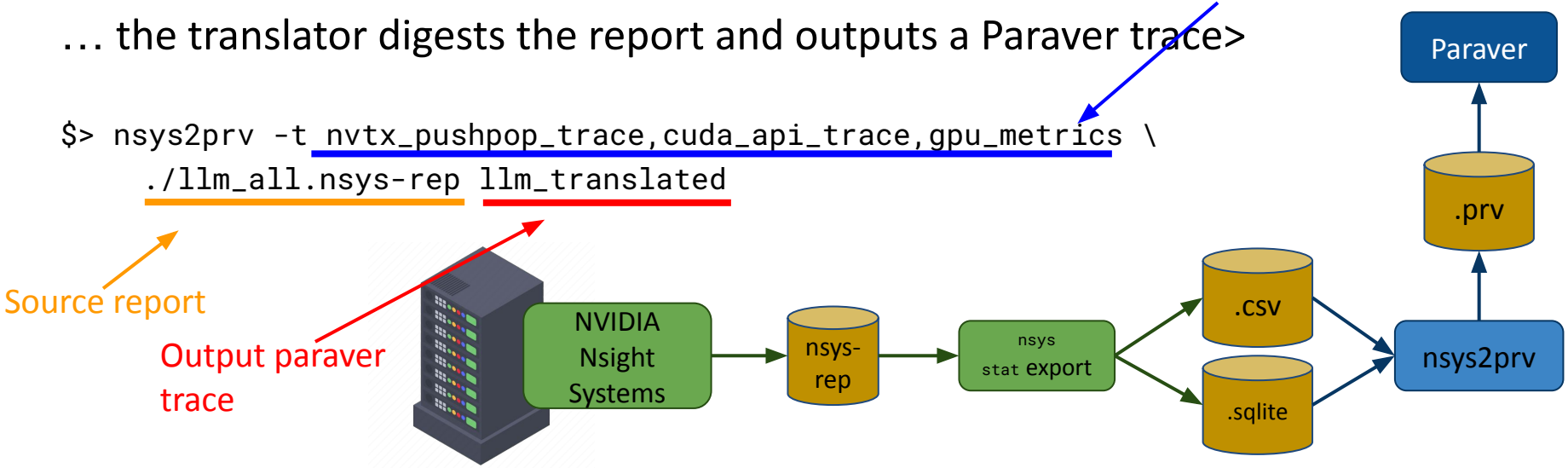
… the translator digests the report and outputs a Paraver trace>

Information to be translated

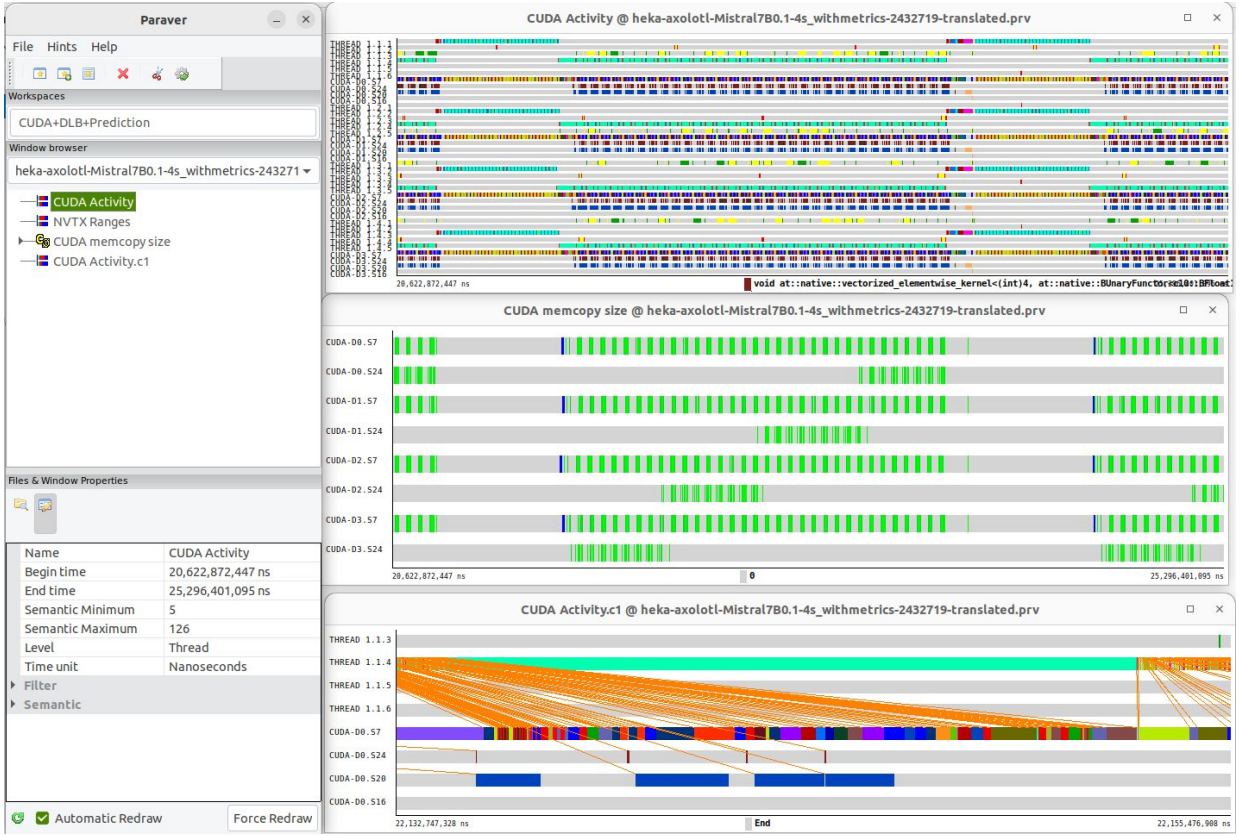
```
$> nsys2prv -t nvtx_pushpop_trace,cuda_api_trace,gpu_metrics \
     ./llm_all.nsys-rep llm_translated
```

Source report

Output paraver trace



Examples: LLM training



Examples: SOD2D