Folding Methods for Event Timelines in Performance Analysis

IPDPS 2015 – HIPS Workshop

Hyderabad, India

Matthias Weber

May 25, 2015

Outline

Introduction
Timeline Folding Methods
Case Studies
Conclusions

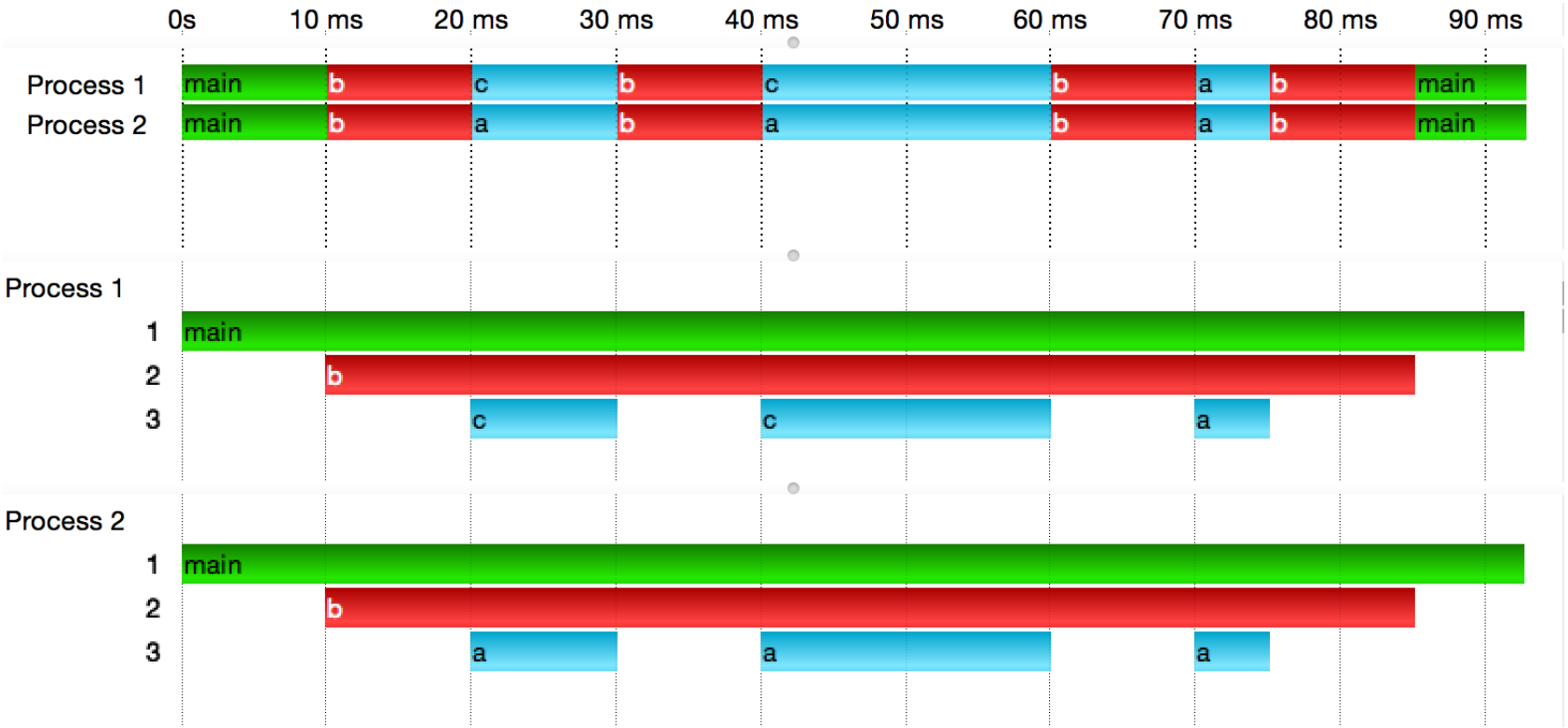
Performance Analysis in HPC

Performance optimization is important for efficient usage of HPC systems.

- The Linpack benchmark primarily achieves only about 60%-85% of the theoretical peak performance across top 10 systems.
- Highly optimized applications, nominated for the Gordon Bell Prize, achieve only 55% or 26% of the theoretical peak performance.
- For the majority of scientific applications this fraction is even below these numbers.

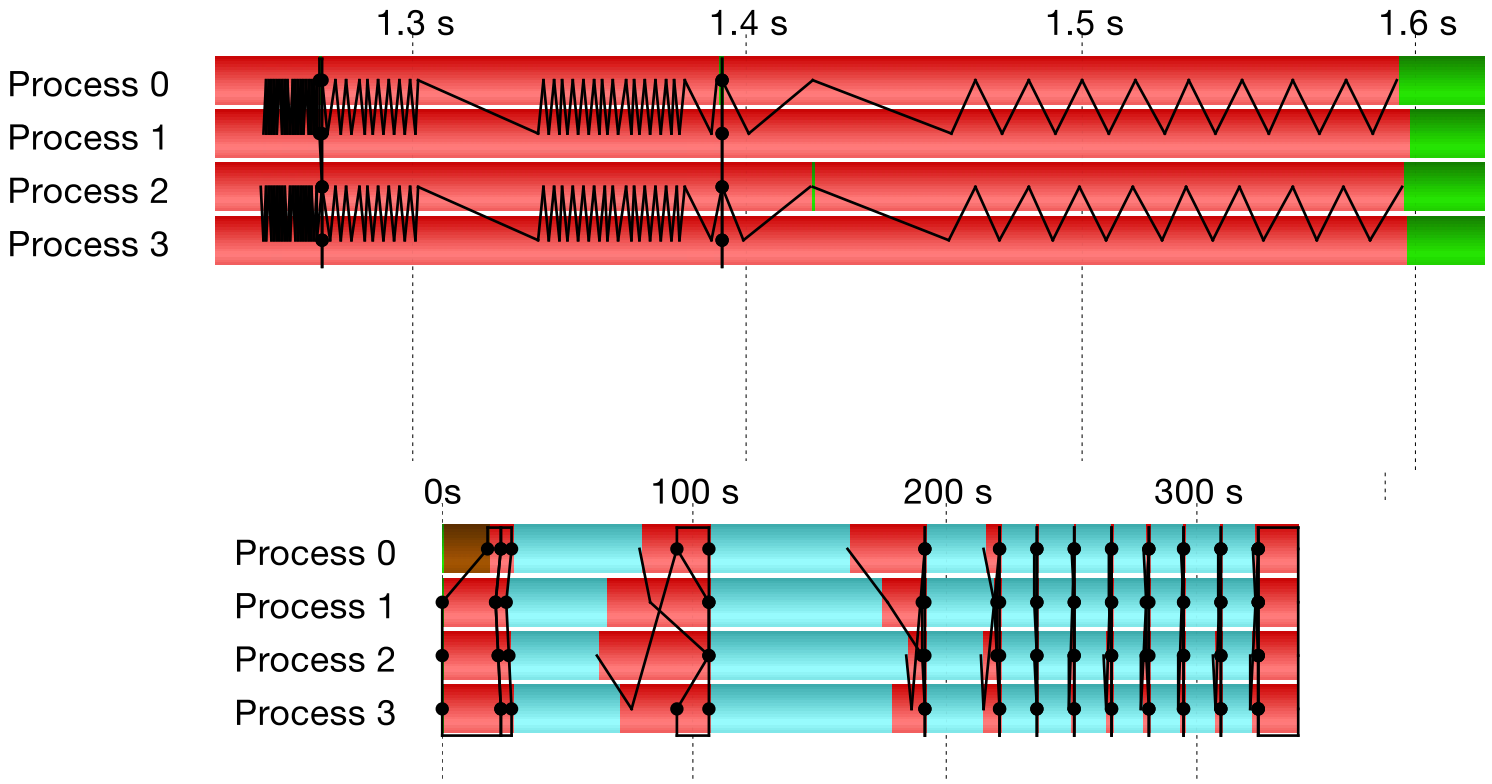
Due to rising complexity of hardware and software performance optimization is no trivial task.

- Tools are needed to support the developer.
- Traces and profiles are used to record performance data of applications.



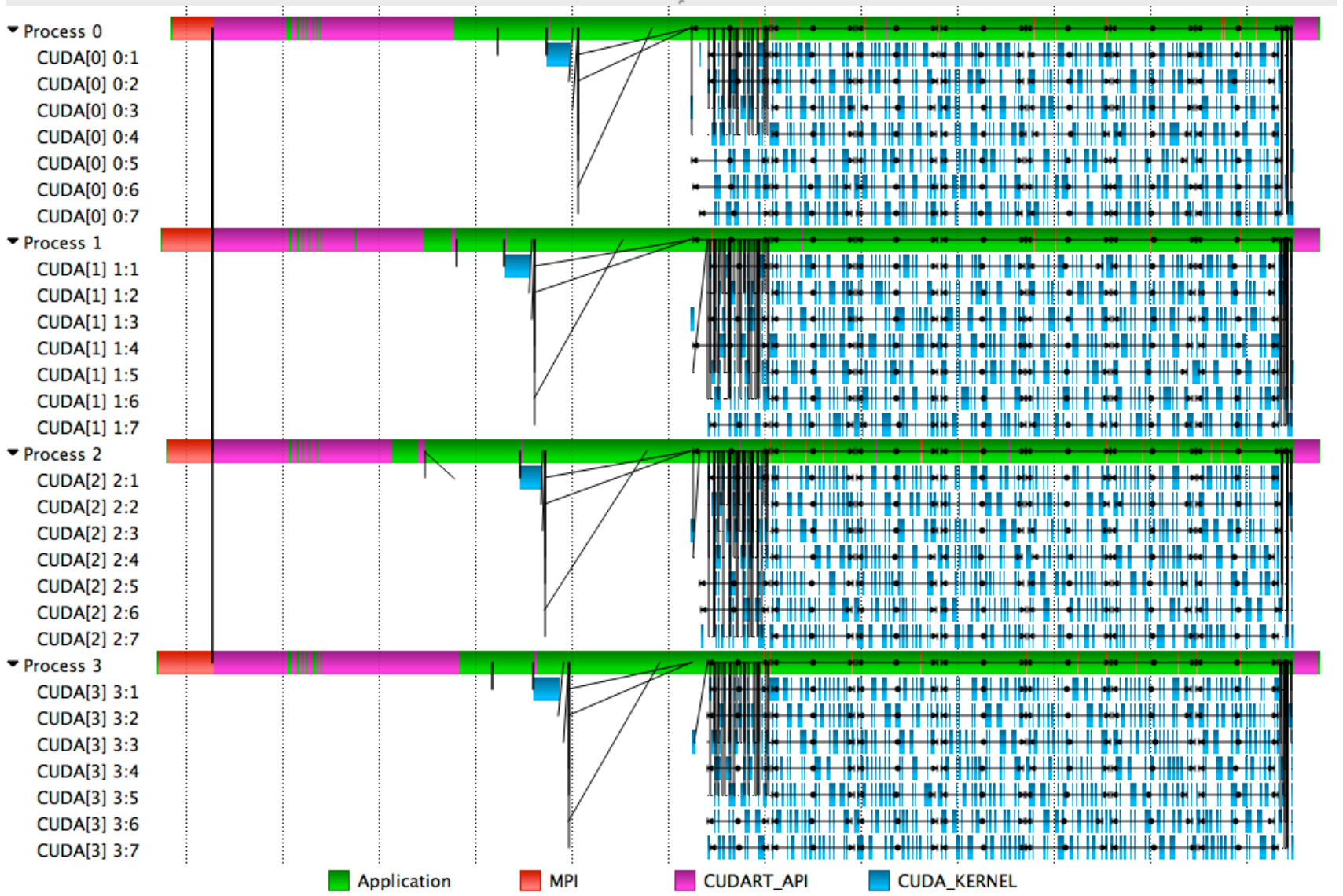
Timeline Visualization in Performance Analysis

- Horizontal lines visualize streams of application states (traces).
- Individual application states are represented with a color.
- A stream represents a logical processing element, e.g., a process, thread, or GPU pipeline.



Timeline Visualization in Performance Analysis

- Timelines are powerful for analysis of communication, imbalances, or simply to study an application execution.



Challenge: Visual Scalability

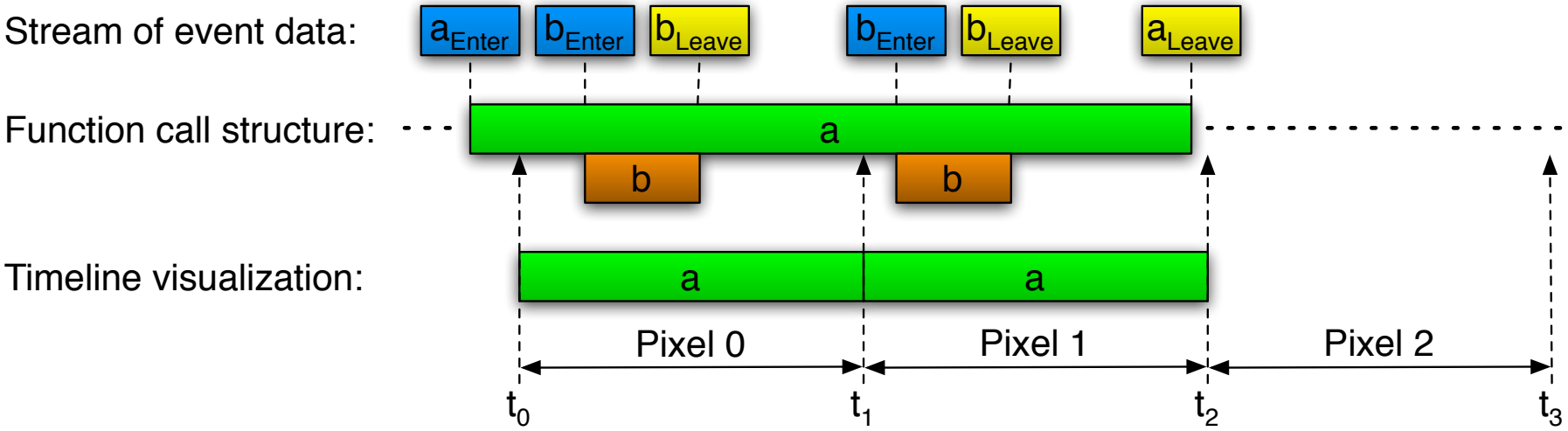
- The scheduling behavior in GPU pipelines renders visual inspection impractical.
- Visual evaluation of GPU device utilization is hard.

Challenge: Visual Scalability

- When visualizing parallel streams at least one vertical pixel is needed for a timeline.
- Since screen space is limited, tools often filter timelines.
- This may result in loss of potentially important information.

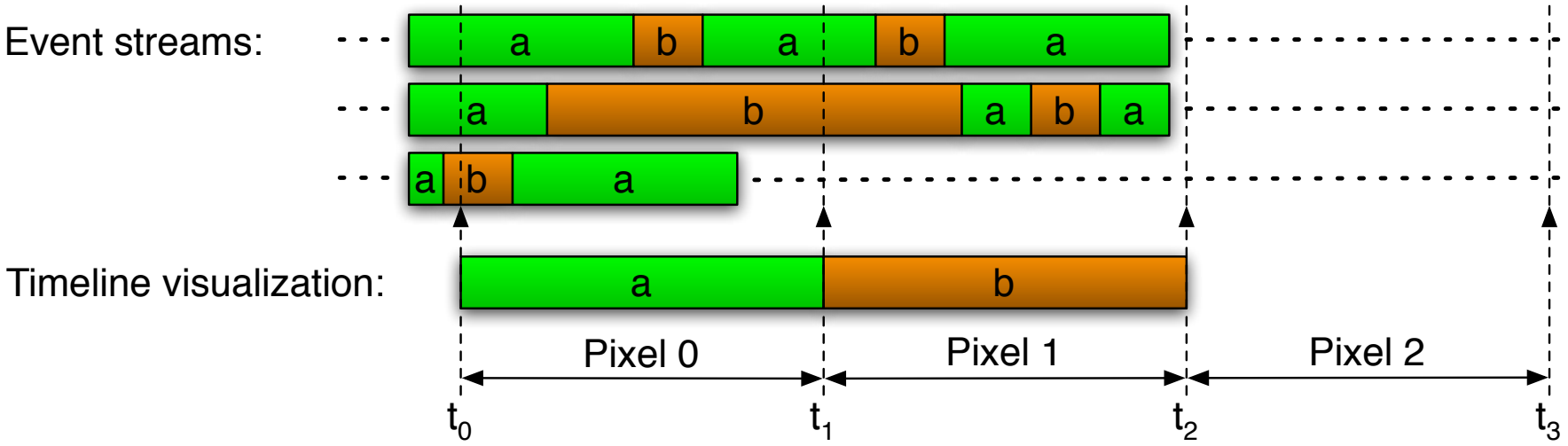
Outline

Introduction
Timeline Folding Methods
Case Studies
Conclusions



Constructing a Timeline – Horizontal Selection

- An association of state to pixel is needed.
- Choosing the state with the most time share in the interval is impractical with O(n) for n events.
- Sampling for event e with t_e<t can be done in O(log n) for n events.



Timeline Folding Operations – Vertical Aggregation

- For each sample point the visualization combines the states of each stream into a single result state.
- Therefore a selection operation must select a result state from a provided sequence of states, e.g., a, a, b for pixel 0.
- In the above example the state with the maximum number of occurrences is selected.

$$seq = (s_0, s_1, \ldots, s_n)$$

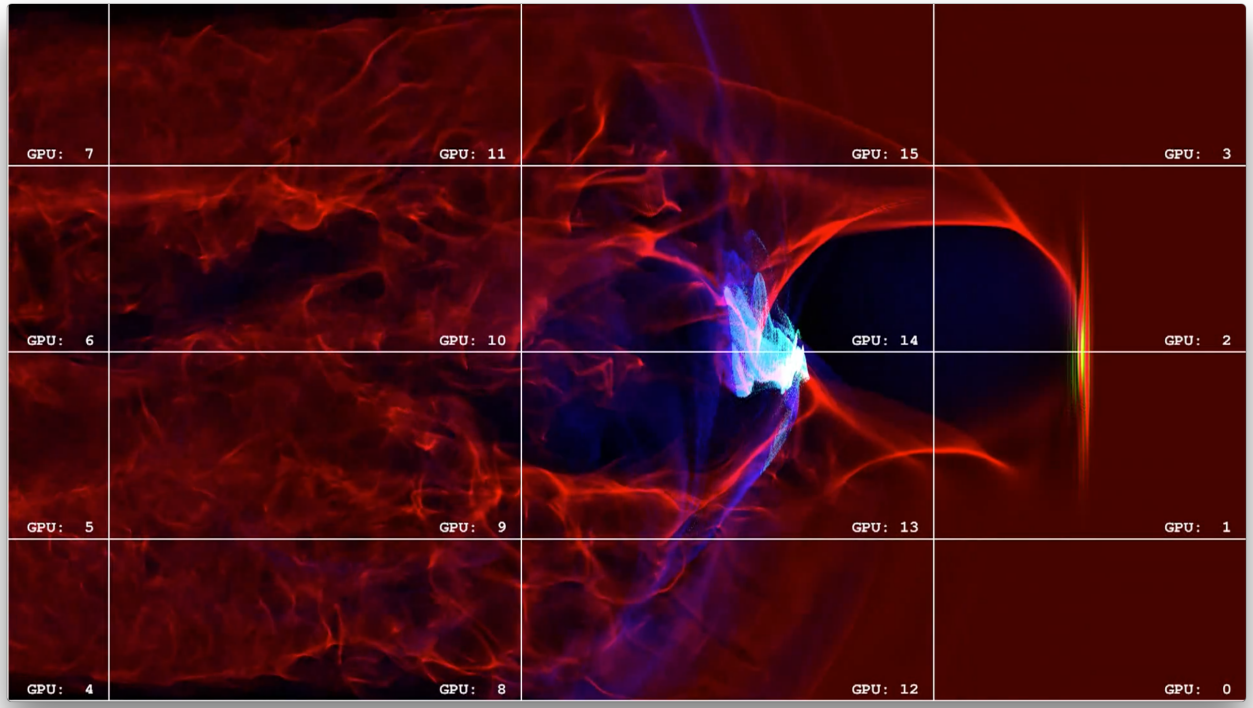
- $select_{max}(seq) := s_i$ with for all $0 \leq k \leq n$ holds
  $count(s_k, seq) \leq count(s_i, seq)$
- $select_{min}(seq) := s_i$ with for all $0 \leq k \leq n$ holds
  $count(s_k, seq) \geq count(s_i, seq)$
- $select_{diff}(seq) := \begin{cases} no\text{-}state & \text{if for all } i \in \{0, \ldots, n\} \\ & \text{holds } s_0 = s_i, \\ select_{min} & otherwise \end{cases}$
- $select_{idle}(seq) := s_i$ with for all $s_k \neq no\text{-}state$ holds
  $count(s_k, seq) \leq count(s_i, seq)$
  $(0 \leq k \leq n)$

Timeline Folding Operations – Paradigm Specific Folding

- *Default:* $select_{max}$ represents dominant behavior across all folded streams
- *Threads and processes:* $select_{diff}$ and $select_{min}$ identification of outliers; $select_{diff}$ replaces regions in which all streams exhibit the same state by no-state, in case of differing regions $select_{min}$ is applied
- *Accelerator paradigms:* $select_{idle}$ behaves like $select_{max}$, except that it only returns *no-state* when all input streams are idle

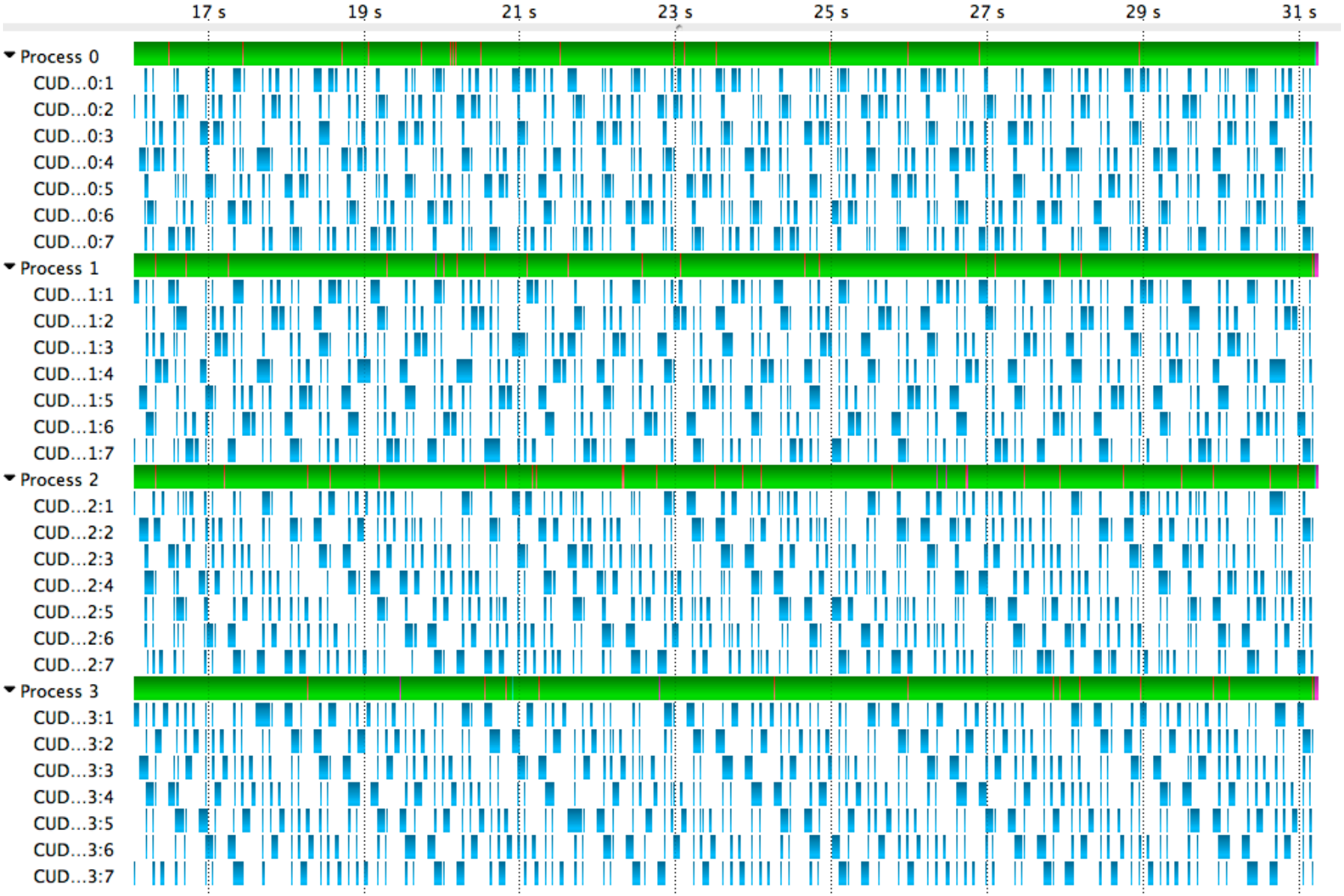
Outline

Introduction
Timeline Folding Methods
Case Studies
Conclusions



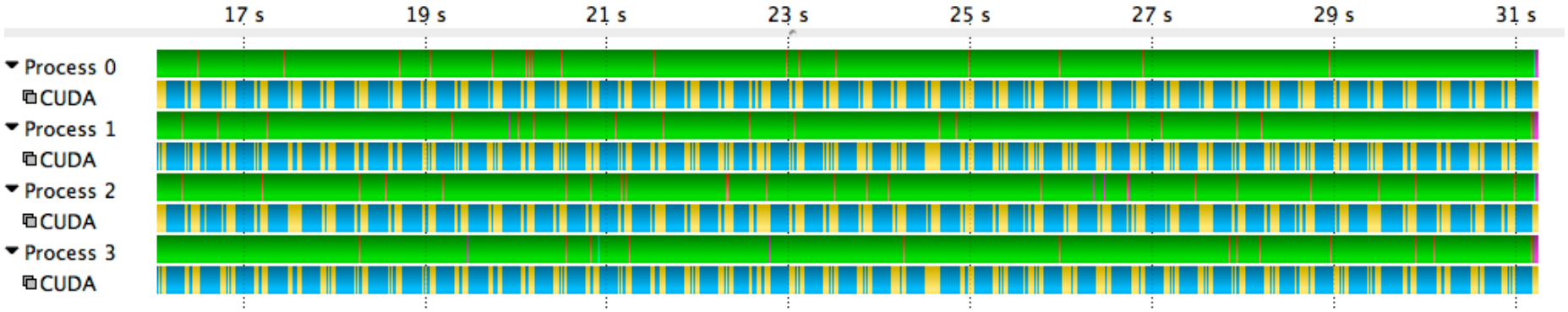
Case Study: PIConGPU

- PIConGPU computes a fully relativistic Particle-In-Cell (PIC) algorithm, which is widely used in computational plasma physics.
- MPI-based parallelism on compute node level.
- Graphics processing units (GPUs) compute large-scale plasma simulations.



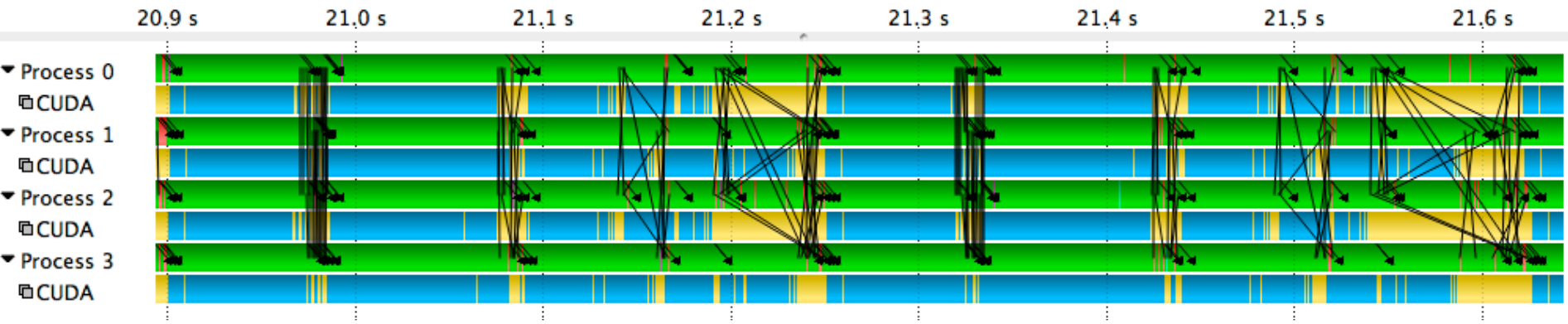
GPU Stream Folding

All device streams visualized independently lead to a *colored noise* effect.



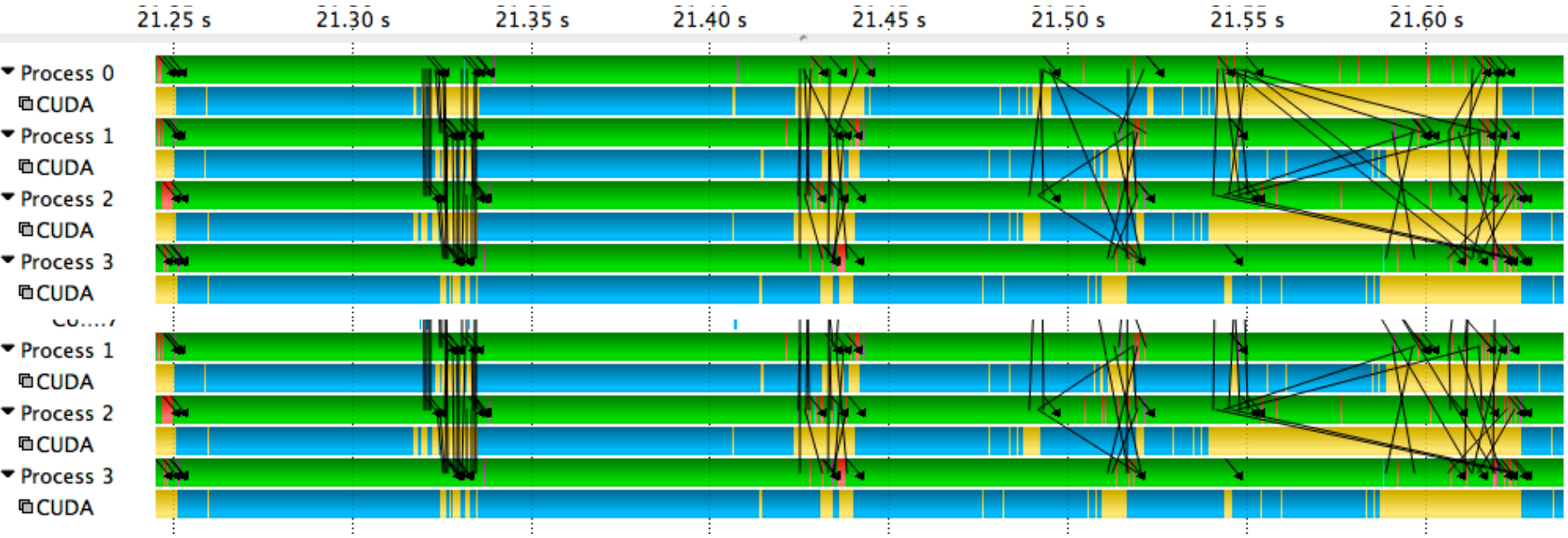
GPU Stream Folding

Using the $select_{idle}$ operation to fold all respective device streams results in an easily visible computational structure.



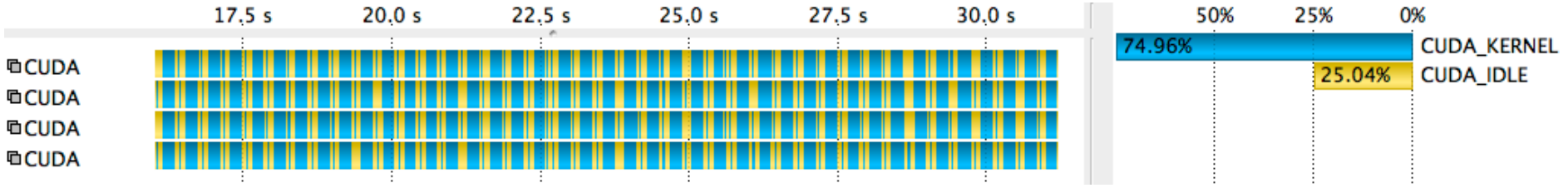
Detailed GPU Stream Analysis

- Two iterations of PIConGPU.
- Device idle times are clearly visible as yellow areas.
- The uneven yellow areas at the end of each iteration indicate load imbalances between devices.
- Communication (black lines) shows that device idle times correlate with communication phases between host processes and CUDA devices.



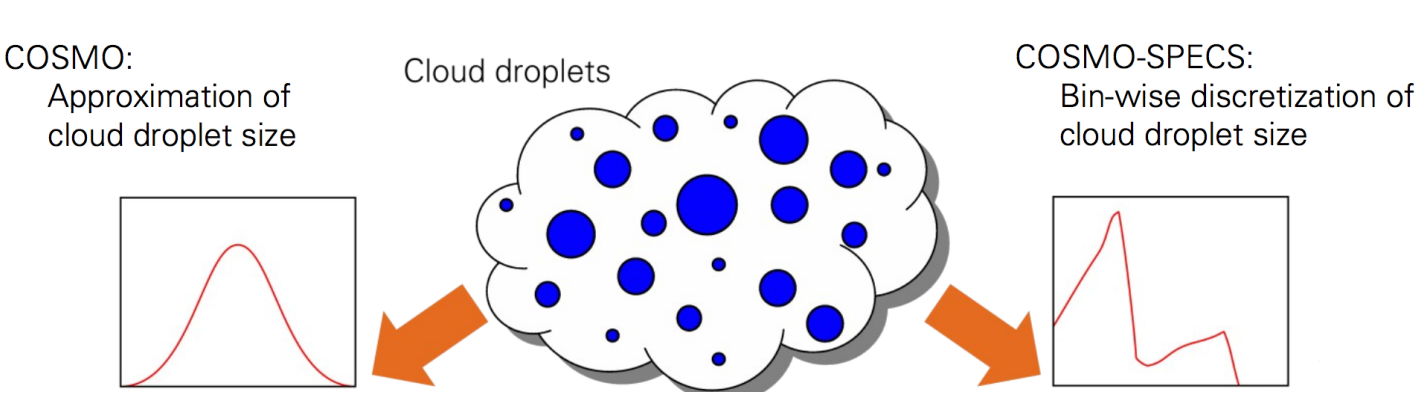
Detailed GPU Stream Analysis

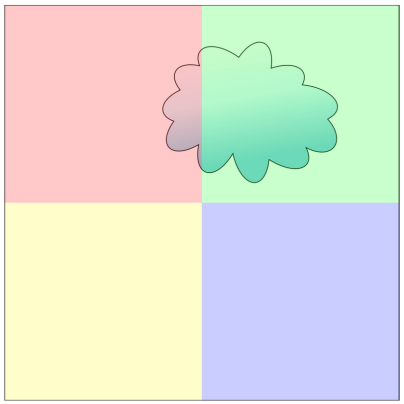
- One iteration of PIConGPU.
- Unfolding of CUDA groups reveals communication between host processes and the individual device streams.
- Also the scheduling of activities across the device streams is visualized.



CUDA device utilization

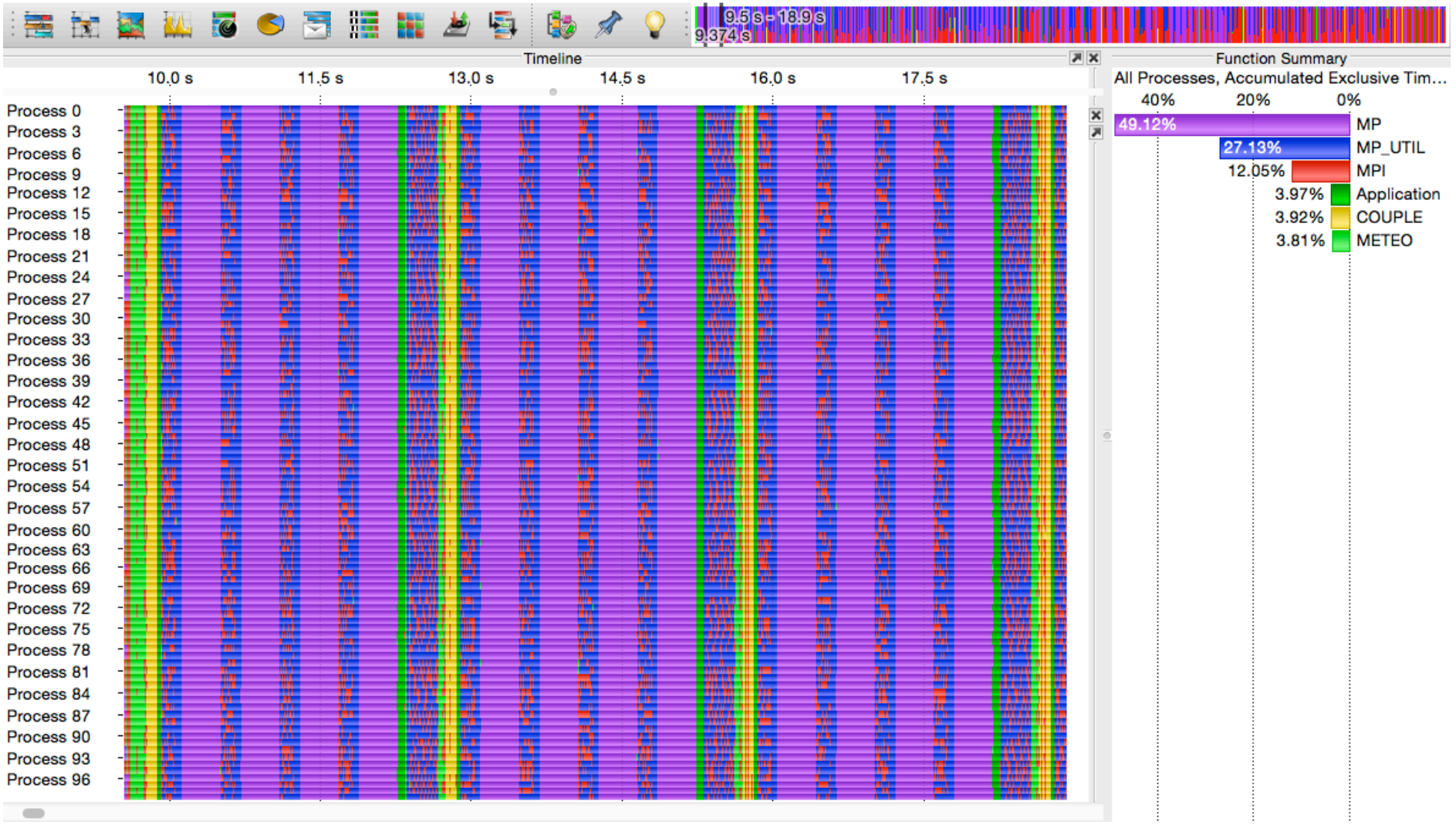
- Statistic allows computation of the CUDA device utilization.





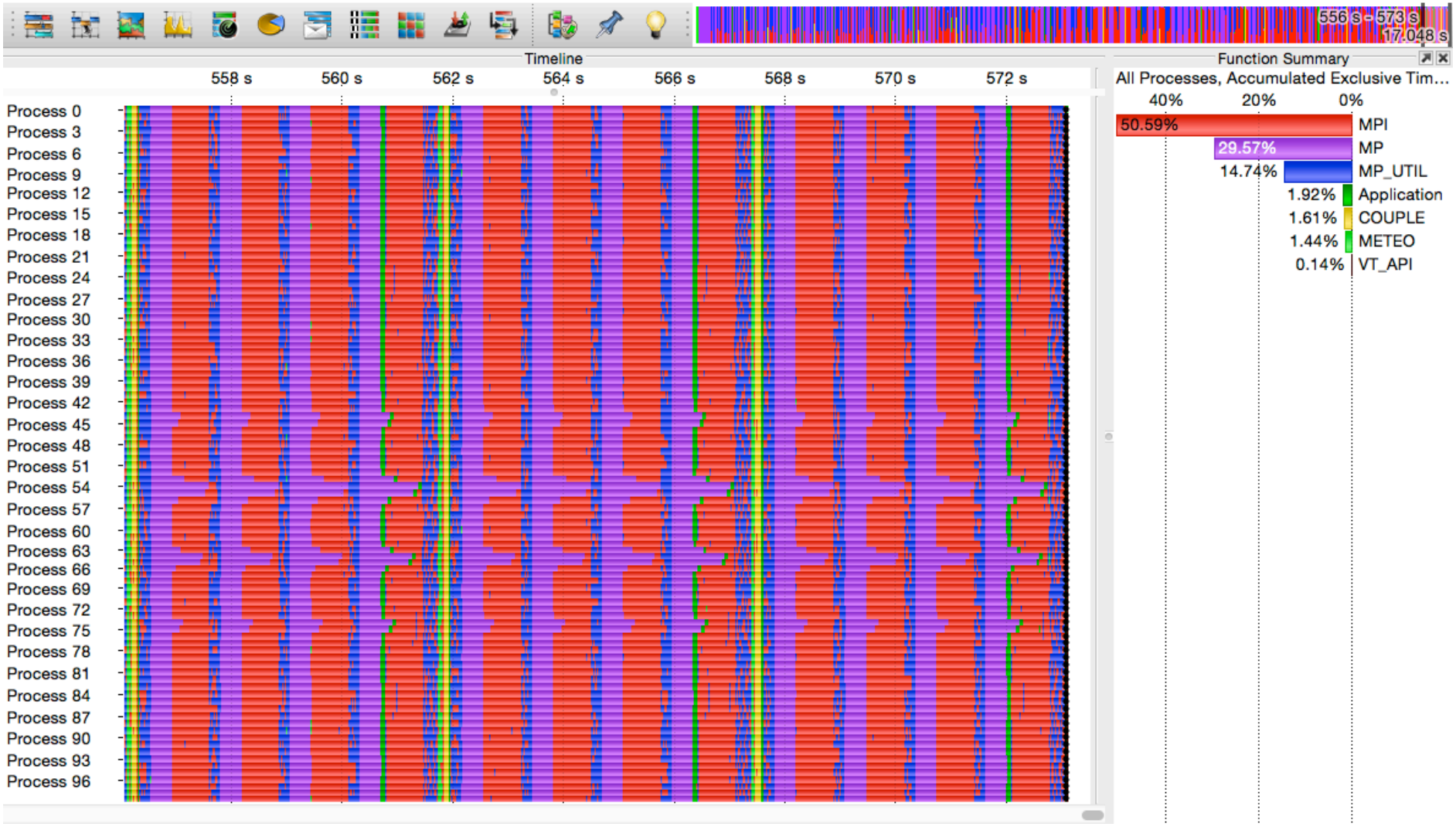
Case Study: COSMO-SPECS

- A weather forecast code that couples two models, COSMO and SPECS.
- COSMO is the regional weather forecast model.
- SPECS is a detailed cloud microphysics model.
- COSMO employs a 2D (horizontal) decomposition into M × N domains (without dynamic load balancing).
- SPECS uses the same data decomposition as COSMO and only replaces its cloud microphysics.



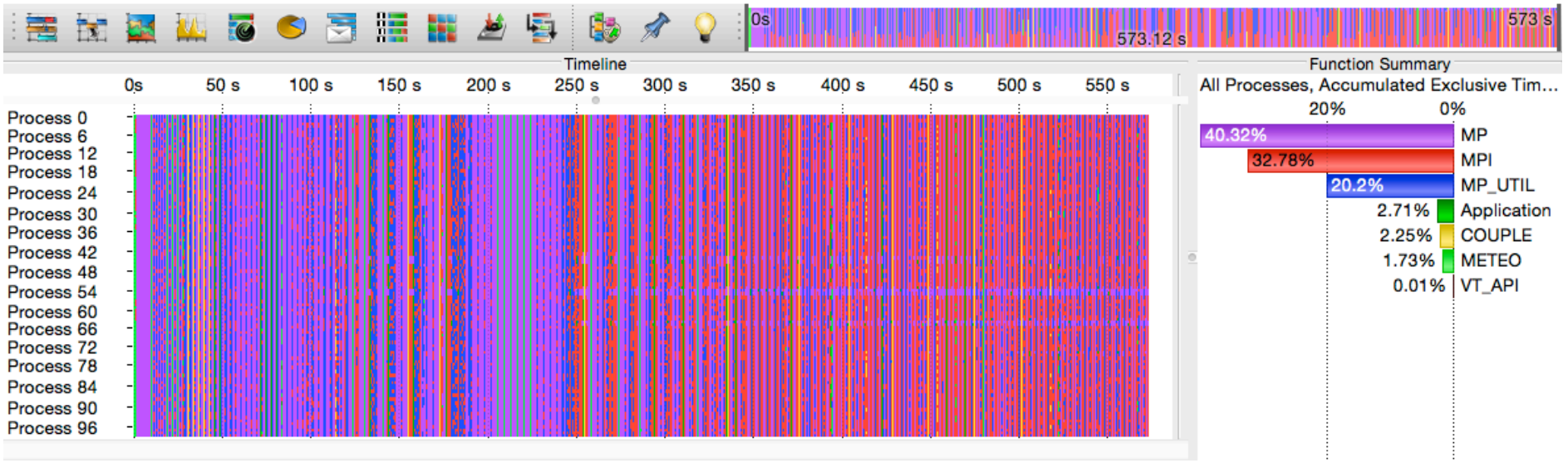
COSMO-SPECS: Common timeline visualization

- First three iterations.
- Compared to COSMO (METEO), the SPECS (MP, MP_UTIL) calculations are significantly more compute intensive.



COSMO-SPECS: Common timeline visualization

- Last three iterations.
- During the execution, SPECS introduces large load imbalances, since its computational cost heavily depends on the presence of cloud particles in the grid cell.

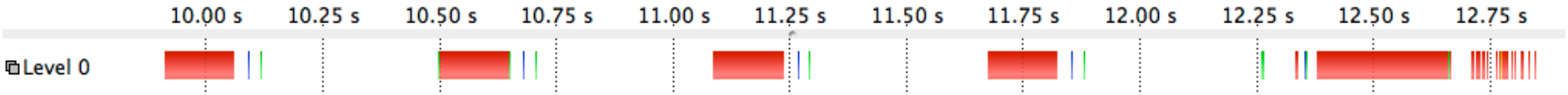


COSMO-SPECS: Common timeline visualization

- The load-imbalance is barely visible.
- With more than only 100 processes it might not be visible/ detectable at all.

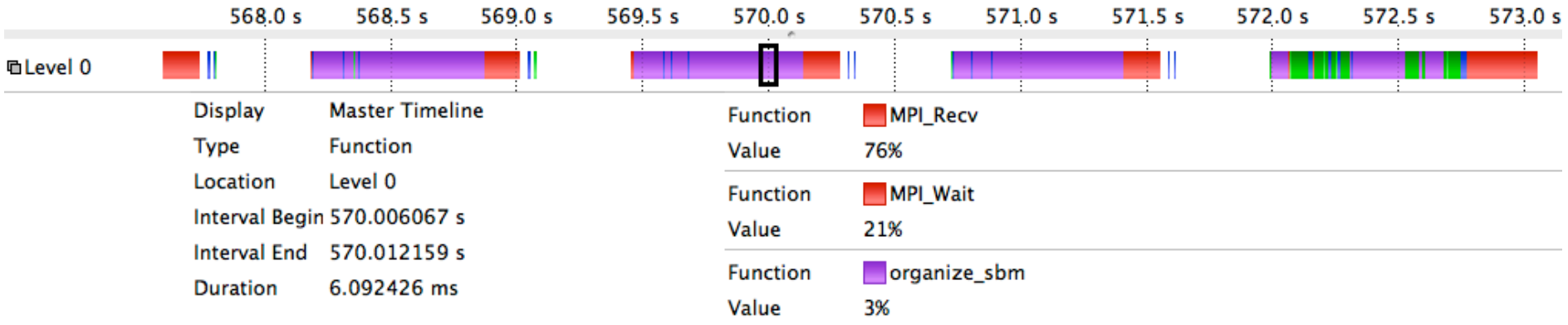
Folded Timelines of COSMO-SPECS

- Visualization of the full application execution.
- Using $select_{diff}$ folding operation to detect imbalances.
- The colored portions indicate areas in that processes execute different functions.
- The frequency of such areas increases over time.



Folded Timelines of COSMO-SPECS

- Visualization of the first iteration.
- Red areas indicate the exchange of simulated values between processes.
- The workload is still balanced and computations are executed in the white areas.



Folded Timelines of COSMO-SPECS

- Visualization of the first iteration.
- Large purple areas are visible.
- In that areas most processes are waiting and only a few processes are computing.

Conclusions

- Visual performance analysis is challenged by increasing application scale and increased complexity in parallelization, while screen space remains limited.
- Fast, simple, and intuitive folding operations can help answer common questions during performance analysis.
- Our folding strategies for timelines help achieving visual scalability and support easy detection of performance problems in parallel applications.