Profiling Tools in PaRSEC

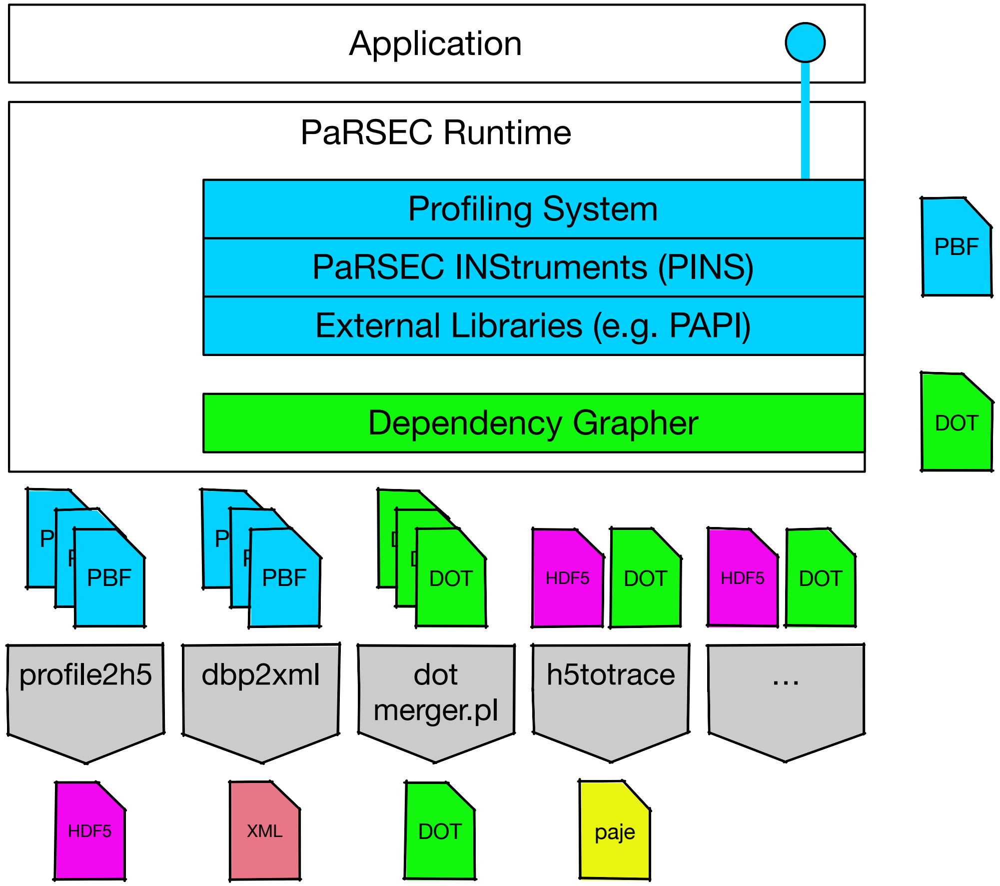
PaRSEC User Group Meeting

INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Profiling System

- Two main runtime components:
  - Dependency Grapher produces a DOT of the DAG of tasks.
    - Cost is non-negligible (I/O at the preamble and epilogue of each task)
  - Profiling System keeps a track of the execution
    - Significant effort to keep that cost down
    - Modular design: goal is to allow power users to inject their own instrumentation if needed
    - Provide interface to PAPI
  - Each component produces one file per rank
- A collection of user-level tools to process the produced files



Compile PaRSEC with Profiling

- Two options:
    - PARSEC_PROF_TRACE
        - Enables ‘profiling’: events can be logged in a binary format
        - Can also use OTF2 format, if libotf2 is available, but functionality is reduced
    - PARSEC_PROF_GRAPHER
        - Enables dependency graphing
- Dependencies:
    - Python 2.7.15 or later and cython 0.21.2 for most profiling **tools (optional, but recommended)**
    - PAPI for the pins_papi MCA module (optional)

```
parsec/build/fast-profiling > ../../configure \
    --prefix=$HOME/parsec/install-dir/fast-profiling \
    -DPARSEC_PROF_TRACE=ON \
    -DPARSEC_PROF_TRACE_SYSTEM=“PaRSEC Binary Tracing
Format” \
    -DPARSEC_PROF_GRAPHER=ON

[…]

-- Found Python: /nfs/apps/spack/opt/spack/linux-
centos7-x86_64/gcc-7.2.0/python-2.7.15-
aczt3ejnkaiikvwqqy7btuccpm2bpgnm/bin/python2.7 (found
version "2.7.15") found components: Interpreter
Development
-- Cython version 0.29.15 found
-- Found Cython: /nfs/apps/spack/opt/spack/linux-
centos7-x86_64/gcc-7.2.0/python-2.7.15-
aczt3ejnkaiikvwqqy7btuccpm2bpgnm/bin/cython (Required is
at least version "0.21.2")
-- -- Found Component ‘pins’
-- The PAPI Library is found at /spack/opt/spack/linux-
scientific7-x86_64/gcc-7.3.0/papi-5.6.0-
dybzvixufstomkhem7ayjmyjdpfprcsc/lib/libpapi.so
-- ---- Module ‘papi’ is ON
-- Profiling uses PaRSEC Binary Tracing Format
```

Run PaRSEC with profiling

- MCA parameter: `profile_filename` -- where to store the PBF files.
  - This is a template name, files will be named `<profile_filename>-<rank>.prof-<random>`
- PaRSEC parameter: `--dot [<filename>]`
  - This parameter is for PaRSEC, not for the app
  - `<filename>` is also a template: the files will be named `<filename>-<rank>.dot`
- Only one of them may be specified
- It is critical for performance that these files are as local as possible (/tmp, scratch directories, etc…)
  - By default, the tracing system uses trunc/mmap to resize the events files, and this sometimes fails on NFS directories. A compilation variable, `PARSEC_PROFILING_USE_MMAP` can be undefined in profiling.c if this prevents profiling to execute
  - 3 MCA parameters exist to tune the profiling:
    - `profile_buffer_pages`: how many pages per buffer per thread are allocated
    - `profile_file_resize`: increment at which event files are resized (in number of buffers)
    - `profile_show_profiling_performance`: displays at the end of the execution the time spent in profiling routines

```
> mpirun -map-by node -np 2 \
    ./tests/stencil/testing_stencil_1D \
    -- --mca profile_filename\
                 /scratch/shared/herault/stencil

[****] TIME(s)      0.00158 : Stencil N= 8 NB= 4
M= 8 MB= 4 PxQ= 1 2 KPxKQ= 1 1 Iteration= 10
Radius= 1 Kernel type= 0 Number_of_buffers= 2
cores= 20 : 0.001213 gflops


> ls /scratch/shared/herault/

stencil-1.prof-Hx6eWl   stencil-0.prof-Hx6eWl
```

```
> mpirun -map-by node -np 2 \
    ./tests/stencil/testing_stencil_1D \
    -- --dot /scratch/shared/herault/stencil

W@00000 /!\ DEBUG LEVEL WILL PROBABLY REDUCE THE
PERFORMANCE OF THIS RUN /!\.
[****] TIME(s)      0.00745 : Stencil N= 8 NB= 4
M= 8 MB= 4 PxQ= 1 2 KPxKQ= 1 1 Iteration= 10
Radius= 1 Kernel type= 0 Number_of_buffers= 2
cores= 20 : 0.000258 gflops

> ls /scratch/shared/herault/

stencil-0.dot  stencil-0.prof-Hx6eWl
stencil-1.dot  stencil-1.prof-Hx6eWl
```

Implementation

- Trace collection is per-stream
- 1 Computing thread in PaRSEC = 1 Profile Stream
- Comm. Thread has 3 Profile Streams (task notification, payload send, payload receive)
- CUDA devices create one stream per CUDA stream
- Most tracing operations on streams are independent: no atomics.
- Each stream appends events of variable sizes on its own even buffer

- Buffer management - MMAP
  - 1 additional thread is created to resize the backend file, and pre-allocate 1 buffer / stream in advance in that file
    - goal is to minimize wait time to acquire a new buffer
    - cost is a file size that is overestimated, and I/O in parallel with computation
    - this thread holds 99% of its time in blocking FS calls or idle on semaphores
- Buffer management – append to file
  - Buffer allocation is centralized
  - buffers are dumped on file one after the other, in a serial manner

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

User API

- Most calls are done by the runtime system, without requiring anything from the user:
  - trace of internal events (memory allocations, notifications, memory transfers)
  - trace of inter-process messages
- DSLs also automatically decorate their code with tracing calls
  - trace of start and end of task execution
- High-level programs can (optionally) add information to the trace

- On any rank:

  -
    ```
    void profiling_save_dinfo(
        const char *key,
        double value);
    ```

  -
    ```
    void profiling_save_iinfo(
        const char *key,
        int value);
    ```

  -
    ```
    void
    profiling_save_uint64info(
        const char *key,
        uint64_t value);
    ```

  -
    ```
    void profiling_save_sinfo(
        const char *key,
        const char *svalue);
    ```
- NB: if two ranks define the same key, the HDF5 will only show one of the values

PaRSEC INStrumentation (PINS)

- In addition to the profiling system, the runtime has hooks placed at many critical steps of a task lifecycle:
  - first time a task is discovered
  - every time one of its input flow becomes ready
  - when the task becomes ready to execute
  - when it prepares its input data
  - when it starts (possibly multiple times) executing, every time it returns from the execution
  - when it is released
- Each of these places can become a logged event using the PINS system
- PINS MCA components can then decorate each event with more information.
  - Example: PINS PAPI module:

```
--mca mca_pins papi --mca pins_papi 1 --mca pins_papi_event “S*:C*:PAPI_L1_DCM”
```

Tools

- Some tools are binary executables
  - parsec-dbp2xml, parsec-dbp2mem, parsec-dbpinfo, …
- Others are Python scripts
  - profile2h5, h5totrace, …
- Some are Perl scripts
  - parsec-dotmerger
- Although it is possible to use them within the build/source directories, the easies way to use them is to install and load the environment provided in the bin/ subdirectory after install in bash.env or csv.env

```
> make -j 20 install
> export PARSEC_ROOT=[…]
> . $PARSEC_ROOT/bin/bash.env
> which parsec-ptgpp
[…]/bin/parsec-ptgpp
```

Tool: parsec-dotmerger

- Takes a set of per-node DOT files, and merges them in one DOT file
- DOT is the format used by graphviz (https://www.graphviz.org/) to visualize graphs (in our case the DAG of tasks)
- parsec-dotmerger has options to
  - Select what nodes or edges to ignore
  - Select the content, form and color of nodes and edges
  - See parsec-dotmerger –h for a full list of options

```
> $PARSEC_ROOT/bin/parsec-dotmerger \
    /scratch/shared/herault/stencil-0.dot \
    /scratch/shared/herault/stencil-1.dot > \
            stencil.dot

> dot -Tpdf -o stencil.pdf stencil.dot
```

Tool: profile2h5

- The binary profiling format is needs to be assembled into a portable file.
- parsec-dbp2xml provides only rudimentary information (tasks start and end dates)
- the recommended approach is to convert the parsec binary format files into pandas dataframes stored in an HDF5 file
- This is the role of profile2h5
  - Note: profile2h5 will check if a target hdf5 file already exists. If it exists, it will not re-generate it
  - i.e. if you want to re-generate an hdf5 from a new profile with the same name, you need to move or delete the existing hdf5 file.
- profile2h5 depends on pandas, with pytables for HDF5 support. Minimal versions are:
  - pandas: 0.24.2
  - numpy: 1.16.6
  - tables: 3.5.1
- All these tools can be installed in the user directory using pip

```
> cd /scratch/shared/herault/

> profile2h5 stencil-*.prof-Hx6eWl

Processing ['stencil-0.prof-Hx6eWl',
'stencil-1.prof-Hx6eWl']

Generated: stencil-ap-Hx6eWl.h5


> file stencil-ap-Hx6eWl.h5

stencil-ap-Hx6eWl.h5: Hierarchical Data
Format (version 5) data
```

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

HDF5 Profiling files

- Hierarchy of pandas DataFrame
  - Most data is in events
  - Meta-data is in information, nodes, streams, and event_names/event_types
  - information holds a key-value store that is application-specific
  - event_names/event_types are key-value stores to identify the events
  - nodes and streams are DataFrames to identify the processes and the streams in each process
    - Streams are profiling streams: there might be more than one per thread (e.g. 3 for the comm. thread, one per CUDA stream, etc..)

/

- errors
- event_attributes
- event_convertors
- event_names
- event_types
- events
- information
- nodes
- streams

HDF5 Profiling: /nodes

- Basic information about the nodes
  - id is the value used in other DataFrames for node_id
  - HWLOC-XML is a dump of the hwloc topology loaded on the node, if hwloc is available.
  - sched is the scheduler loaded at init time
  - nb_cores is the number of computing threads (including the main thread) used by this run
  - MEMORY_USAGE and MEMORY_USAGE_list hold some statistics of mempool memory usage internal of PaRSEC (data repositories and tasks contexts)

/nodes
CMDLINE
DEVICE_MODULES
DIMENSION
GIT_BRANCH
GIT_HASH
HWLOC-XML
MEMORY_USAGE
MEMORY_USAGE_list
cwd
error
exe
exe_abspath
filename
hostname
id
nb_cores
nb_vps
sched

HDF5 Profiling: /streams

- begin/end/duration: times related to the stream (in unit of the realtime timer of the machine that did the run, usually nanoseconds)
- boundto: binding information (core). Displayed as a float. NaN for non-binding
- description: human-readable information about the stream
- node_id/th_id/vp_id: the identifiers of the node/thread/virtual process that hosts this stream
- stream_id: the identifier used in other DataFrames to relate to this stream

/streams	begin
	end
	duration
	boundto
	description
	node_id
	stream_id
	th_id
	vp_id

HDF5 Profiling: /information

- Key/Value store: used by the application to store meta-information
  - For the stencil example, here is the information stored by the application. They overlap some information stored in the DataFrame /nodes.
  - This is purely user-defined.

/information	CMDLINE
	DEVICE_MODULES
	DIMENSION
	GIT_BRANCH
	GIT_HASH
	cwd
	error
	exe
	exe_abspath
	last_error
	nb_cores
	nb_nodes
	nb_vps
	sched
	worldsize

HDF5 Profiling: /event_types

- /event_types: key-value store that maps event names (string) to event type (number)
- /event_names: key-value store that maps event types (number) to event names (string)

/event_types	N/A
	TASK_MEMORY
	Device delegate
	cuda
	movein
	moveout
	prefetch
	cuda_mem_alloc
	cuda_mem_use
	MPI_ACTIVATE
	…
/event_names	0
	1
	2
	3
	4
	5
	6
	7
	8
	…

INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

HDF5 Profiling: /events

- Large DataFrame holding all ‘events’
- Rows in this dataframe are a pair of event: an event.begin and an event.end.
- Each event pair happens on a given stream (stream_id), on a given node (node_id), and has a unique type, a begin time and an end time.
- Some events are related to a given taskpool (e.g. execution of a task) (taskpool_id)
- Other events are related to a data (e.g. a GPU memory transfer) and have a data_collection_unique_key and a data_collection_data_id.
- Some events are network-related (e.g. communication) and have a source (src), a destination (dst), and sometimes a task identifier (did – type identifier of the task/tid – task identifier in this type).
- Each event type may define additional columns in this table, and the values in these columns make sense only for these event types.

/events	begin
	end
	stream_id
	node_id
	type
	taskpool_id
	data_collection_unique_key
	data_colection_data_id
	data_collection_padding
	src
	dst
	did
	tid
	… [taskpool-specific]

HDF5 Profiling: example.

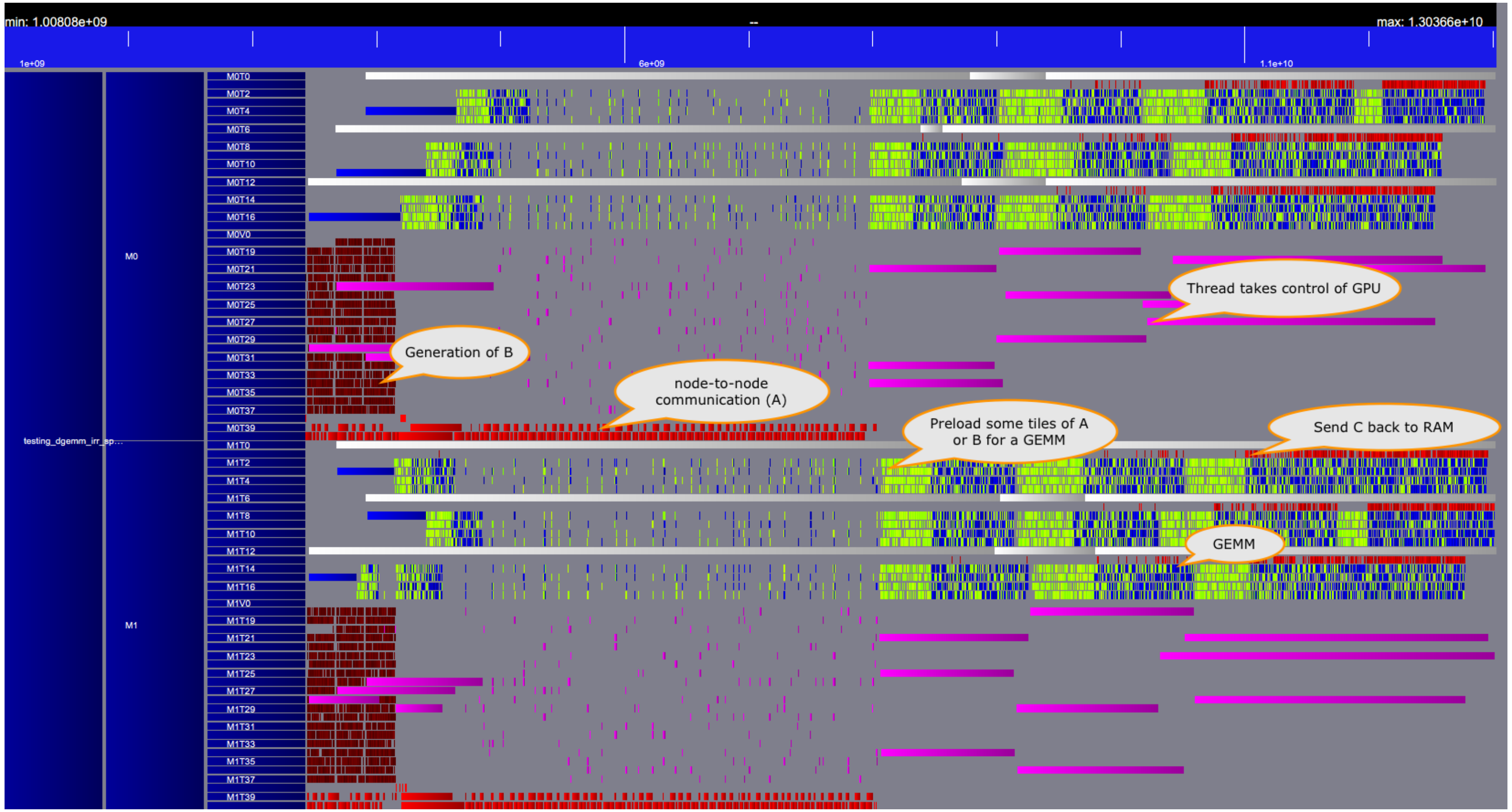
```
>>> import pandas as pd
>>> import numpy as np
>>> t=pd.HDFStore(‘stencil-ap-Hx6eWl.h5’)
>>> t.events.taskpool_id.unique()
array([2, -1], dtype=object)
>>> m = t.events[t.events.taskpool_id == 2]
>>> m['duration'] = m['end']-m['begin']
>>> m['duration'] = m['duration'].astype(np.int)
>>> m[['duration','type']].groupby(['type']).describe()
     duration
        count        mean         std    min    25%    50%    75%     max
type
15       22.0  780.590909  297.918758  269.0  624.5  724.0  980.0  1373.0
>>> t.event_names[15]
'task'
```

HDF5 Profiling: example.

```
>>> c=t.events[((t.events.type >= t.event_types[‘MPI_ACTIVATE’]) &
                (t.events.type <= t.event_types[‘MPI_DATA_PLD_RCV’]))]

>>> c['duration']=c['end'].astype(np.int)-c['begin'].astype(np.int)

>>> c[['duration','node_id']].groupby('node_id').describe()

        duration

           count          mean           std     min      25%     50%       75%        max

node_id

0           16.0  1.070306e+04  1.199261e+04  1461.0  2082.25  2739.0  16566.50    33097.0

1           16.0  1.587329e+06  3.404030e+06  1308.0  1914.75  4253.0   8713.75  8449498.0
```

Tool: h5totrace

- Converts an HDF5 profiling database into a PAJE trace (http://paje.sourceforge.net/)
- Can be visualized with Paje (old), or Vite (still in dev.): http://vite.gforge.inria.fr/
- See h5totrace –help for list of options



–counter: some events (e.g. memory allocation events) are better represented as a line that accumulates values than a block. This option allows to select which event names are considered counters

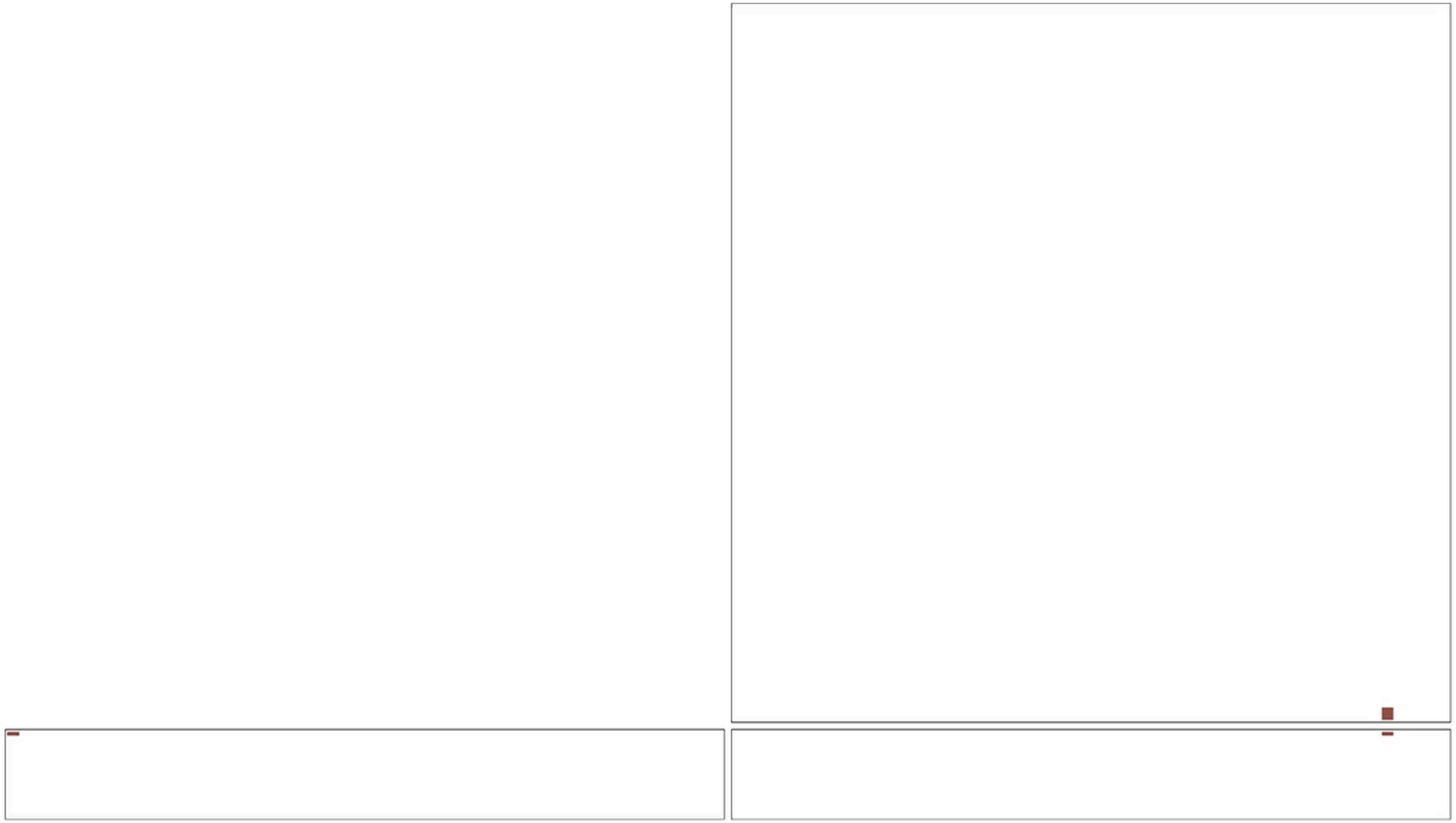
–ignore-type / –ignore-stream: allows to trim the trace to make it easier to process or visualize

–list: just list the event types in the HDF5 (useful for –ignore-type)

–dot / --dot-DAG : optional, can take the DOT information to add arrows between tasks to represent the DAG of dependencies on top of the Gantt diagram

–COMM: represents communications between nodes with arrows

Ad-hoc visualizations



ICL

INNOVATIVE
COMPUTING LABORATORY

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE