State System and History for Trace Viewers

Alexandre Montplaisir
Michel Dagenais

December 9th, 2011
École Polytechnique de Montréal

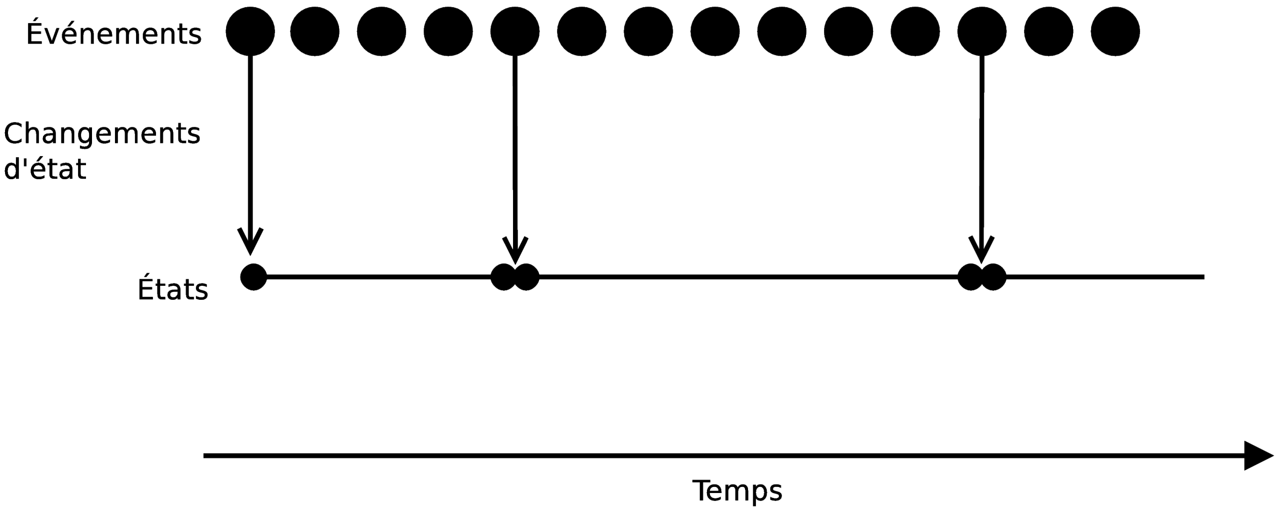
Contents

Definitions

- Event
  - Punctual record of an action that happened in the traced system, at a particular time. It has no duration.
- State (or *state interval*)
  - Record that has a start time and end time, hence a *duration*. We can describe each state with a *state value*.

Definitions (continued)

- State change
  - We can specify how events modify our model of the state. To do this, we assign *state changes* to certain types of events.



Definitions (continued)

- Attribute
  - ✓ Smallest unit of our model that can be in a particular state at a given time.
  - ✓ Can be referred to by its path in the *attribute tree*, or by its unique integer identifier (*quark*).

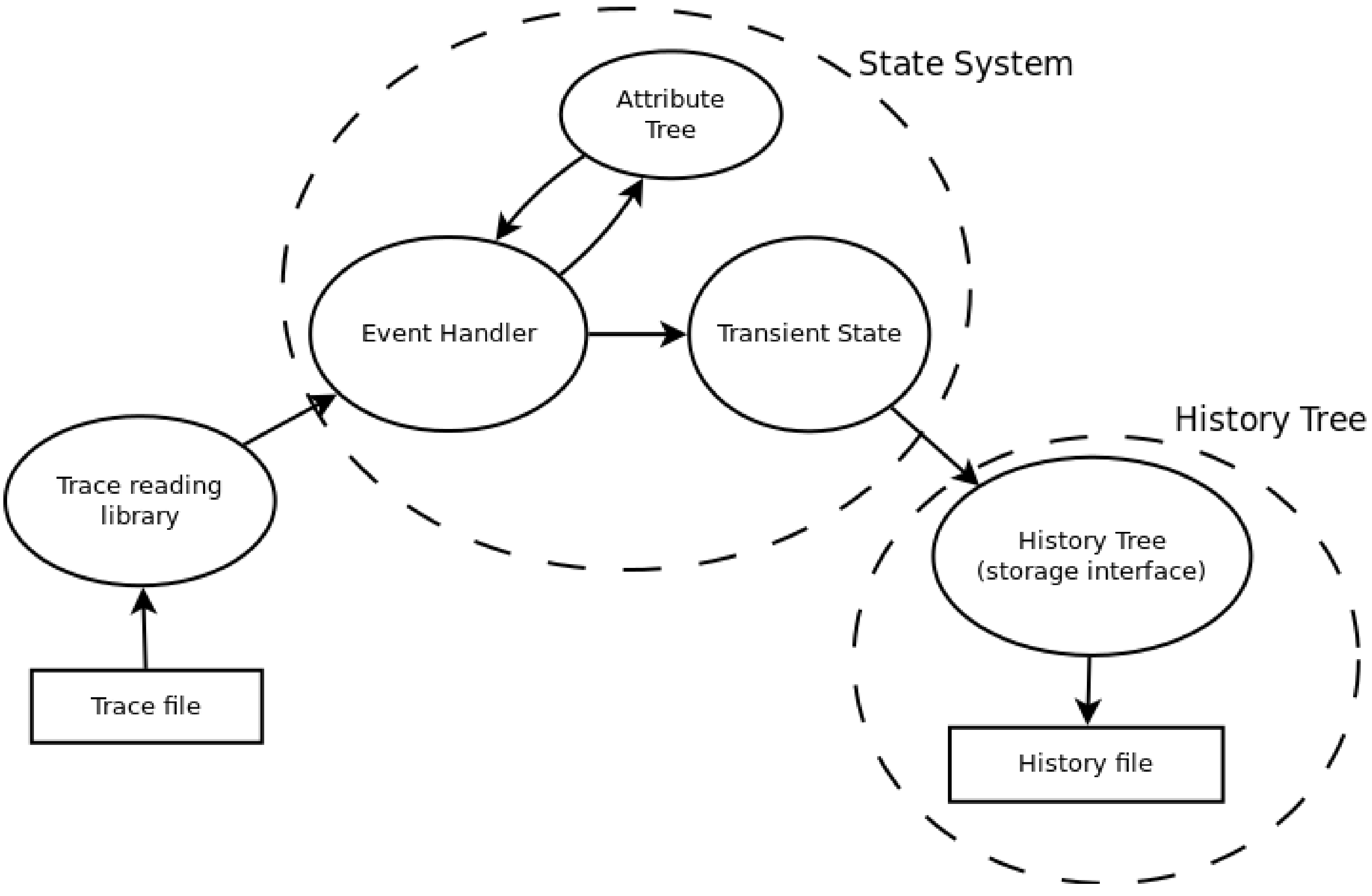
```
<hostname>
 \_ CPUs
 |    \_ CPU0
 |    |    \_ Current_thread
 |    |    \_ IRQ_stack
 |    |
 |    \_ CPU1
 |         ...
 |
 \_ Threads
      \_ PID1
      |    \_ PPID
      |    \_ TGID
      |    \_ Status
      |    \_ Executable_name
      |    \_ Exec_mode_stack
      |
      \_ PID2
        ...
```

Definitions (continued)

- Current State
  - The *current state* is the complete state of the (traced) system, as it was at a given time in the trace.
  - It is an array of *state values*, one for each attribute in the model (the index in the array corresponds to the *quark*).

- The role of the State System is to restore “current states”, for any given point in the trace.

The Complete State System



The Complete State System

- When building the state history the first time, we read through all the events from the trace.
- The *Event handler* is where we assign *state changes* to events. Those state changes are then sent to the Transient State.
- The *Transient State* represents the *Current State*, at the point where the reading descriptor is in the trace file. It is used to generate the state intervals.

The Complete State System
Event handler

- We can describe state changes with the following methods:

  modify(timestamp, state_value, attribute)

  remove(ts, attribute)

  push(ts, value, attribute)

  pop(ts, attribute)

  increment(ts, attribute)

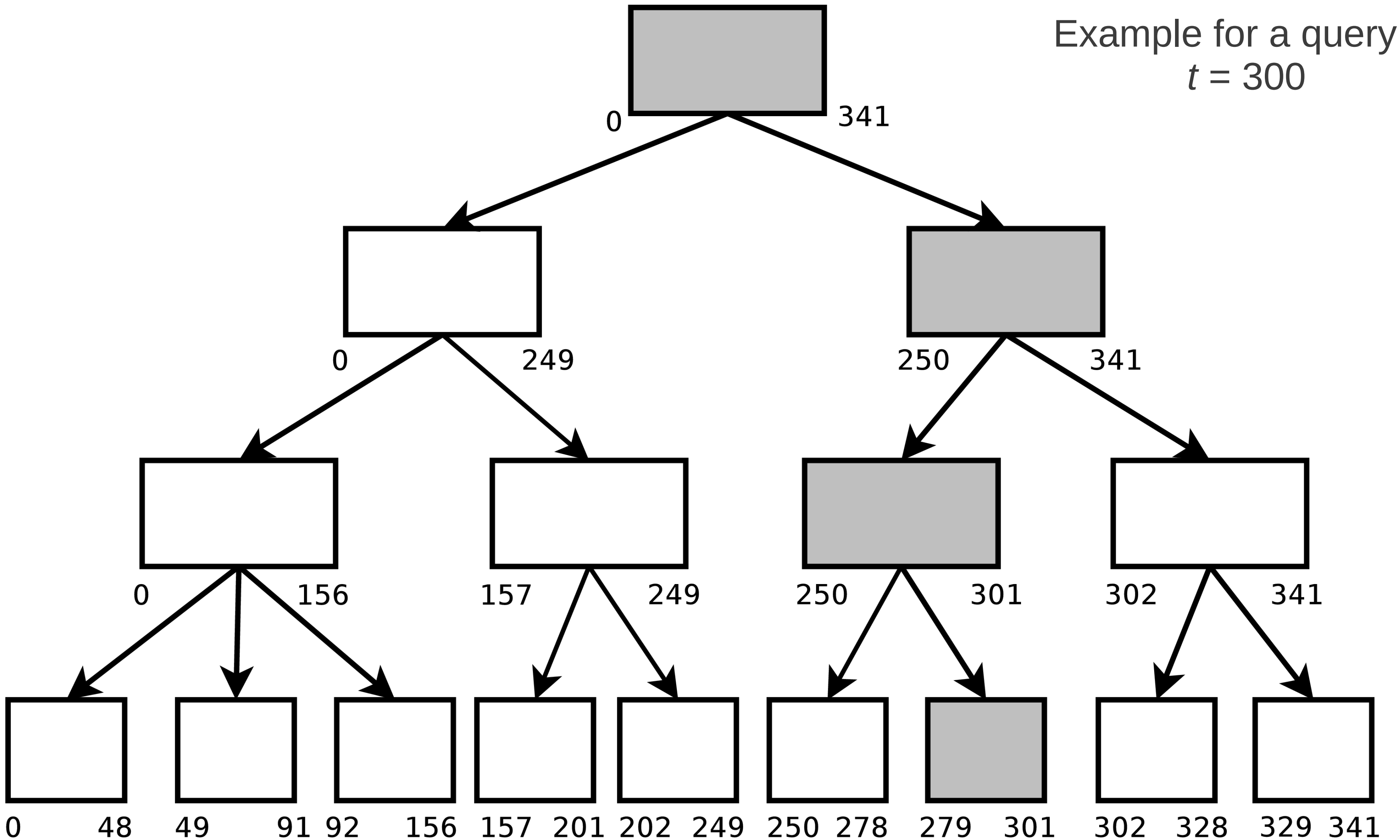
The Complete State System Event handler (example)

```
case LTT_EVENT_SCHED_SCHEDULE:
    /* Read information from the event payload */
    nextPid = (Long) event.getContent().getField(0).getValue();
    prevPid = (Long) event.getContent().getField(1).getValue();
    stateOut = (Long) event.getContent().getField(2).getValue();

    /* Set the status of the new scheduled process */
    ss.modifyAttribute(ts,
                       LTTV_STATE_RUN,
                       ["Threads", nextPid.toString(), "Status"]);

    /* Set the status of the process that got scheduled out */
    ss.modifyAttribute(ts,
                       stateOut.intValue(),
                       ["Threads", prevPid.toString(), "Status"]);

    /* Set the current scheduled process on the relevant CPU */
    ss.modifyAttribute(ts,
                       nextPid.intValue(),
                       ["CPUs", event.getCPU().toString(), "Current_thread"]);
    break;
...
}
```

The History Tree

- Data structure for intervals, optimized for disk storage.
- Intervals have to be inserted in ascending order of their end times (this is the case with intervals generated by the state system).
- Only one branch of the tree has to be explored for a *stabbing query*, which gives theoretical *O(log n)* scalability.

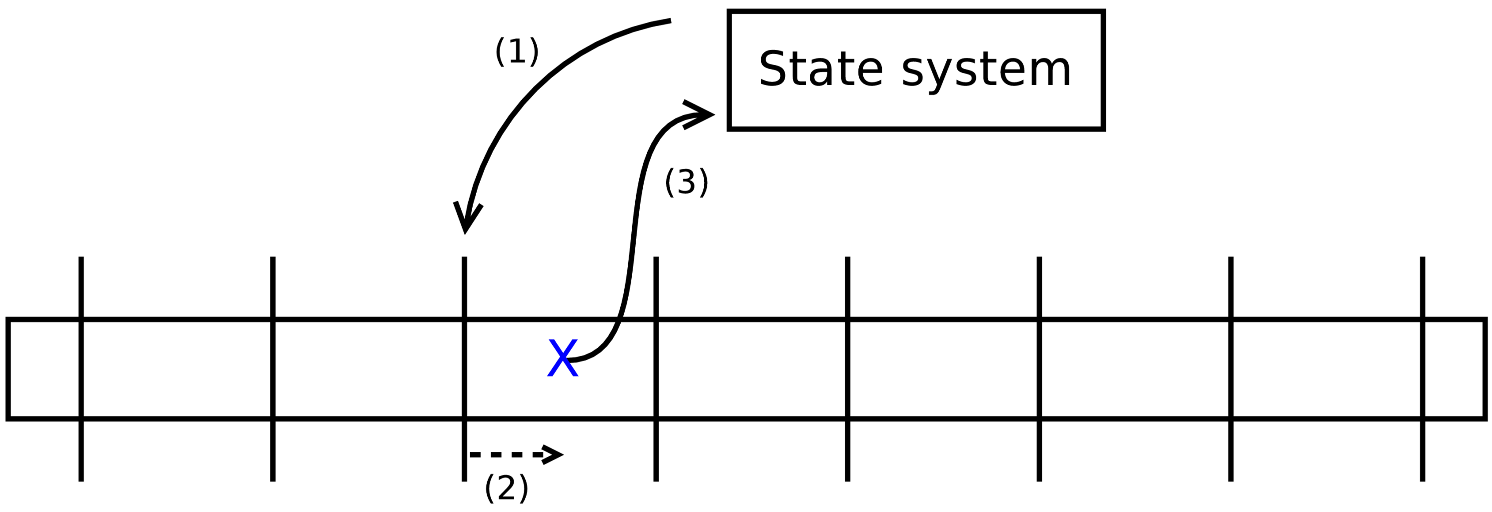
The History Tree

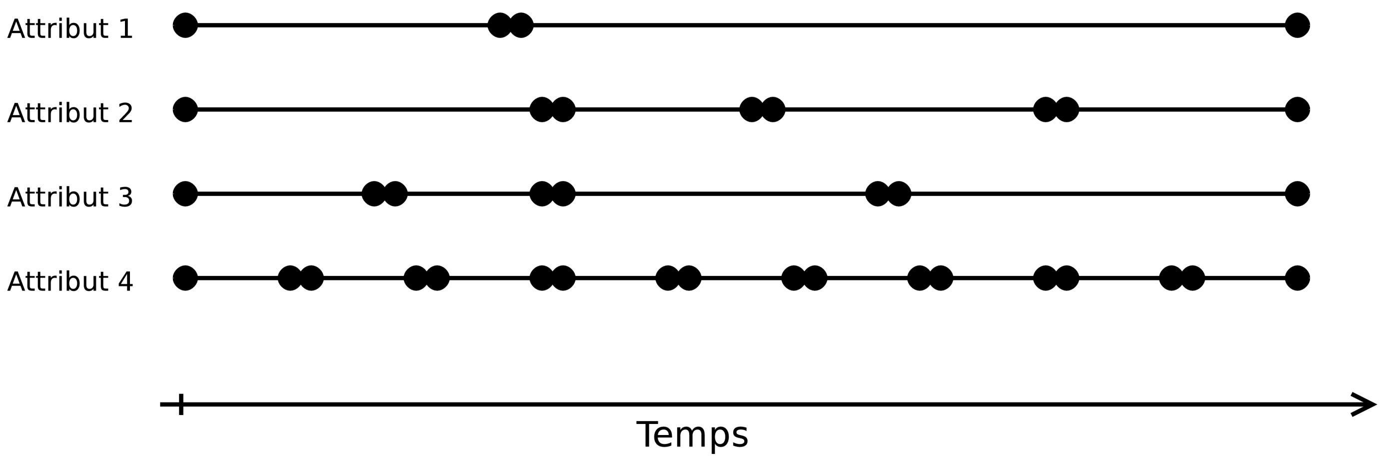


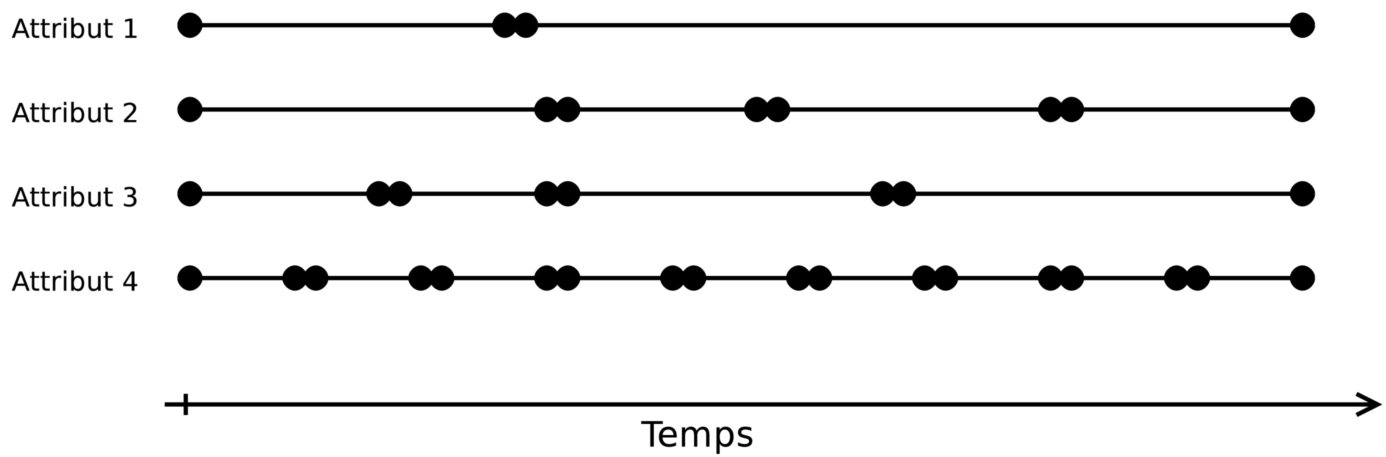
Example for a query at
t = 300

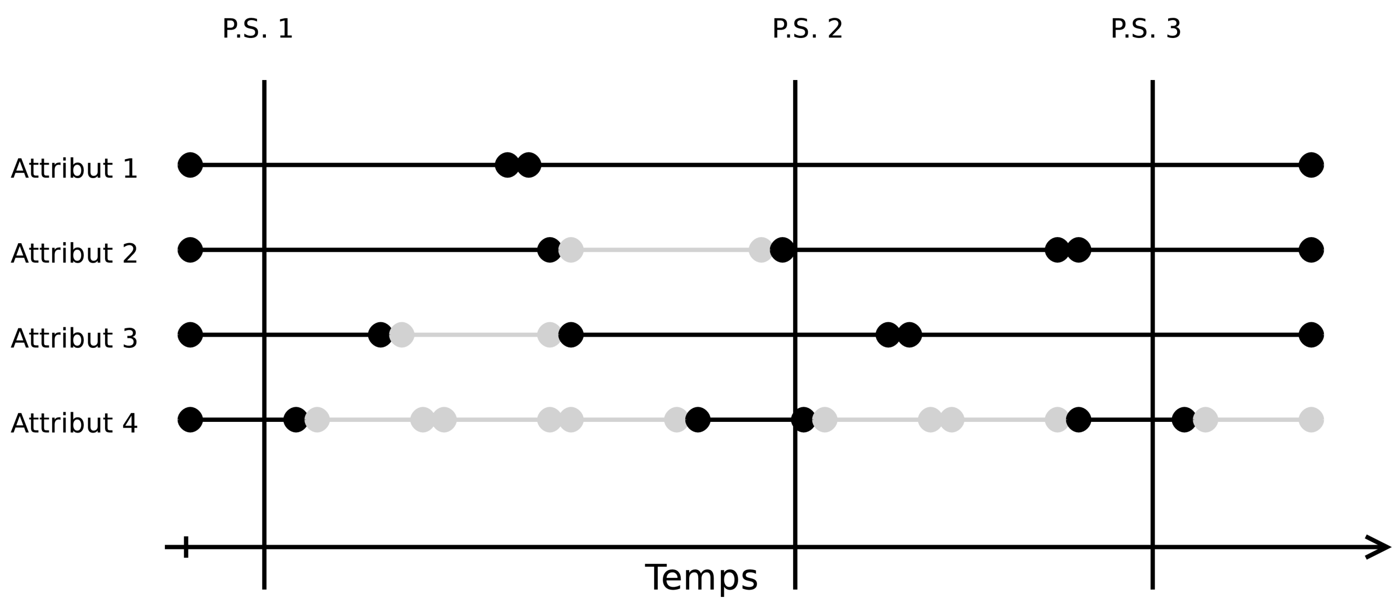
Partial History

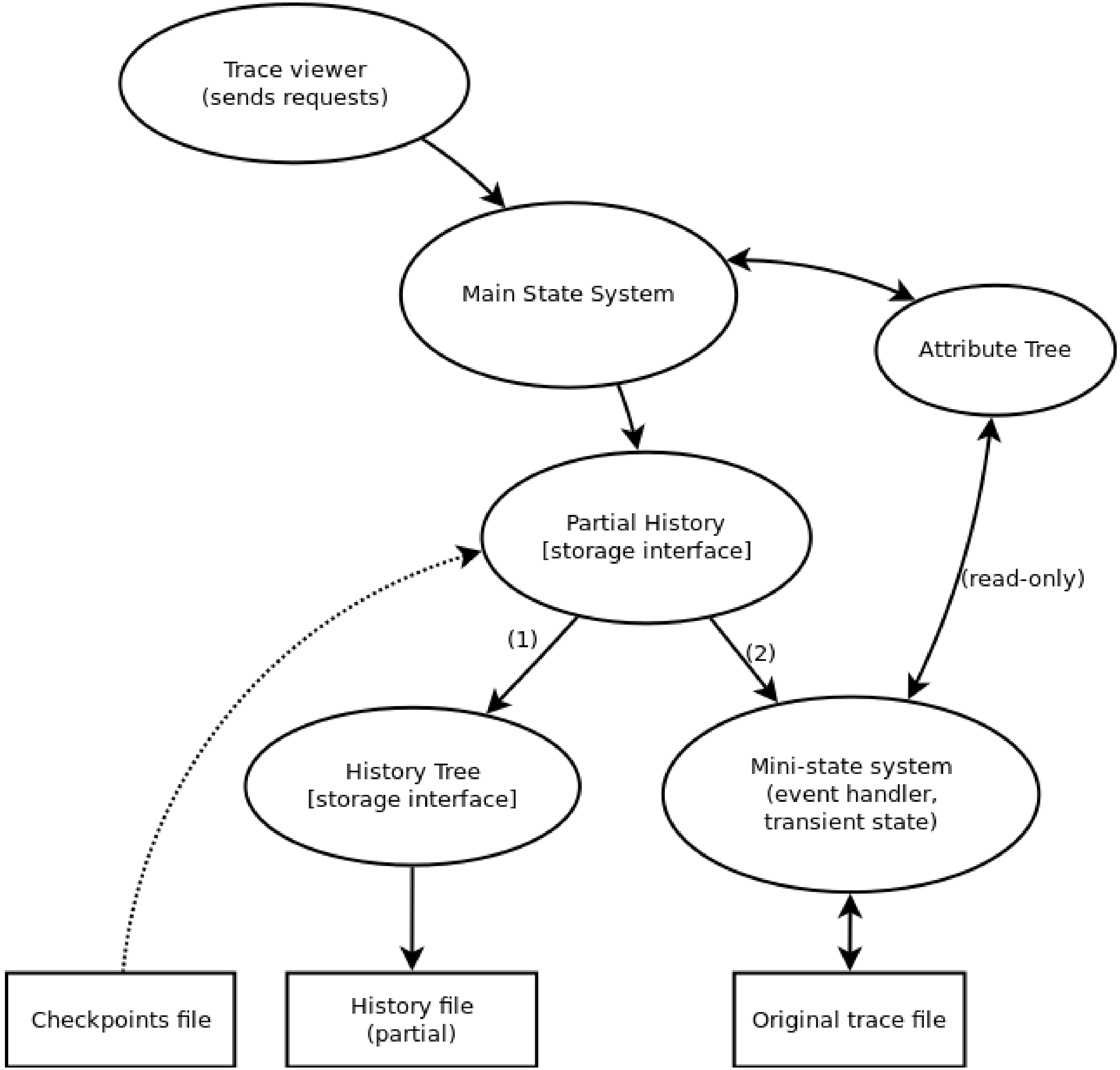
- Complete state histories could be very large (~2x the size of the original trace if we included statistics).
- What if we only store the complete state at checkpoints, then use the trace to regenerate the state at arbitrary times?



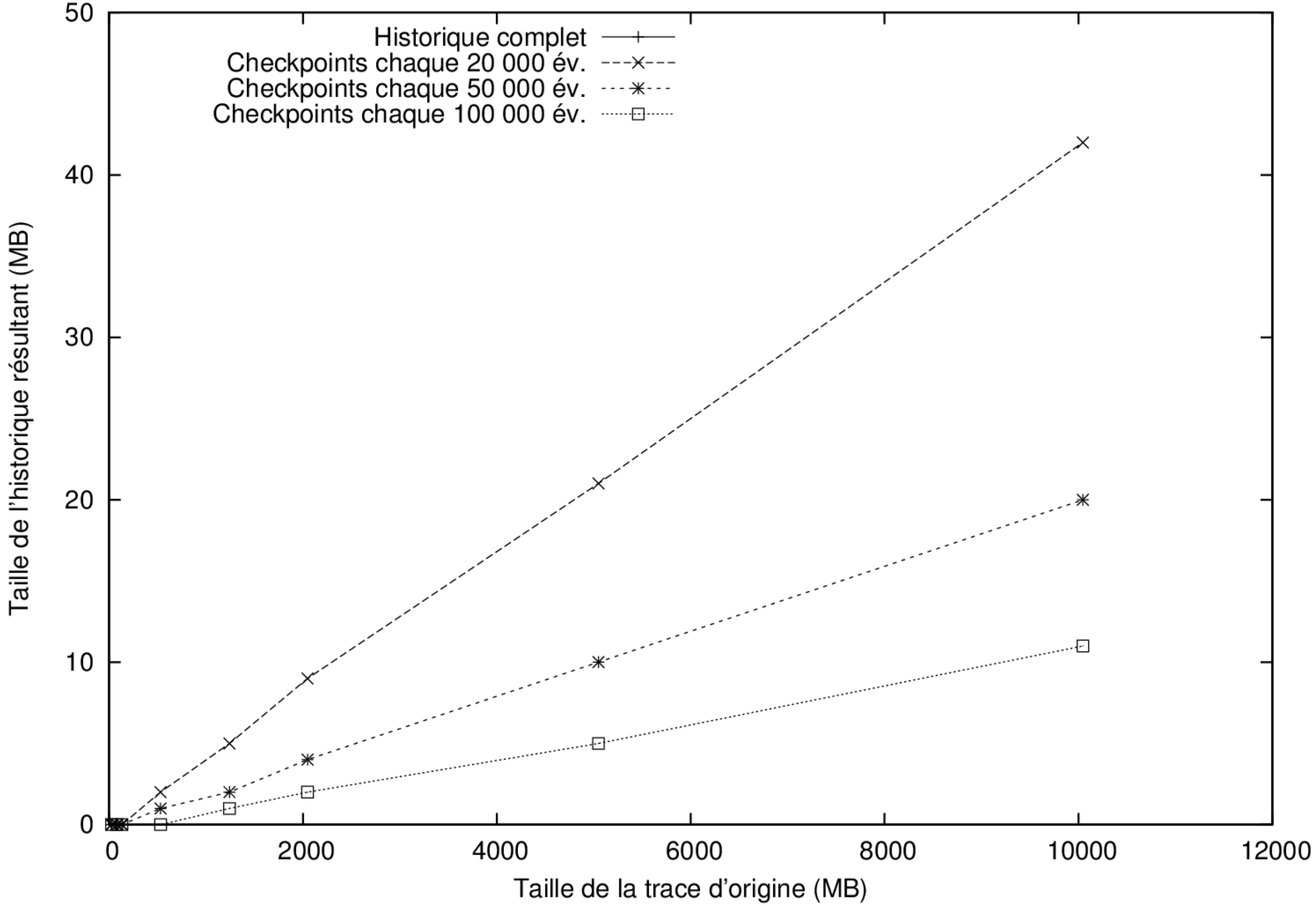

Attribut 1
Attribut 2
Attribut 3
Attribut 4
Temps


Attribut 1
Attribut 2
Attribut 3
Attribut 4
Temps

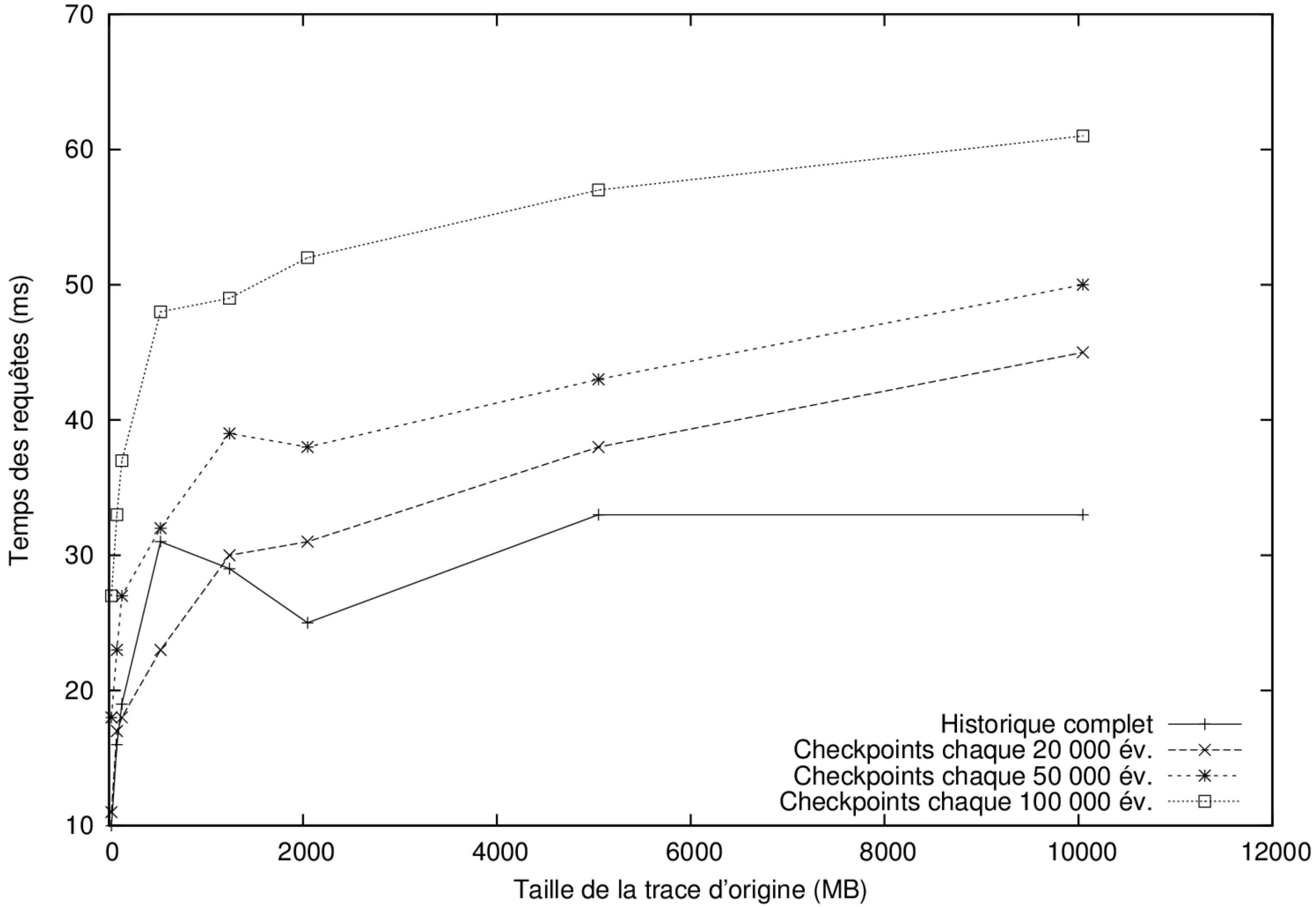

P.S. 1
P.S. 2
P.S. 3
Attribut 1
Attribut 2
Attribut 3
Attribut 4
Temps


Trace viewer
(sends requests)
Main State System
Attribute Tree
Partial History
[storage interface]
(read-only)
(1)
(2)
History Tree
[storage interface]
Mini-state system
(event handler,
transient state)
Checkpoints file
History file
(partial)
Original trace file

Partial History



Partial History



Partial History

- Compared to a complete history, a partial one:
  - Takes MUCH less space on disk
    (about a *thousand* times less!)
  - Query times increase, but stay well within the same order of magnitude (roughly doubles with a granularity of 100 000 events).
  - We need the original trace to be available.
  - We lose the ability to run *punctual queries* efficiently.

Conclusion

- I had many more things to show you!
  - Performance comparisons with generic R-Trees and a PostgreSQL database.
  - Hybrid storage
  - *Claudette* nodes
  - ...
- For more details you can read my thesis, which should (hopefully) be available in the coming months.

Questions?

Thank you!