Automatic Detection of Performance Anomalies in Task-Parallel Programs

Work in progress on the Aftermath trace analysis tool

Andi Drebes

Université Pierre et Marie Curie
Laboratoire d'Informatique de Paris VI
andi.drebes@lip6.fr

Joint work with:
Antoniu Pop, Karine Heydemann
Albert Cohen, Nathalie Drach

RACING'14, May 30th, 2014

Context

Hardware and software environment

- Multi-core / many-core systems
- How to exploit the hardware efficiently?
- Task-parallel languages based on fine-grained tasks

Context

Hardware and software environment

- Multi-core / many-core systems
- How to exploit the hardware efficiently?
- Task-parallel languages based on fine-grained tasks

Performance debugging

- Requires analysis of complex interactions at execution time: Application / Run-time / Machine
- Possible solution: Record dynamic events to a trace file
- Post-mortem (offline) analysis

Context

Hardware and software environment

- Multi-core / many-core systems
- How to exploit the hardware efficiently?
- Task-parallel languages based on fine-grained tasks

Performance debugging

- Requires analysis of complex interactions at execution time: Application / Run-time / Machine
- Possible solution: Record dynamic events to a trace file
- Post-mortem (offline) analysis

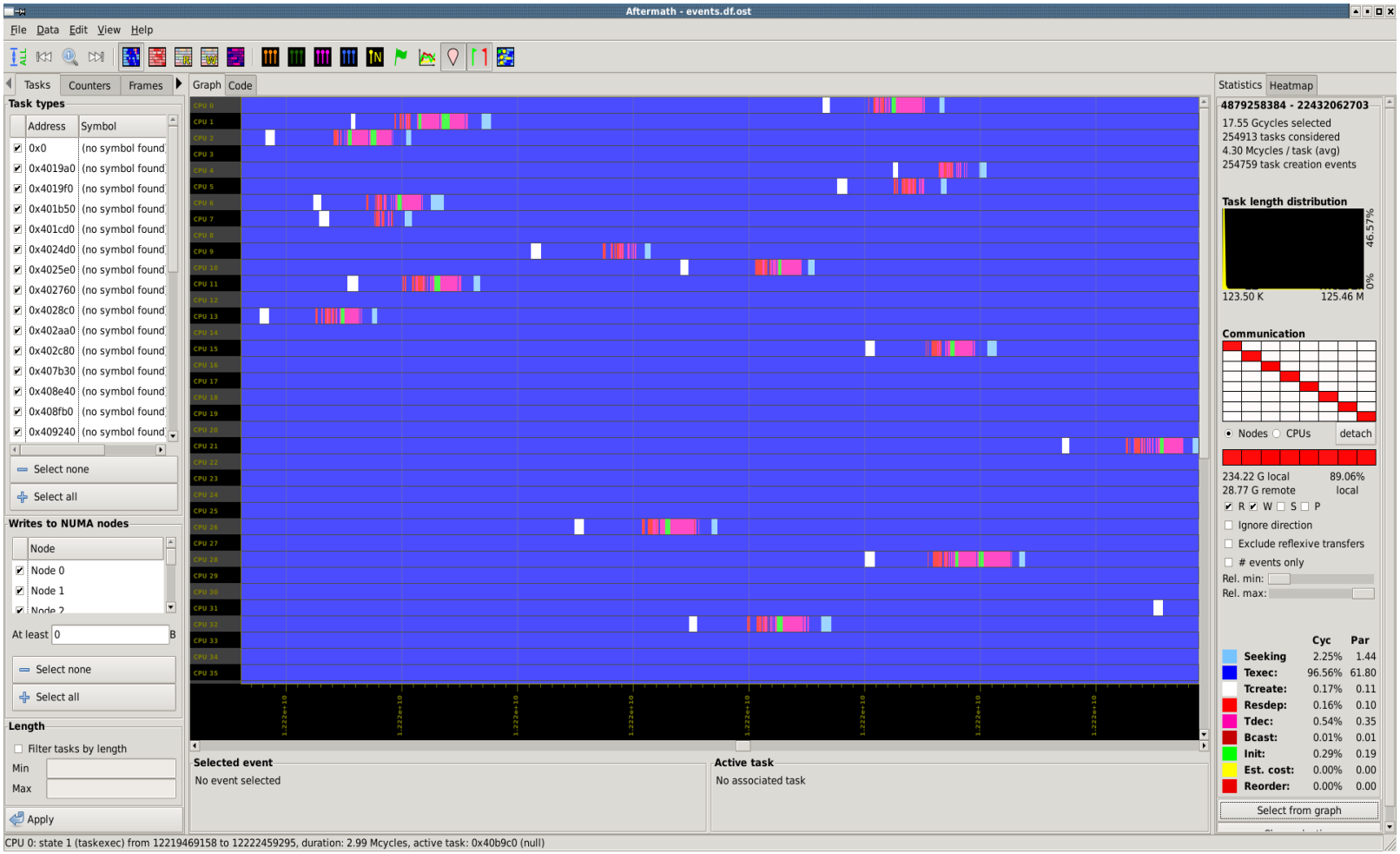
Aftermath

- Trace visualization and support for manual analysis
- Originally developed for OpenStream language & run-time
- Work in progress: Automate repetitive tasks

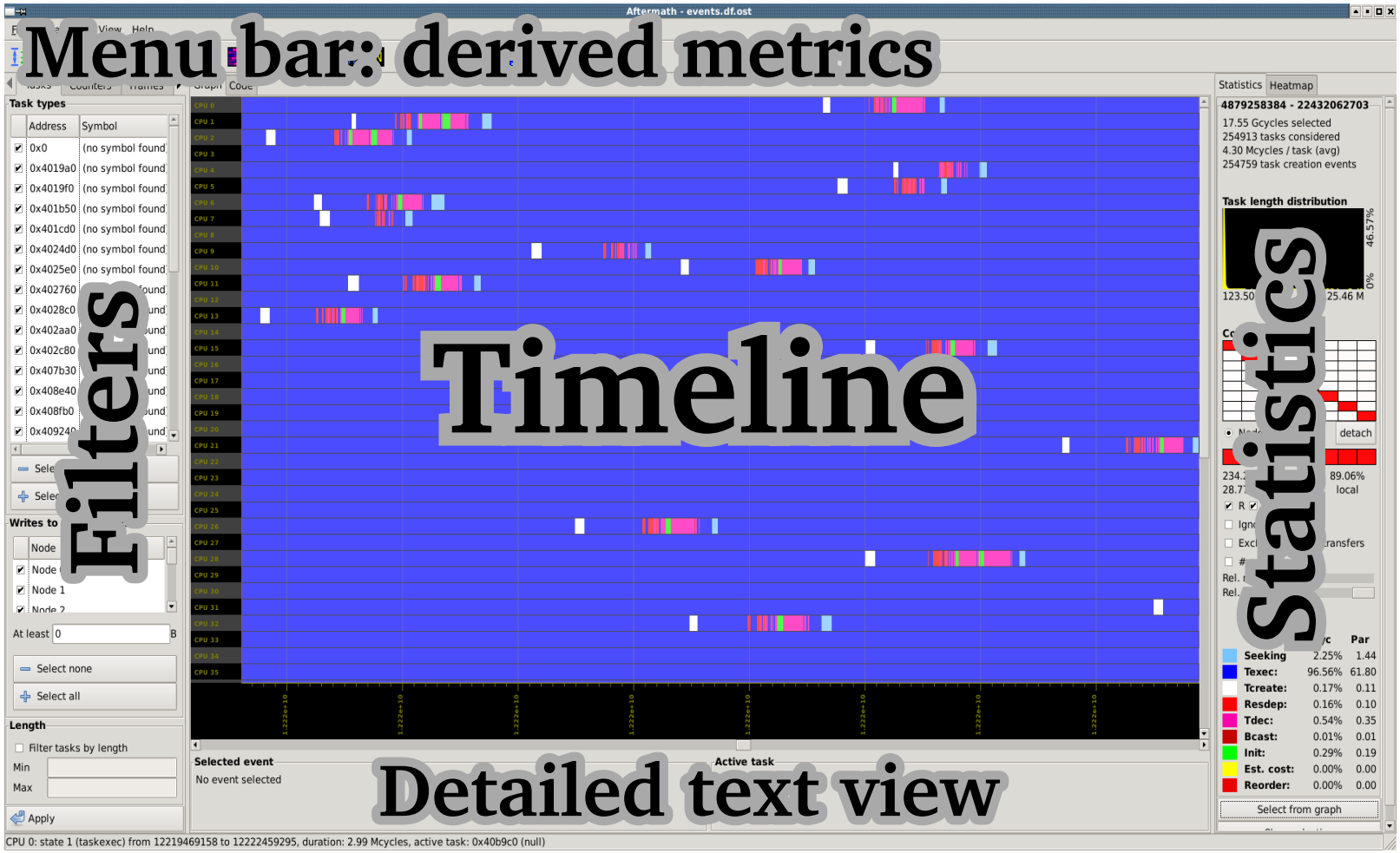
Outline

1. Aftermath
2. Insufficient parallelism and its causes
3. Performance anomalies during task execution
4. Work in progress: Status
5. Summary & Questions

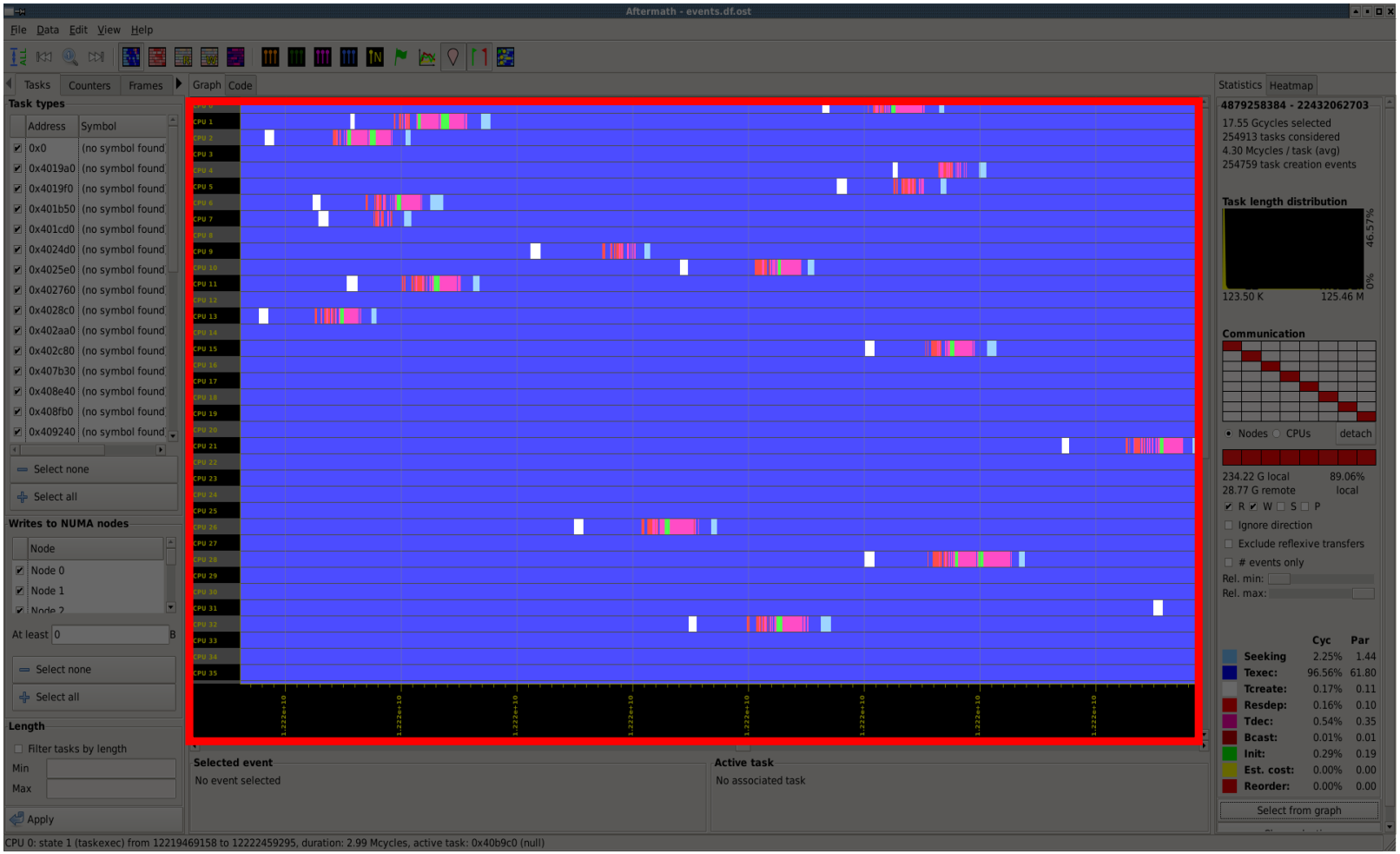
Aftermath


Aftermath - events.df.ost
File Data Edit View Help
Tasks Counters Frames
Task types
Address Symbol
0x0 (no symbol found)
0x4019a0 (no symbol found)
0x4019f0 (no symbol found)
0x401b50 (no symbol found)
0x401cd0 (no symbol found)
0x4024d0 (no symbol found)
0x4025e0 (no symbol found)
0x402760 (no symbol found)
0x4028c0 (no symbol found)
0x402aa0 (no symbol found)
0x402c80 (no symbol found)
0x407b30 (no symbol found)
0x408e40 (no symbol found)
0x408fb0 (no symbol found)
0x409240 (no symbol found)
Select none
Select all
Writes to NUMA nodes
Node
Node 0
Node 1
Node 2
At least 0 B
Select none
Select all
Length
Filter tasks by length
Min
Max
Apply
Graph Code
Selected event
No event selected
Active task
No associated task
Statistics Heatmap
4879258384 - 22432062703
17.55 Gcycles selected
254913 tasks considered
4.30 Mcycles / task (avg)
254759 task creation events
Task length distribution
46.57%
0%
123.50 K
125.46 M
Communication
Nodes CPUs
detach
234.22 G local
89.06%
28.77 G remote
local
R W S P
Ignore direction
Exclude reflexive transfers
events only
Rel. min:
Rel. max:
Cyc Par
Seeking 2.25% 1.44
Texec: 96.56% 61.80
Tcreate: 0.17% 0.11
Resdep: 0.16% 0.10
Tdec: 0.54% 0.35
Bcast: 0.01% 0.01
Init: 0.29% 0.19
Est. cost: 0.00% 0.00
Reorder: 0.00% 0.00
Select from graph
CPU 0: state 1 (taskexec) from 12219469158 to 12222459295, duration: 2.99 Mcycles, active task: 0x40b9c0 (null)

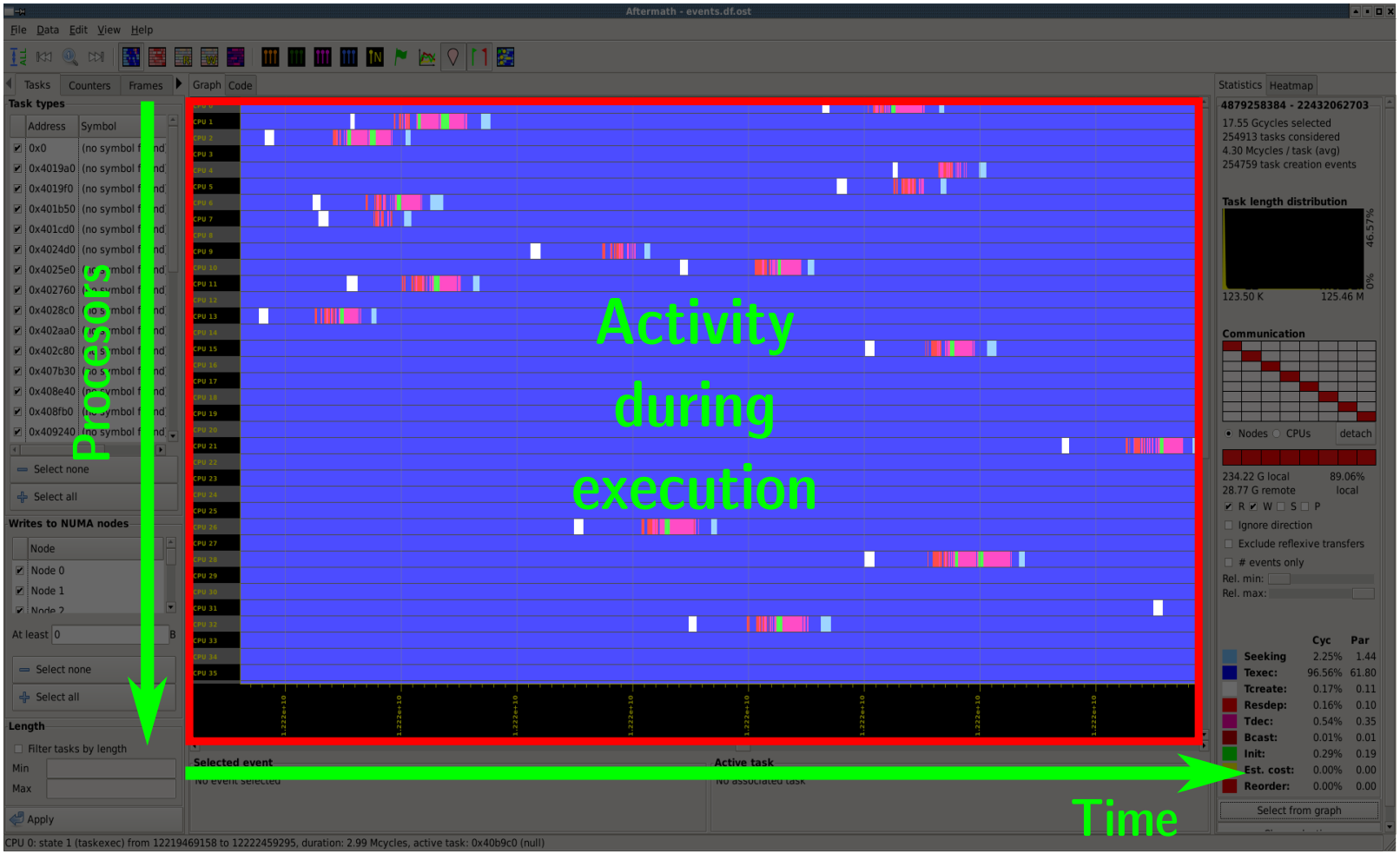
Aftermath


Menu bar: derived metrics
Filters
Timeline
Statistics
Detailed text view
Aftermath - events.df.ost
Task types
Address
Symbol
Writes to
Node
Node 1
At least
Select none
Select all
Length
Filter tasks by length
Min
Max
Apply
Statistics
Heatmap
4879258384 - 22432062703
17.55 Gcycles selected
254913 tasks considered
4.30 Mcycles / task (avg)
254759 task creation events
Task length distribution
Seeking 2.25% 1.44
Texec: 96.56% 61.80
Tcreate: 0.17% 0.11
Resdep: 0.16% 0.10
Tdec: 0.54% 0.35
Bcast: 0.01% 0.01
Init: 0.29% 0.19
Est. cost: 0.00% 0.00
Reorder: 0.00% 0.00
Select from graph
Selected event
No event selected
Active task
CPU 0: state 1 (taskexec) from 12219469158 to 12222459295, duration: 2.99 Mcycles, active task: 0x40b9c0 (null)

Aftermath


Aftermath - events.df.ost
File Data Edit View Help
Tasks Counters Frames
Graph Code
Statistics Heatmap
Task types
Address Symbol
0x0 (no symbol found)
0x4019a0 (no symbol found)
0x4019f0 (no symbol found)
0x401b50 (no symbol found)
0x401cd0 (no symbol found)
0x4024d0 (no symbol found)
0x4025e0 (no symbol found)
0x402760 (no symbol found)
0x4028c0 (no symbol found)
0x402aa0 (no symbol found)
0x402c80 (no symbol found)
0x407b30 (no symbol found)
0x408e40 (no symbol found)
0x408fb0 (no symbol found)
0x409240 (no symbol found)
Select none
Select all
Writes to NUMA nodes
Node
Node 0
Node 1
Node 2
At least 0 B
Select none
Select all
Length
Filter tasks by length
Min
Max
Apply
4879258384 - 22432062703
17.55 Gcycles selected
254913 tasks considered
4.30 Mcycles / task (avg)
254759 task creation events
Task length distribution
46.57%
0%
123.50 K
125.46 M
Communication
Nodes CPUs
detach
234.22 G local
89.06%
28.77 G remote
local
R W S P
Ignore direction
Exclude reflexive transfers
events only
Rel. min:
Rel. max:
Cyc Par
Seeking 2.25% 1.44
Texec: 96.56% 61.80
Tcreate: 0.17% 0.11
Resdep: 0.16% 0.10
Tdec: 0.54% 0.35
Bcast: 0.01% 0.01
Init: 0.29% 0.19
Est. cost: 0.00% 0.00
Reorder: 0.00% 0.00
Select from graph
Selected event
No event selected
Active task
No associated task
CPU 0: state 1 (taskexec) from 12219469158 to 12222459295, duration: 2.99 Mcycles, active task: 0x40b9c0 (null)

Aftermath


Aftermath - events.df.ost
File Data Edit View Help
Tasks Counters Frames
Graph Code
Statistics Heatmap
Task types
Address Symbol
Writes to NUMA nodes
Node
Node 0
Node 1
Length
Filter tasks by length
Min
Max
Apply
Select none
Select all
Processes
Activity during execution
Time
Selected event
Active task
4879258384 - 22432062703
17.55 Gcycles selected
254913 tasks considered
4.30 Mcycles / task (avg)
254759 task creation events
Task length distribution
123.50 K
125.46 M
Communication
Nodes CPUs
detach
234.22 G local
89.06%
28.77 G remote
local
Ignore direction
Exclude reflexive transfers
events only
Rel. min:
Rel. max:
Cyc Par
Seeking 2.25% 1.44
Texec: 96.56% 61.80
Tcreate: 0.17% 0.11
Resdep: 0.16% 0.10
Tdec: 0.54% 0.35
Bcast: 0.01% 0.01
Init: 0.29% 0.19
Est. cost: 0.00% 0.00
Reorder: 0.00% 0.00
Select from graph
CPU 0: state 1 (taskexec) from 12219469158 to 12222459295, duration: 2.99 Mcycles, active task: 0x40b9c0 (null)

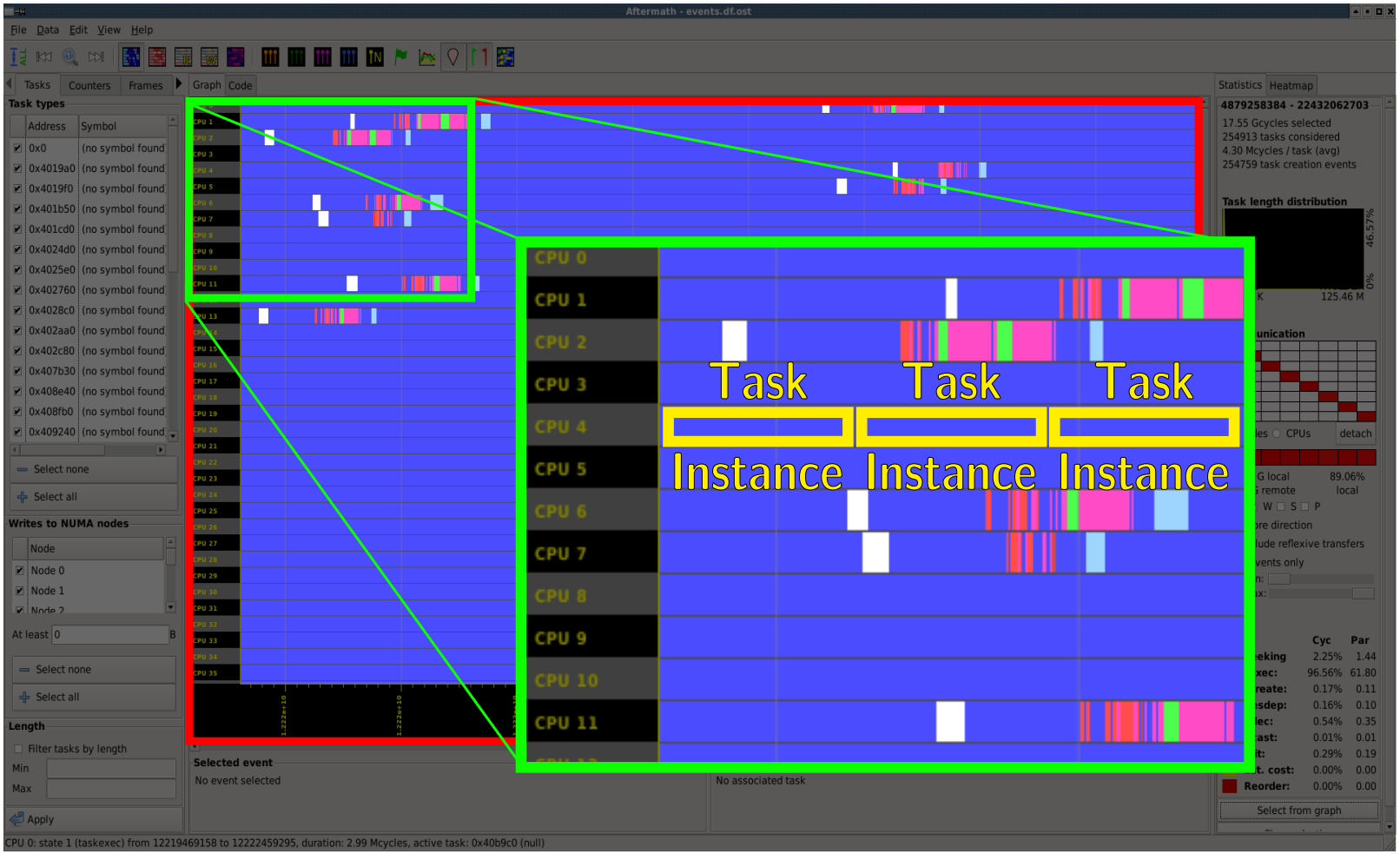
Aftermath


Aftermath - events.df.ost
File Data Edit View Help
Tasks Counters Frames
Graph Code
Task types
Address Symbol
0x0 (no symbol found
0x4019a0 (no symbol found
0x4019f0 (no symbol found
0x401b50 (no symbol found
0x401cd0 (no symbol found
0x4024d0 (no symbol found
0x4025e0 (no symbol found
0x402760 (no symbol found
0x4028c0 (no symbol found
0x402aa0 (no symbol found
0x402c80 (no symbol found
0x407b30 (no symbol found
0x408e40 (no symbol found
0x408fb0 (no symbol found
0x409240 (no symbol found
Select none
Select all
Writes to NUMA nodes
Node
Node 0
Node 1
At least 0 B
Select none
Select all
Length
Filter tasks by length
Min
Max
Apply
CPU 0
CPU 1
CPU 2
CPU 3
CPU 4
CPU 5
CPU 6
CPU 7
CPU 8
CPU 9
CPU 10
CPU 11
Selected event
No event selected
No associated task
Statistics Heatmap
4879258384 - 22432062703
17.55 Gcycles selected
254913 tasks considered
4.30 Mcycles / task (avg)
254759 task creation events
Task length distribution
125.46 M
CPUs
detach
89.06%
local
Cyc Par
2.25% 1.44
96.56% 61.80
0.17% 0.11
0.16% 0.10
0.54% 0.35
0.01% 0.01
0.29% 0.19
0.00% 0.00
Reorder: 0.00% 0.00
Select from graph
CPU 0: state 1 (taskexec) from 12219469158 to 12222459295, duration: 2.99 Mcycles, active task: 0x40b9c0 (null)

Aftermath

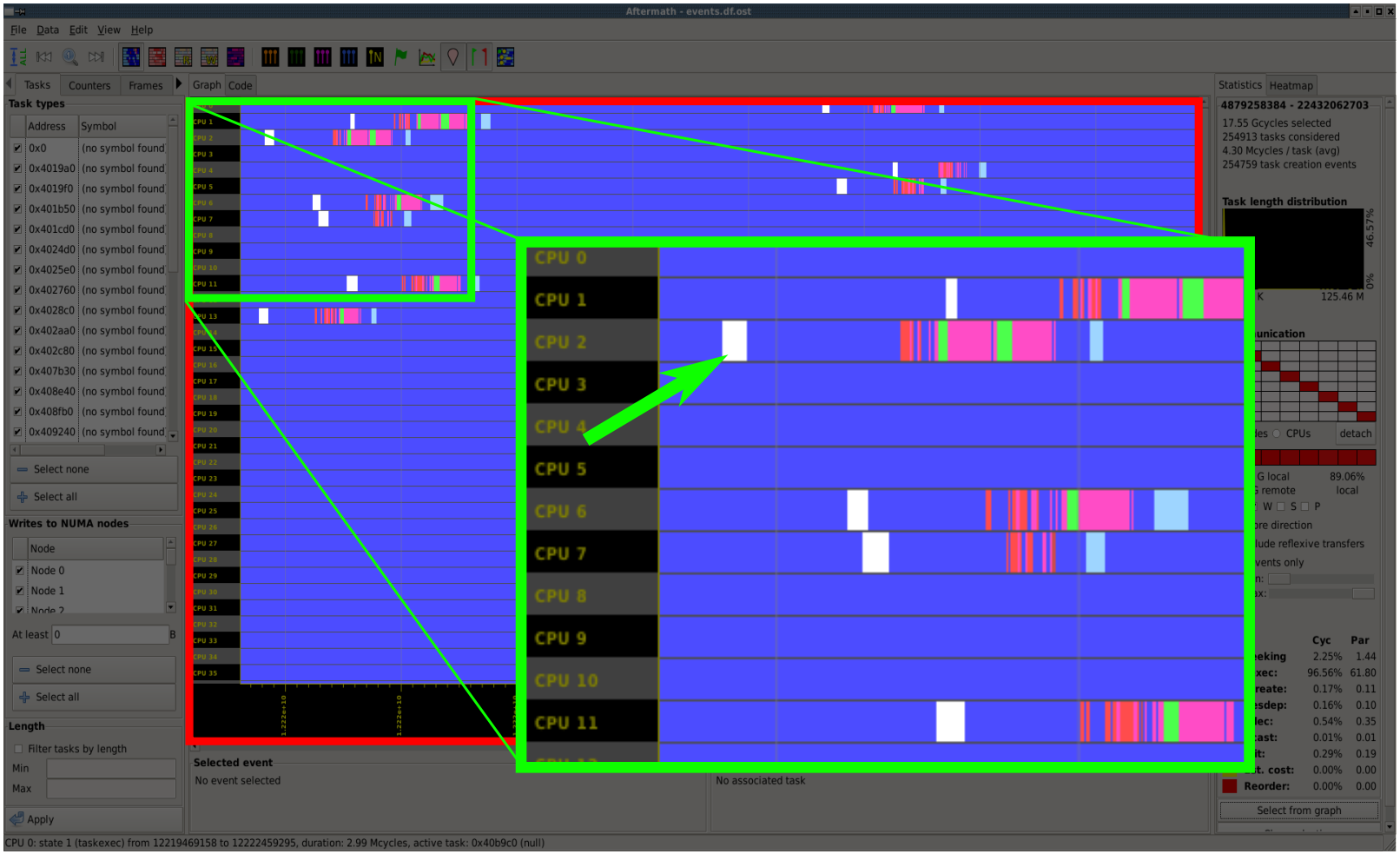
Task execution (dark blue)

Aftermath



Task execution (dark blue)

Aftermath

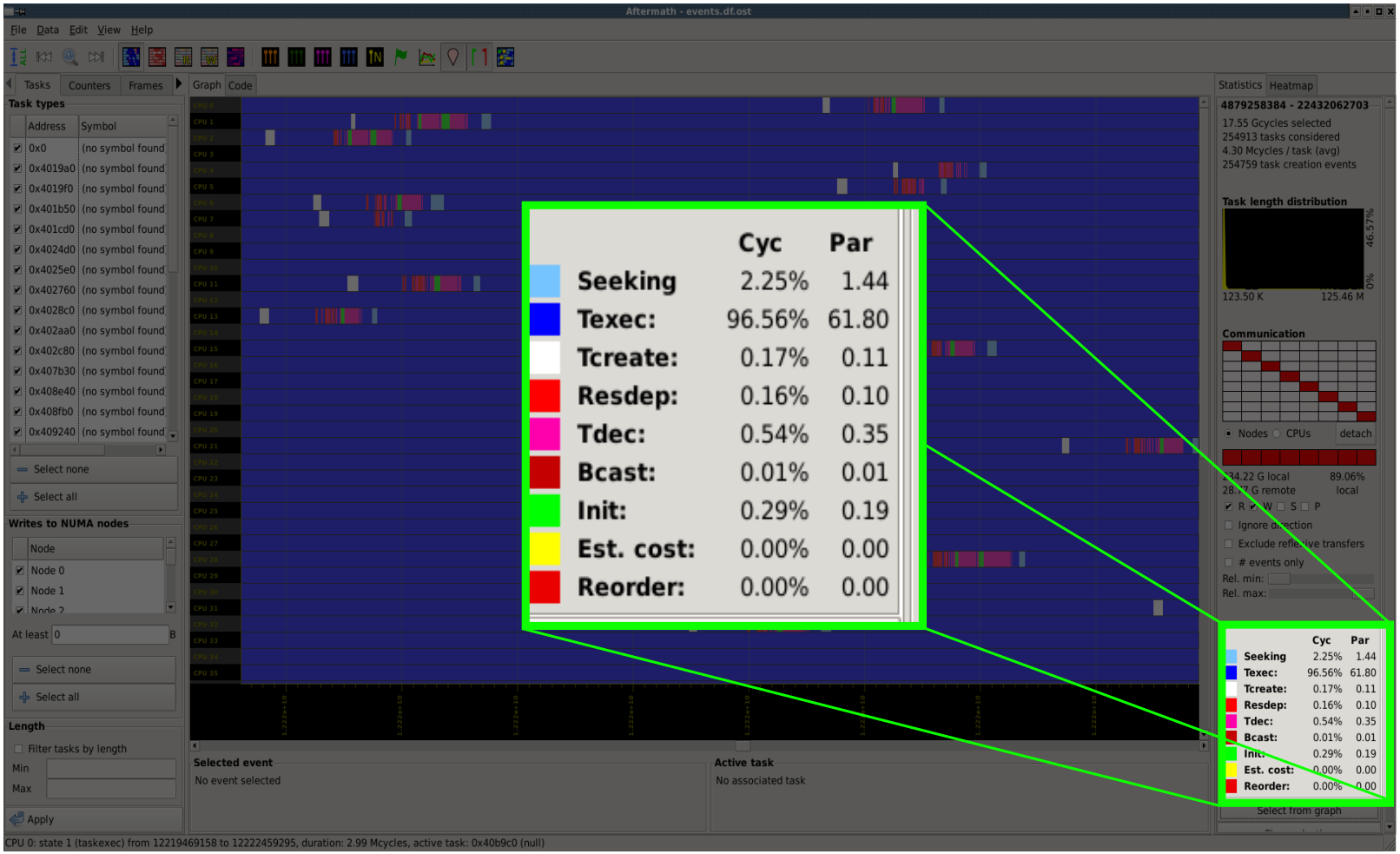


Task creation (white)

Aftermath

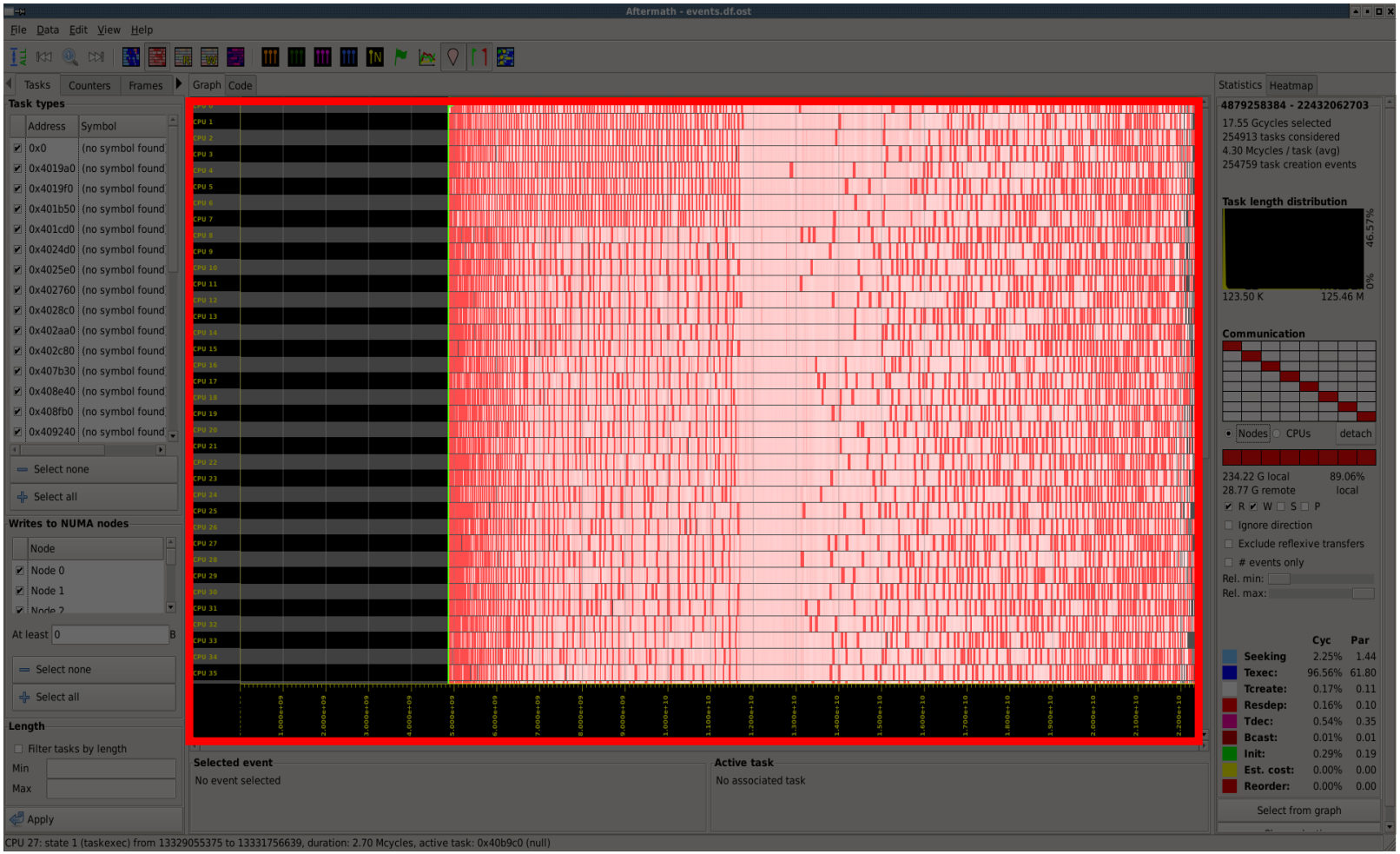
Searching for work (light blue)

Aftermath



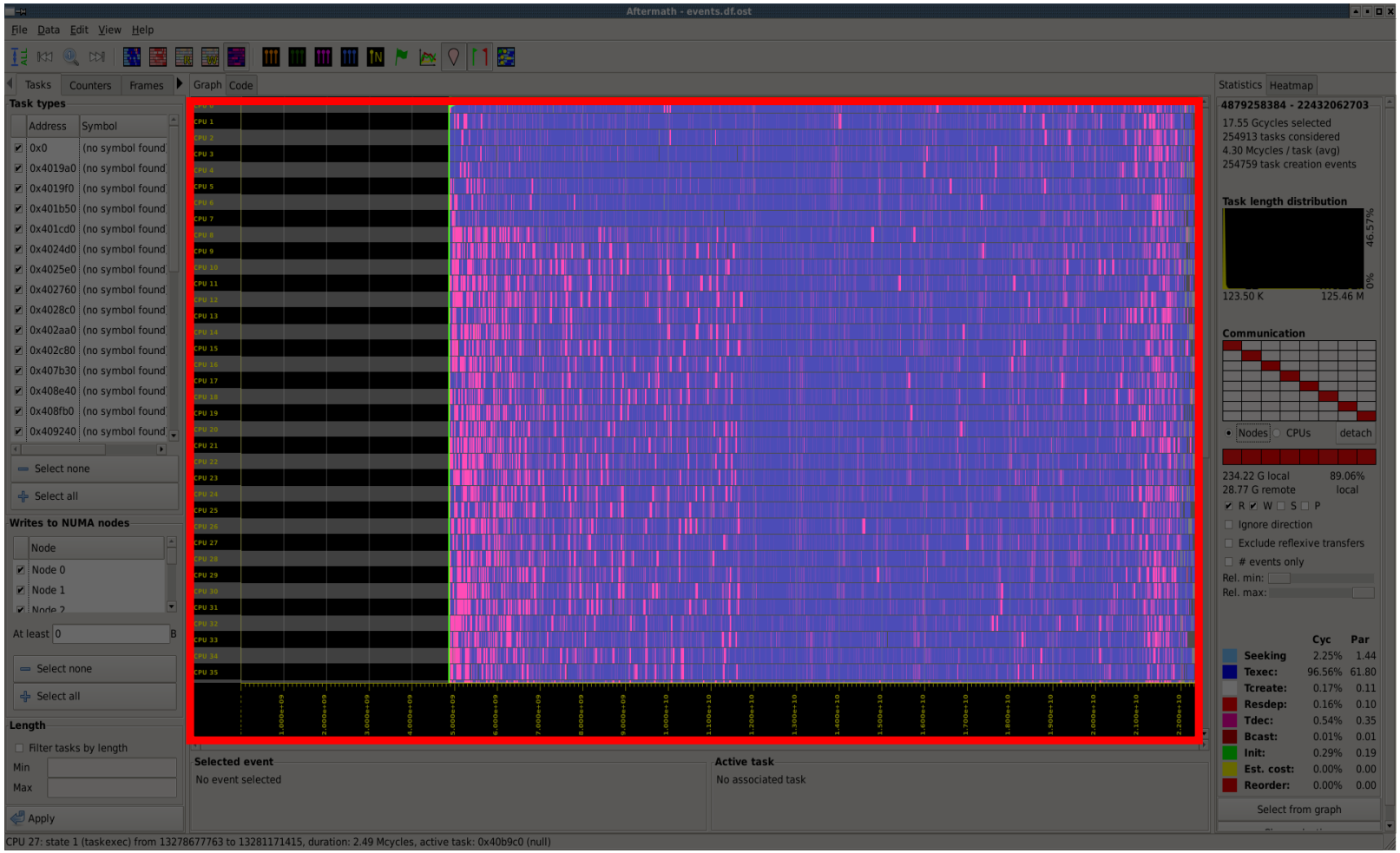
Basic statistics for run-time states

Aftermath

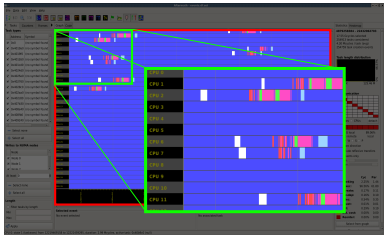
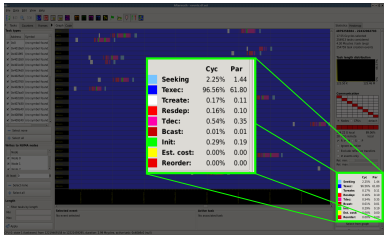


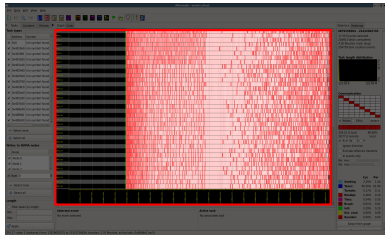
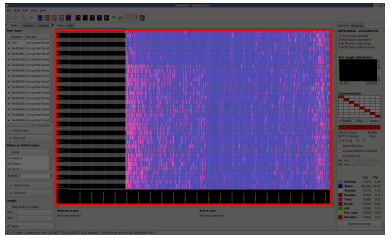
Heatmap indicating task duration (white: fast, red: slow)

Aftermath



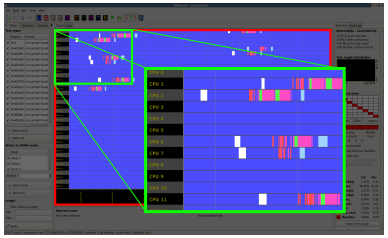
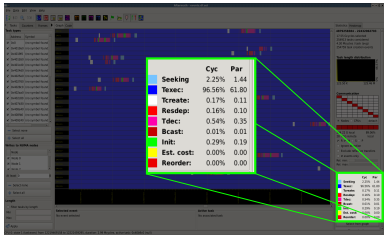
NUMA heatmap indicating locality of memory accesses (blue: local, pink: remote)

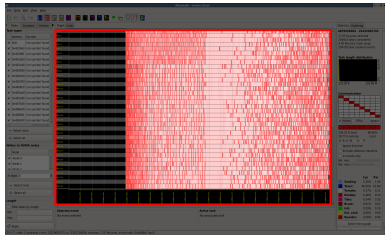
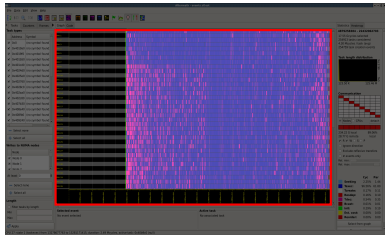



Navigating through a trace

- Huge amounts of high-dimensional data
- Lots of features $\rightarrow$ lots of possibilities for analysis
- Where to look? What to look for?
- Expertise & lots of time required

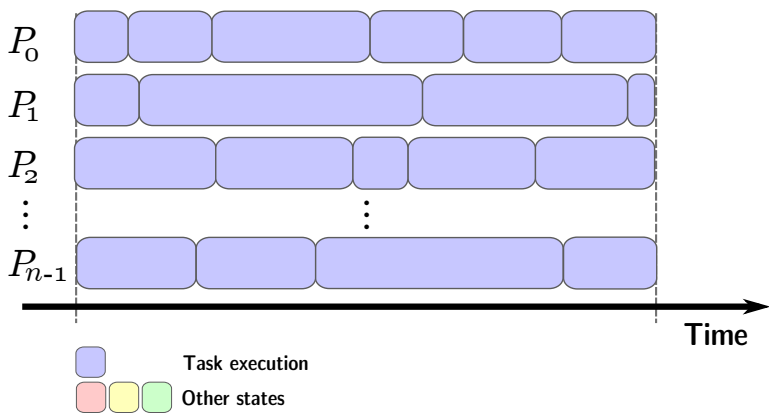
Navigating through a trace

- Huge amounts of high-dimensional data
- Lots of features $\rightarrow$ lots of possibilities for analysis
- Where to look? What to look for?
- Expertise & lots of time required

Guide the user through performance analysis

- Refine analysis in several steps
  - Start by analyzing parallelism
  - Analyze what happens inside tasks
- Automate repetitive tasks

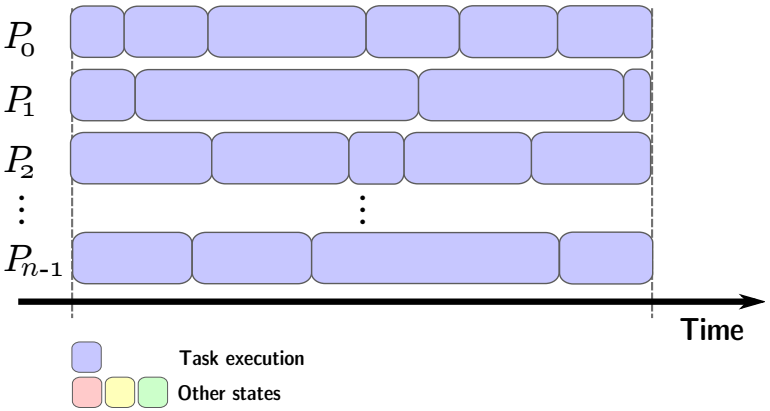
Detecting insufficient parallelism



Ideal situation

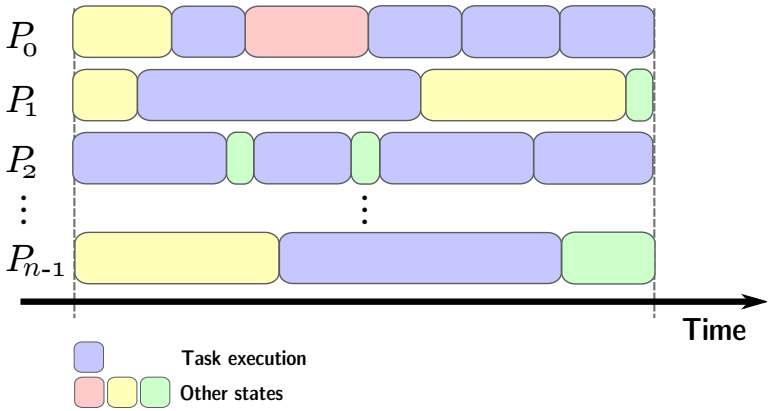
All CPUs are in task execution state without any interruption

Detecting insufficient parallelism



Ideal situation

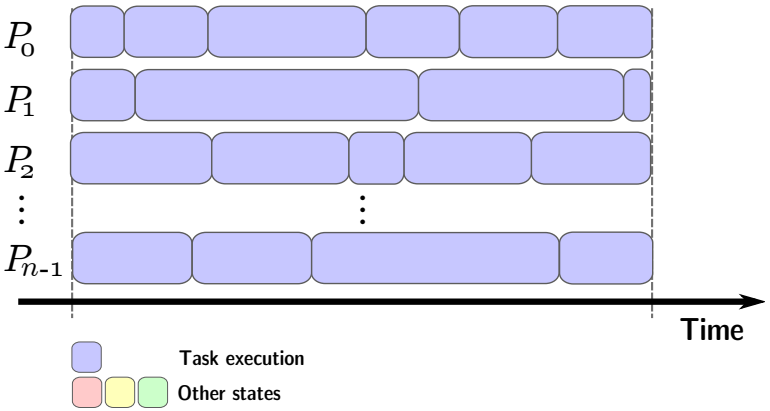
All CPUs are in task execution state without any interruption



Realistic scenario

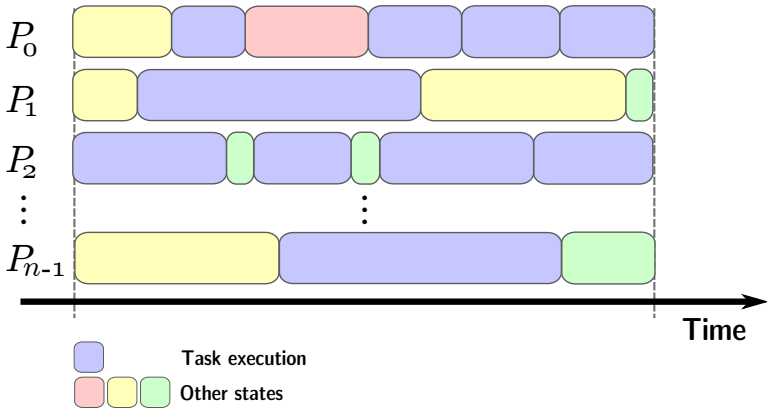
Task creation, memory allocation, idle time, over-synchronization, ...

Detecting insufficient parallelism



Ideal situation

All CPUs are in task execution state without any interruption

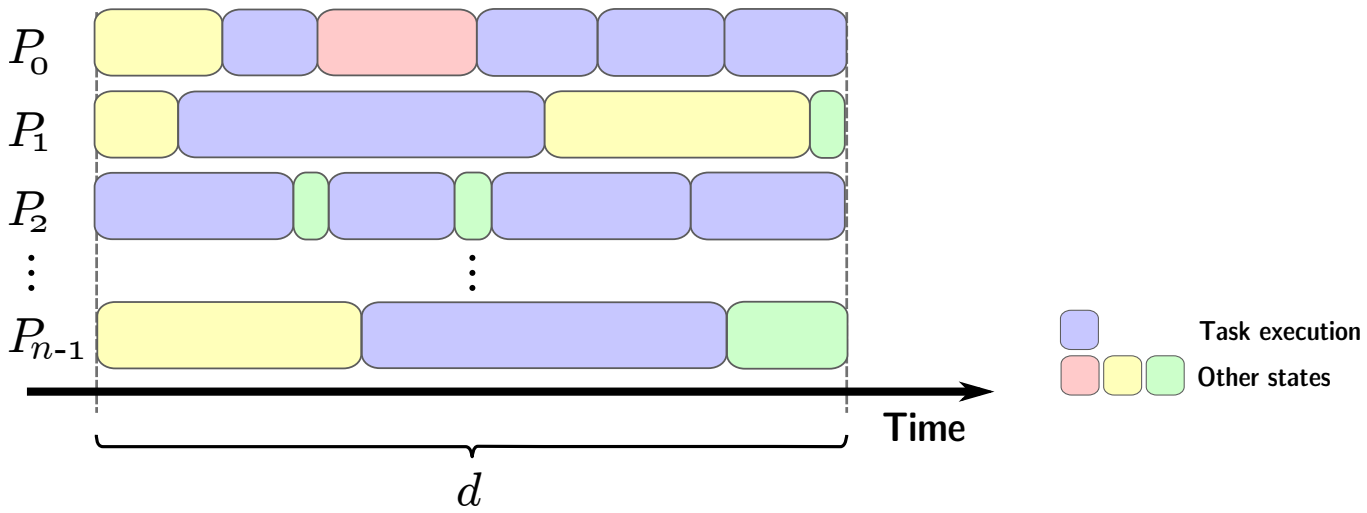


Realistic scenario

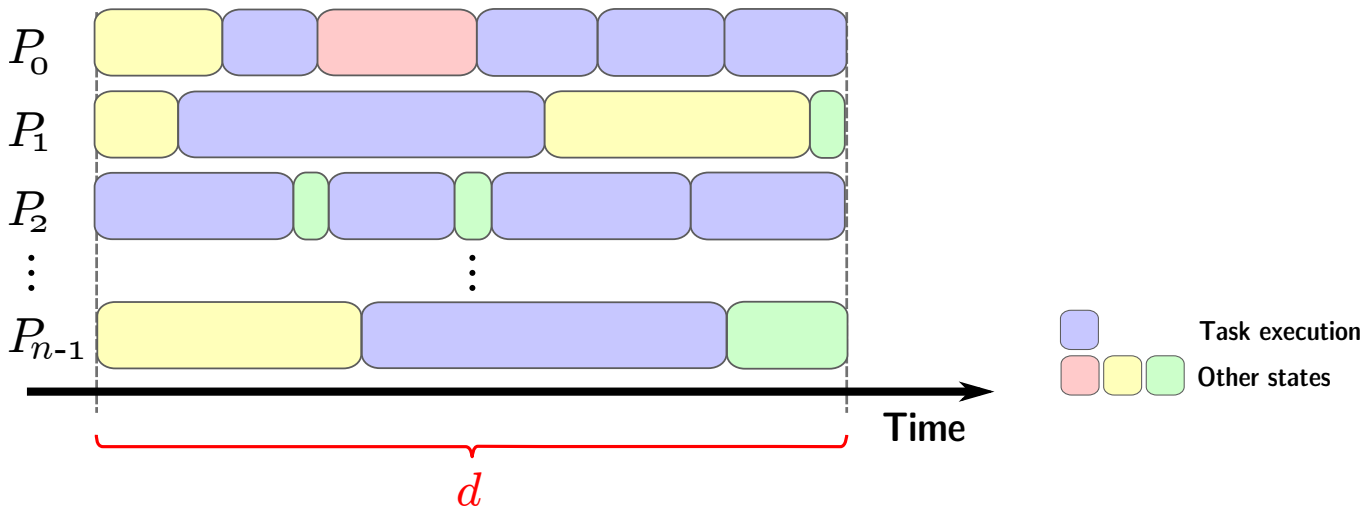
Task creation, memory allocation, idle time, over-synchronization, ...

Detect insufficient parallelism / high overhead automatically

Threshold-based analysis of parallelism

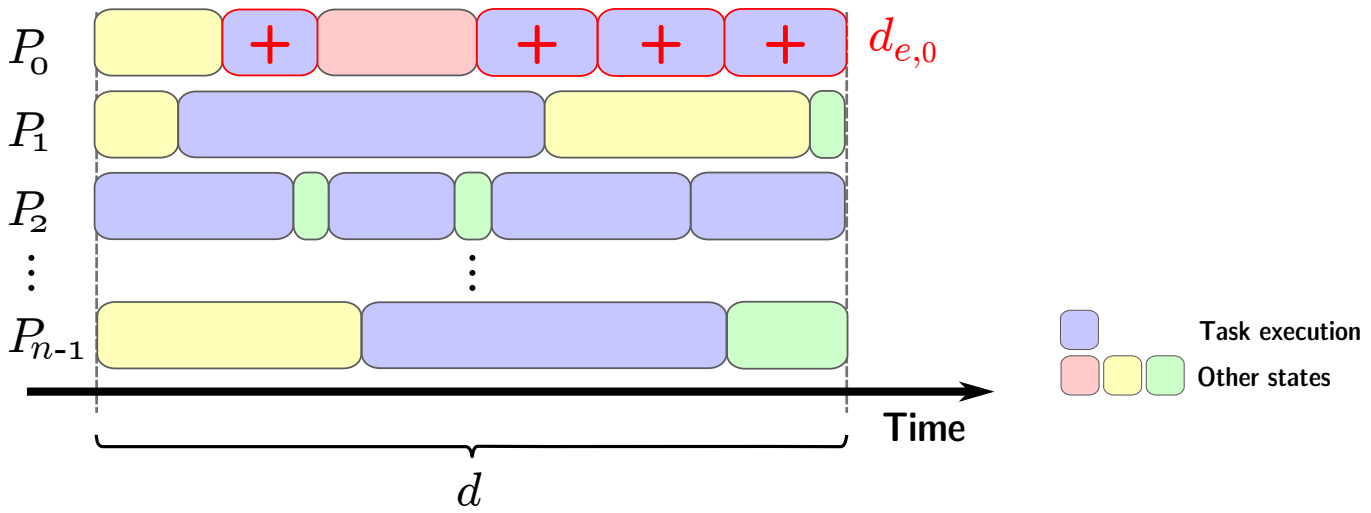


Threshold-based analysis of parallelism



d: Duration of the interval

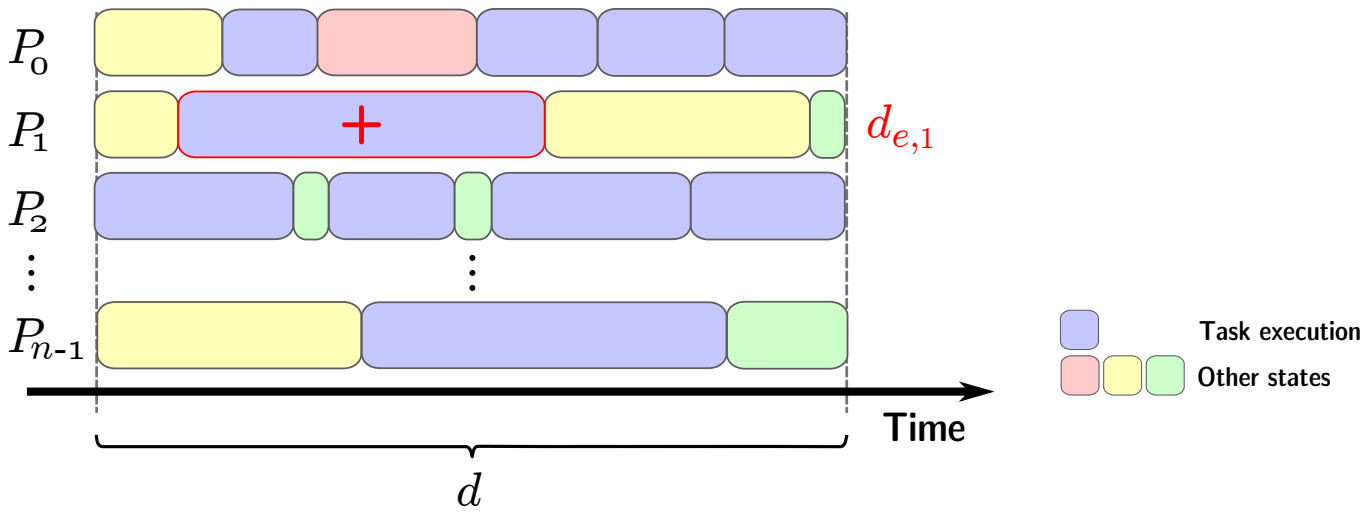
Threshold-based analysis of parallelism



d: Duration of the interval

$d_{e,i}$: Time that processor i spends in task execution state

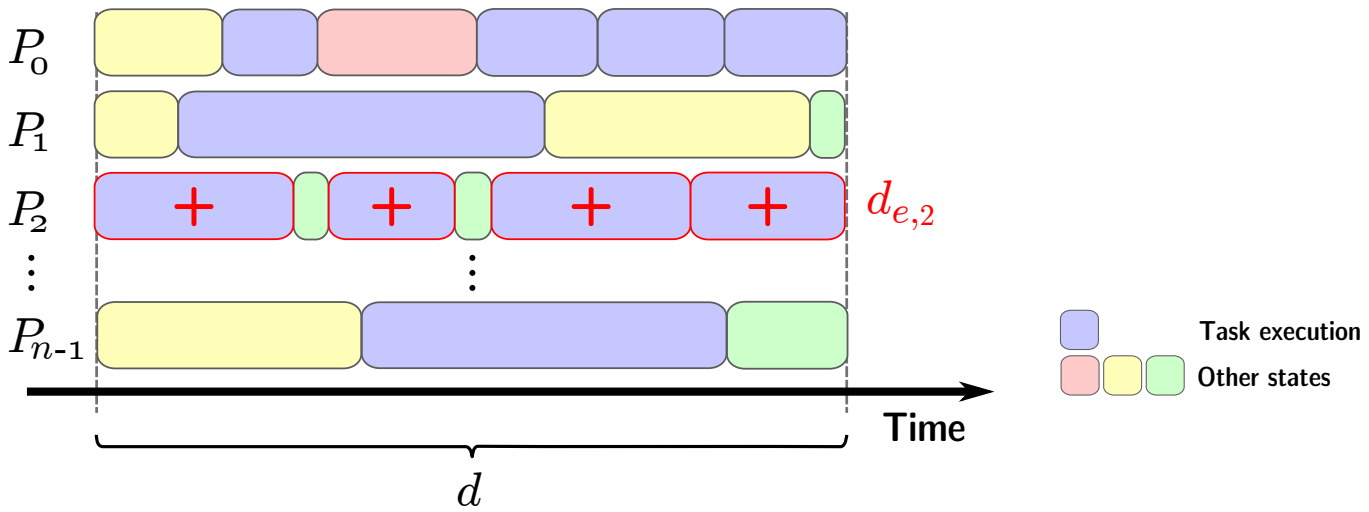
Threshold-based analysis of parallelism



d: Duration of the interval

$d_{e,i}$: Time that processor i spends in task execution state

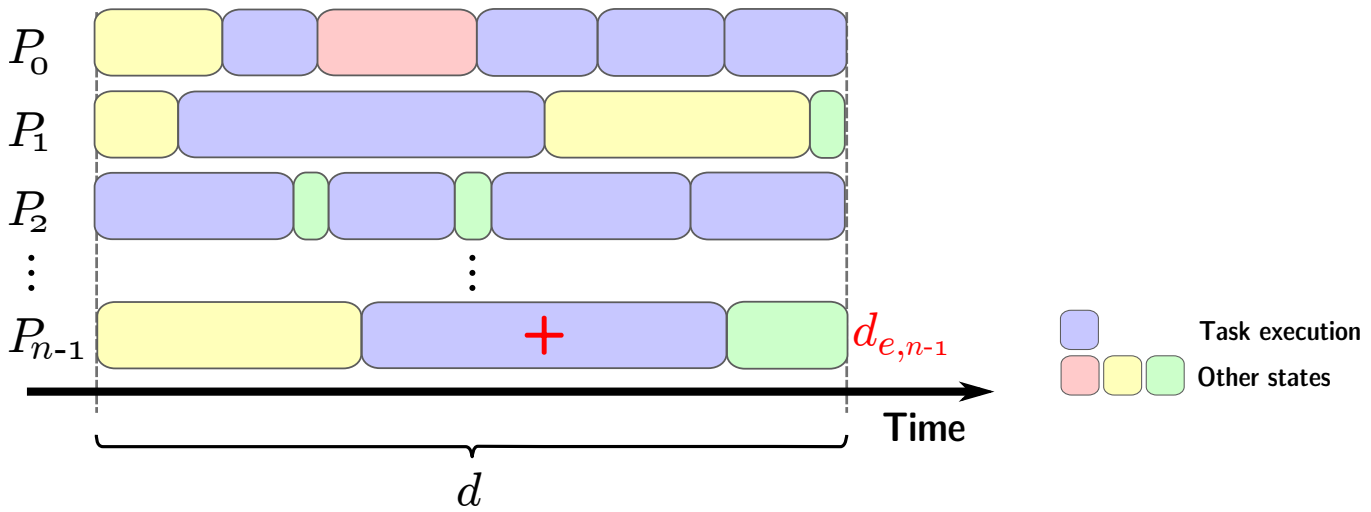
Threshold-based analysis of parallelism



d: Duration of the interval

$d_{e,i}$: Time that processor i spends in task execution state

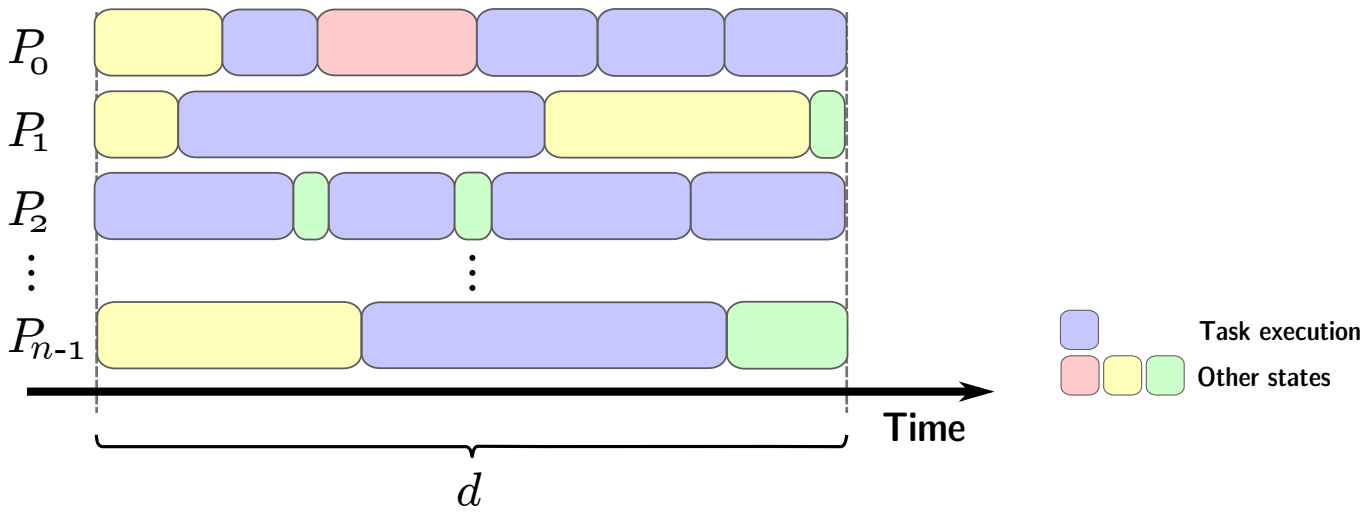
Threshold-based analysis of parallelism



d: Duration of the interval

$d_{e,i}$: Time that processor i spends in task execution state

Threshold-based analysis of parallelism

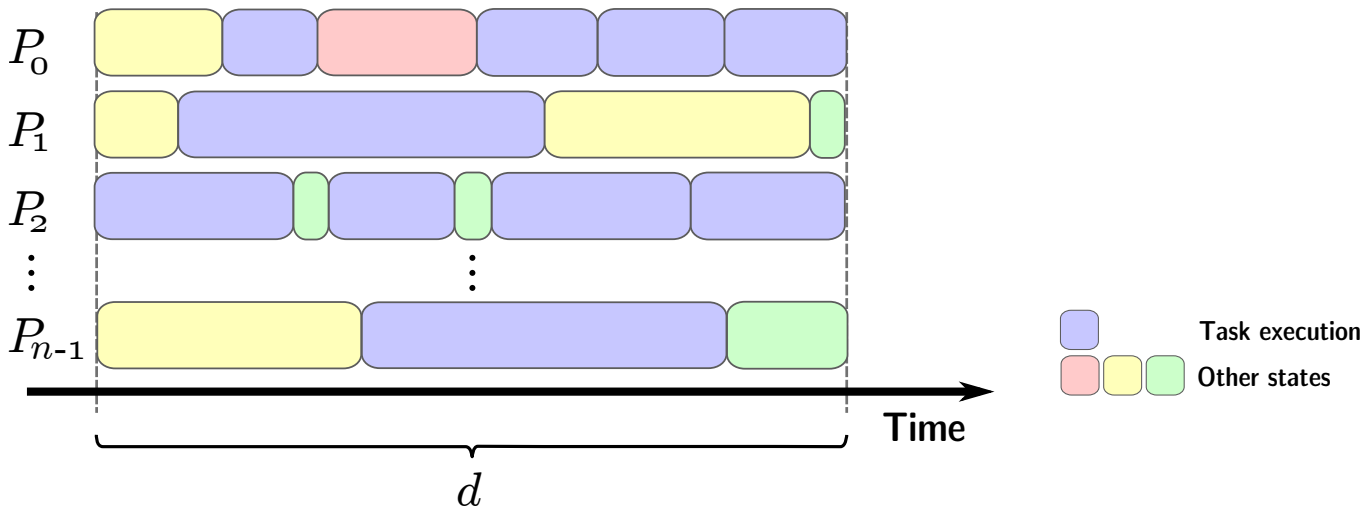


d: Duration of the interval

$d_{e,i}$: Time that processor i spends in task execution state

t_e: Threshold for task execution, e.g. $t_e = 0.95$

Threshold-based analysis of parallelism



d: Duration of the interval

$d_{e,i}$: Time that processor i spends in task execution state

t_e: Threshold for task execution, e.g. $t_e = 0.95$

Consider that there is sufficient parallelism if inequation holds:

$$\sum_{i=1}^{n} d_{e,i} > t_e \cdot n \cdot d$$

Detecting the cause of insufficient parallelism

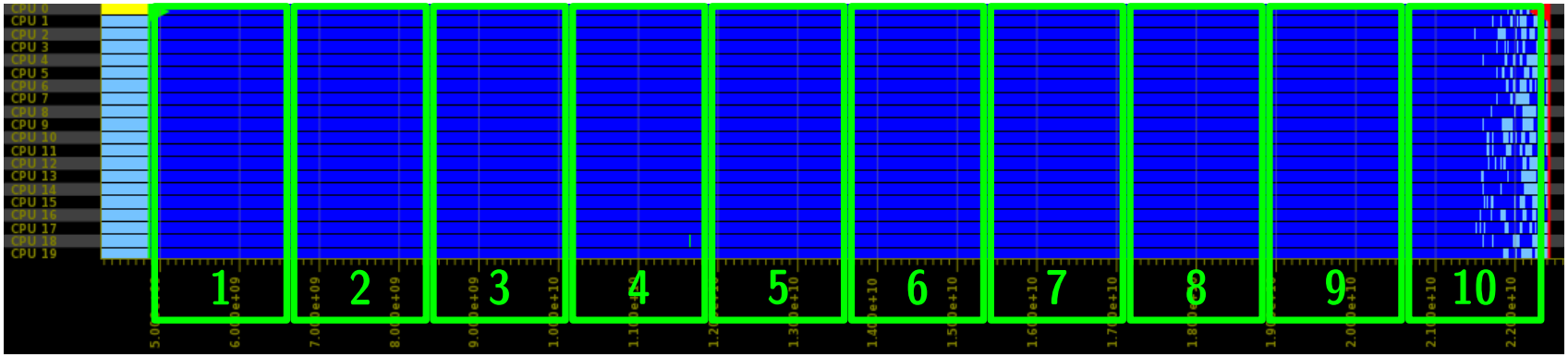
Multiple stages during analysis

- If inequation does not hold, find out why
- Possible causes: task creation overhead, memory allocation, not enough tasks available for execution, ...
- Use thresholds for associated states:
  t_c (task creation), t_i (idle time)

Detecting the cause of insufficient parallelism

Multiple stages during analysis

- If inequation does not hold, find out why
- Possible causes: task creation overhead, memory allocation, not enough tasks available for execution, ...
- Use thresholds for associated states:
  t_c (task creation), t_i (idle time)

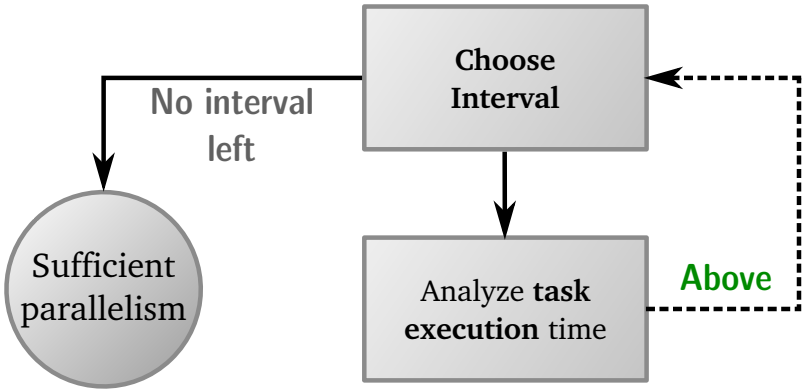


Interval selection

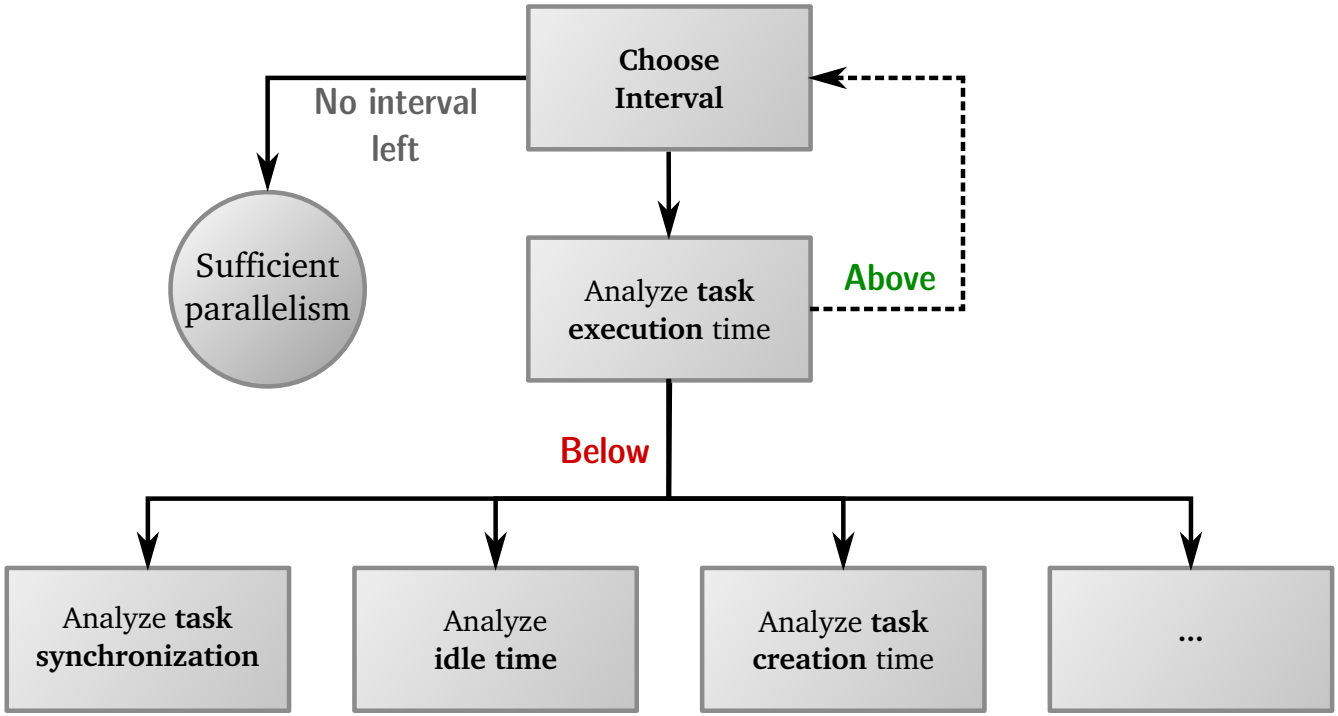
- Multiple intervals: initialization, termination, etc.
- Repeat analysis for different intervals

Per-interval analysis of parallelism & overhead

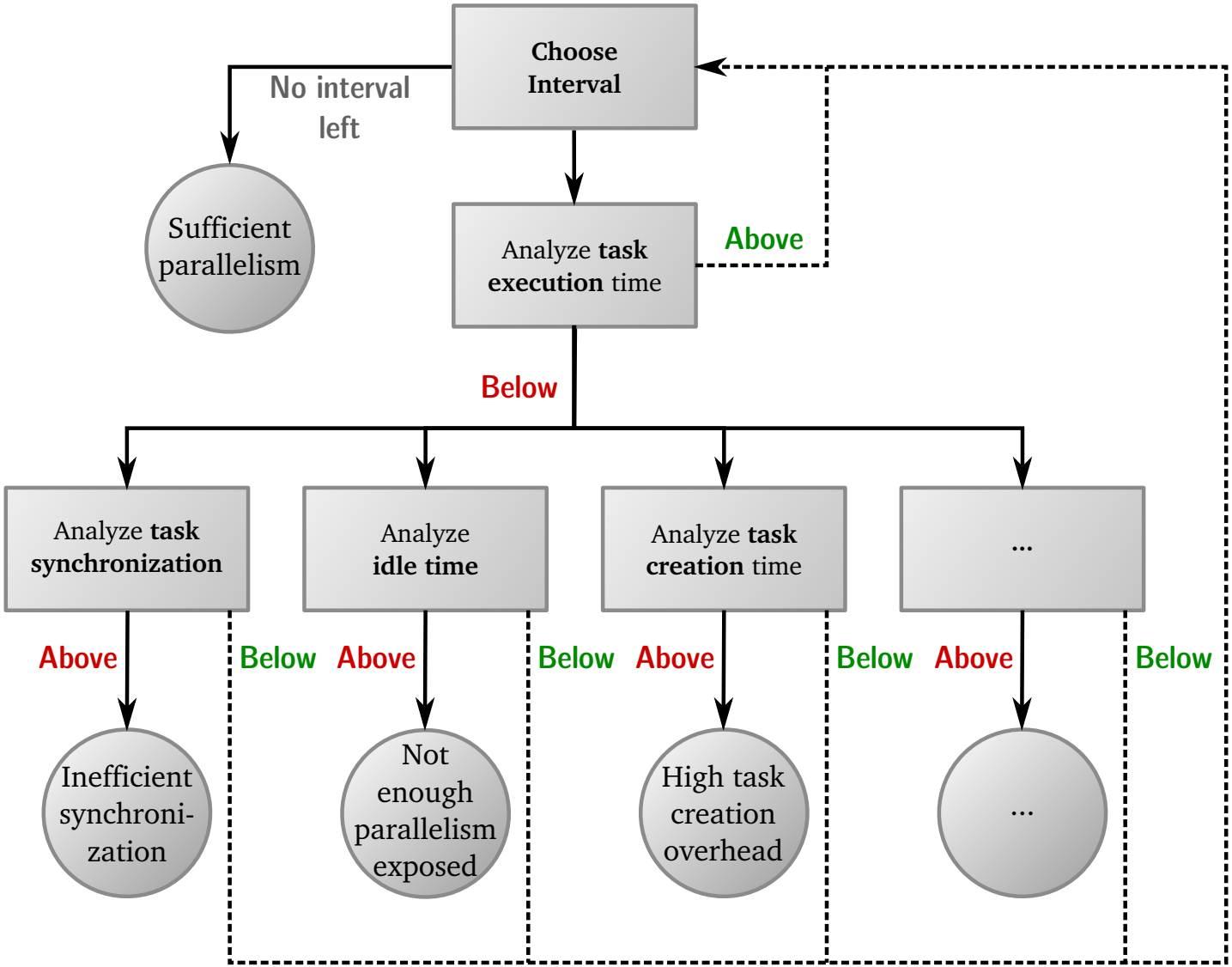
Per-interval analysis of parallelism & overhead



Per-interval analysis of parallelism & overhead



Per-interval analysis of parallelism & overhead



Detecting performance anomalies during task execution

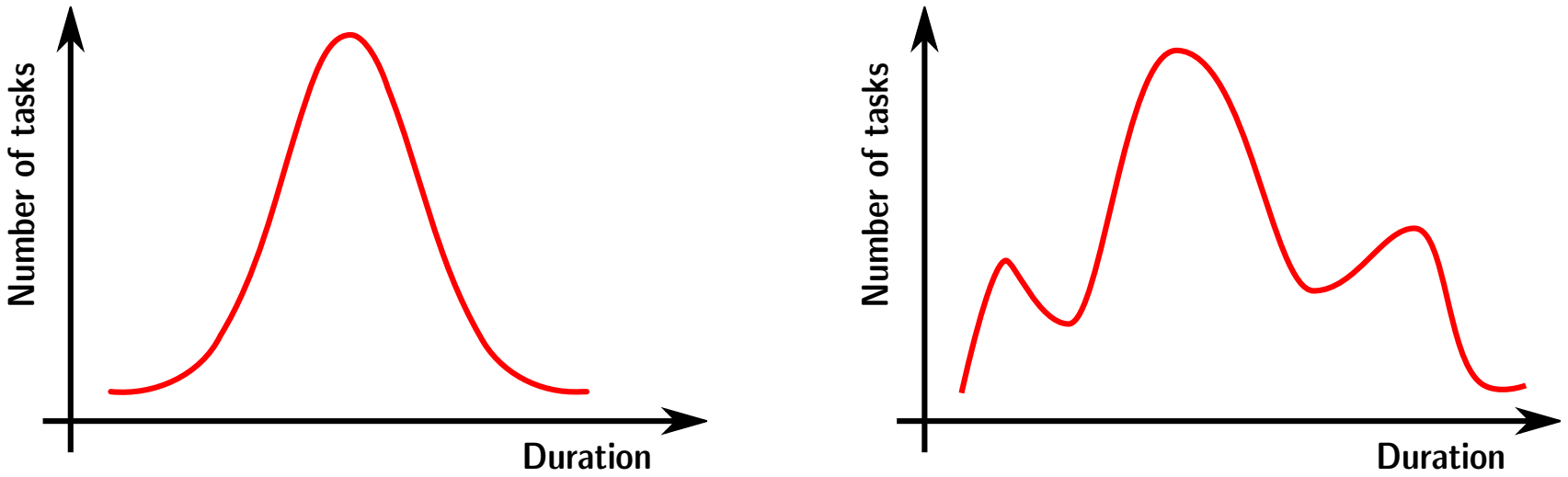
During task execution

- Performance anomaly possible even at 100% task execution (ineffective use of caches, remote memory accesses, branch misprediction)

Detecting performance anomalies during task execution

During task execution

- Performance anomaly possible even at 100% task execution (ineffective use of caches, remote memory accesses, branch misprediction)



Impact on the distribution of task duration

- Slowdown of all tasks
- Different groups / peaks

Using performance counters

Hardware performance counters

- Implemented in hardware, no slowdown of the application
- Low tracing overhead if sampled at beginning / end of a task
- Dozens of hardware events can be monitored

Using performance counters

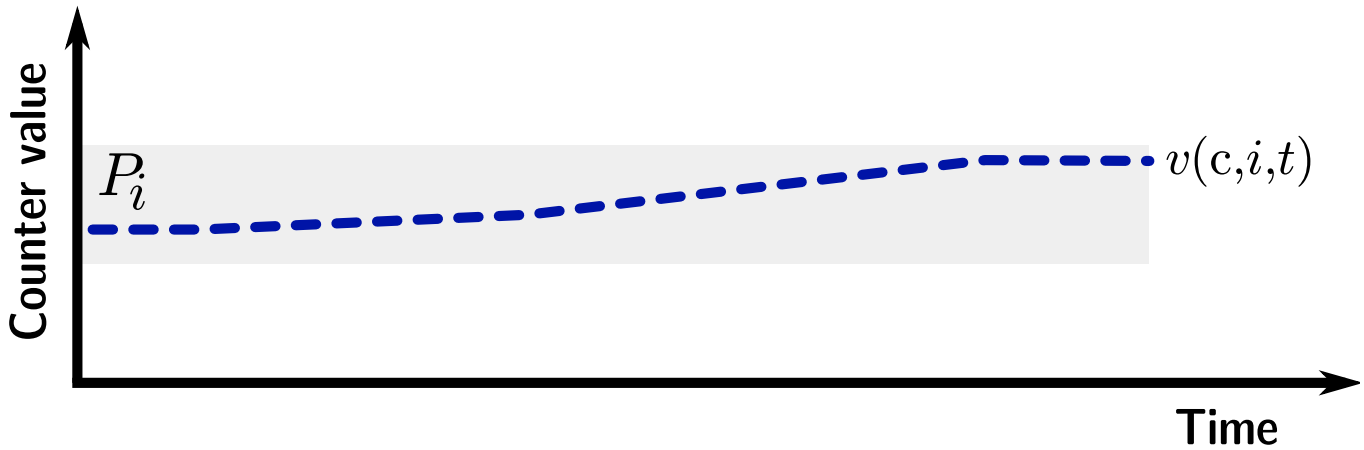
Hardware performance counters

- Implemented in hardware, no slowdown of the application
- Low tracing overhead if sampled at beginning / end of a task
- Dozens of hardware events can be monitored

Automatic analysis of performance counters

- Which hardware events are relevant?
- Manual testing tedious & time consuming

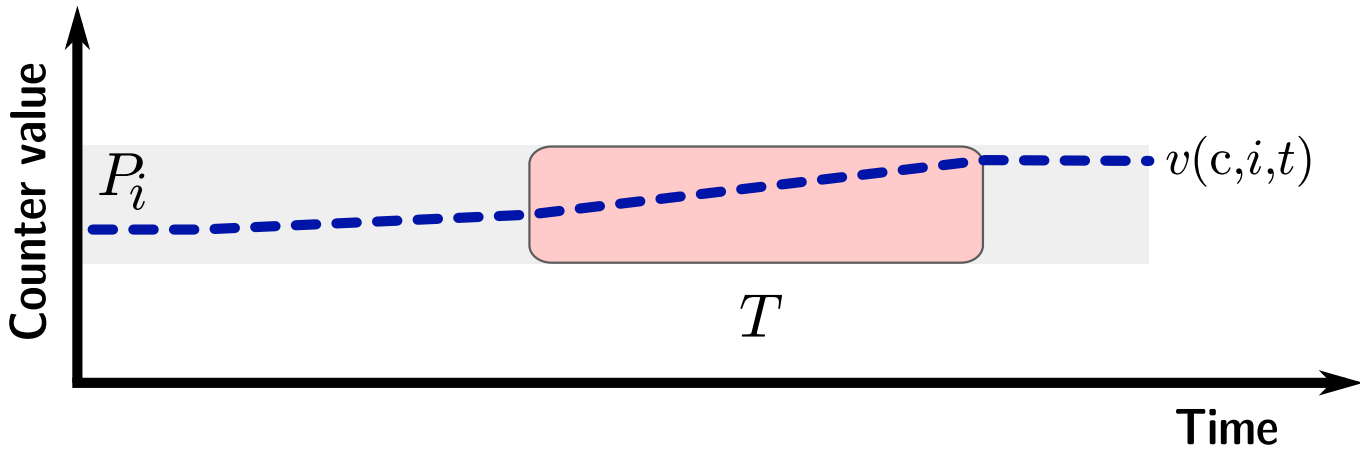
Analyzing performance counters



Per-CPU performance counter

- Absolute value $v(c, i, t)$, monotonically increasing
  - c: Counter (e.g. cache misses)
  - i: Processor identifier
  - t: Timestamp

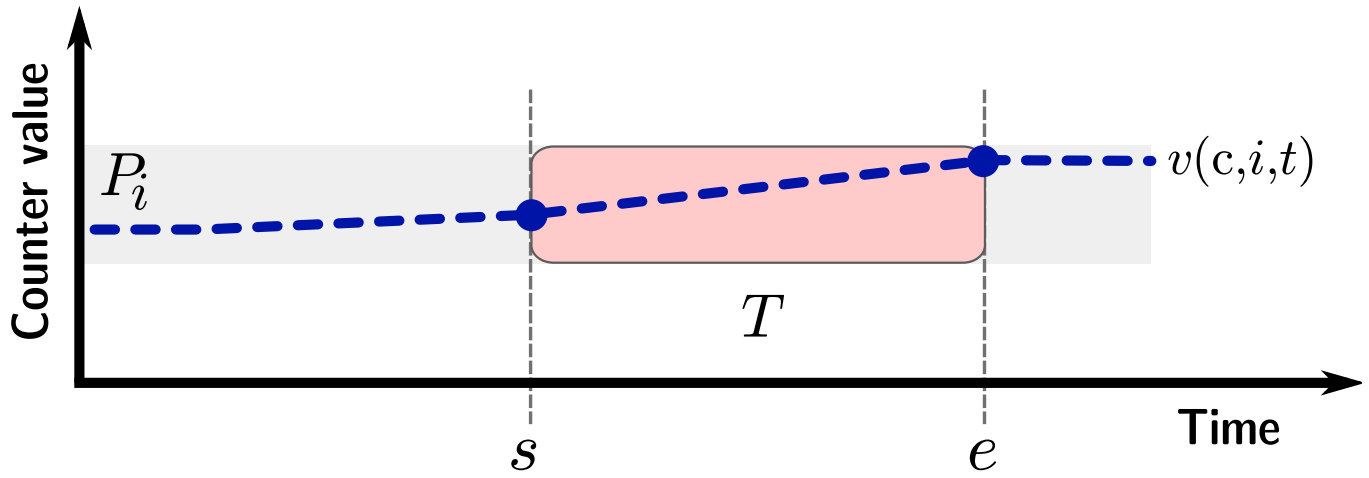
Analyzing performance counters



Per-CPU performance counter

- Absolute value $v(c, i, t)$, monotonically increasing
  - c: Counter (e.g. cache misses)
  - i: Processor identifier
  - t: Timestamp

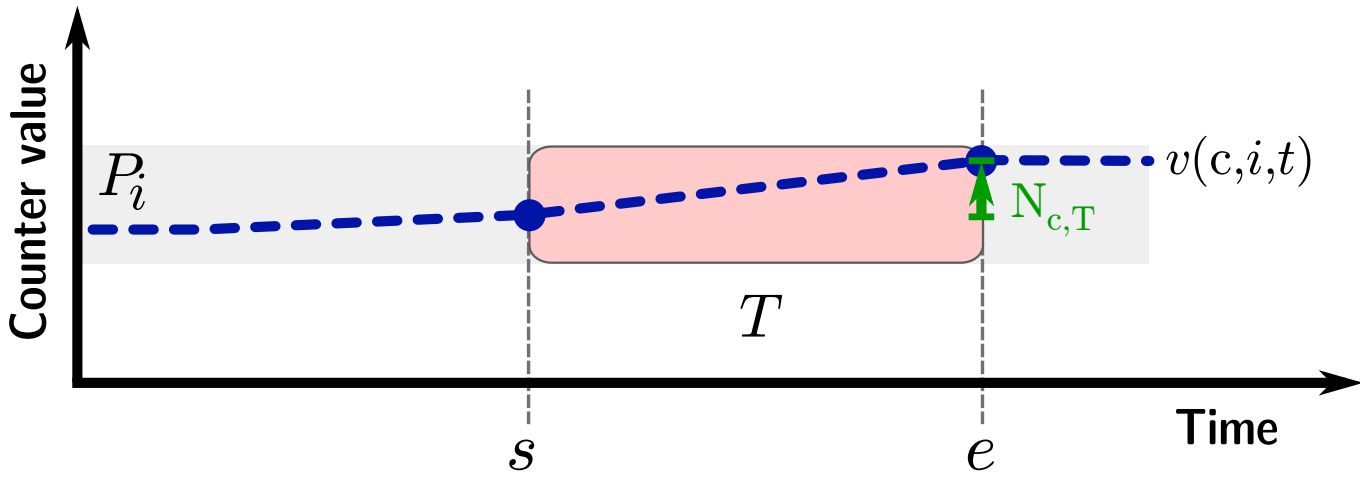
Analyzing performance counters



Per-CPU performance counter

- Absolute value $v(c, i, t)$, monotonically increasing
  - c: Counter (e.g. cache misses)
  - i: Processor identifier
  - t: Timestamp
- Sampled at the beginning and end of a task

Analyzing performance counters



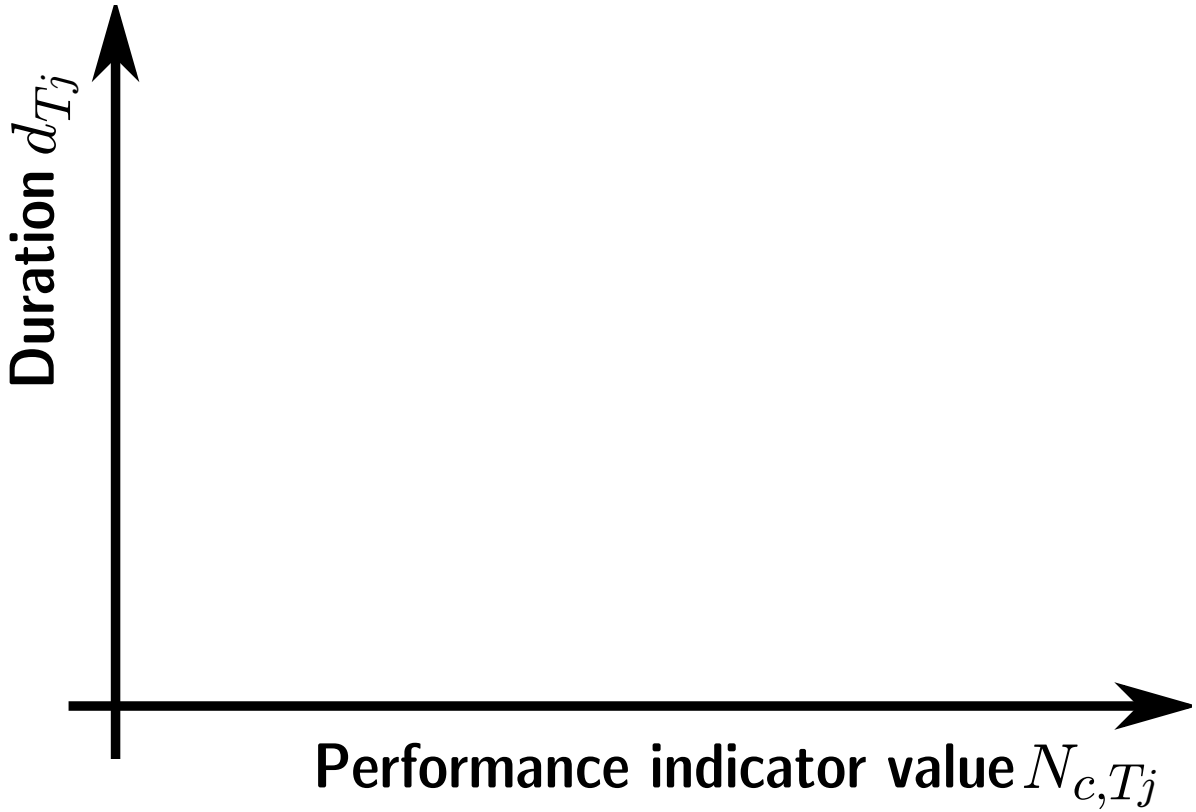
Per-CPU performance counter

- Absolute value $v(c, i, t)$, monotonically increasing
  - c: Counter (e.g. cache misses)
  - i: Processor identifier
  - t: Timestamp
- Sampled at the beginning and end of a task

Break down counter evolution to task instances

- Increase of c by task T: $N_{c,T} = v(c, i, e) - v(c, i, s)$

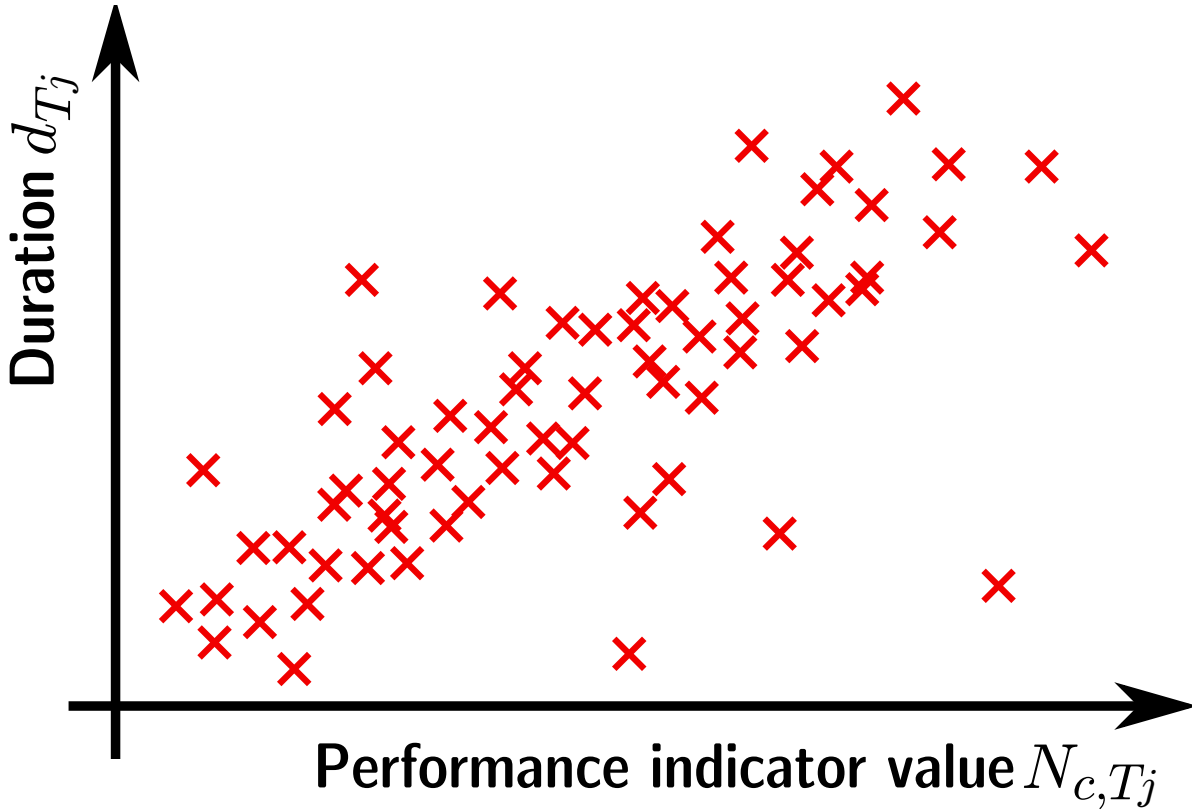
Linear regression



Perform linear regression

- Assume linear model: $d_{T_j} = \alpha \cdot N_{c,T_j} + \beta$ (α and β constant)
- Compare coefficient of determination with threshold

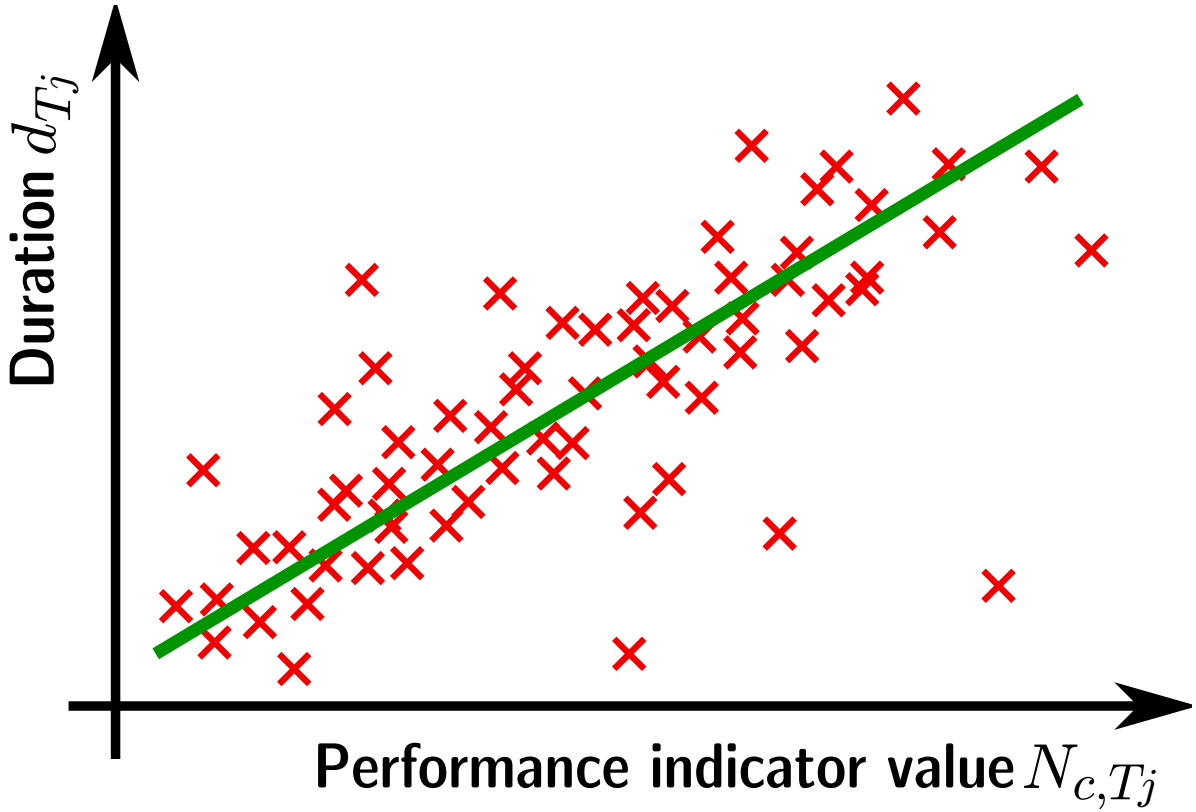
Linear regression



Perform linear regression

- Assume linear model: $d_{T_j} = \alpha \cdot N_{c,T_j} + \beta$ (α and β constant)
- Compare coefficient of determination with threshold

Linear regression



Perform linear regression

- Assume linear model: $d_{T_j} = \alpha \cdot N_{c,T_j} + \beta$ (α and β constant)
- Compare coefficient of determination with threshold

Shortcuts & refinement

Variation of task duration

- Determine coefficient of variation for task duration
- Only perform analysis if significant

Shortcuts & refinement

Variation of task duration

- Determine coefficient of variation for task duration
- Only perform analysis if significant

Task types

- Different task types in an application
  - Auxiliary tasks: initialization, termination
  - Work tasks: matrix multiplication, decomposition, etc.
- Performance anomaly not necessarily present in *all* types

Shortcuts & refinement

Variation of task duration

- Determine coefficient of variation for task duration
- Only perform analysis if significant

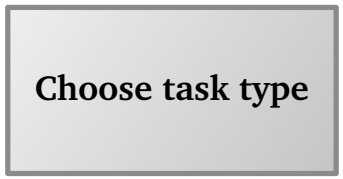
Task types

- Different task types in an application
  - Auxiliary tasks: initialization, termination
  - Work tasks: matrix multiplication, decomposition, etc.
- Performance anomaly not necessarily present in *all* types

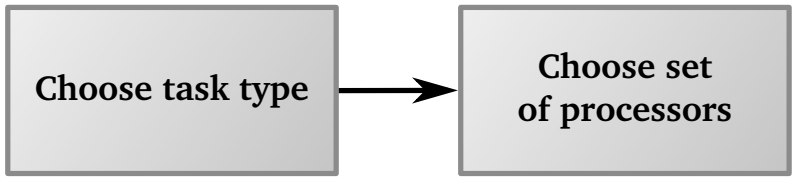
Topology of the machine

- Anomaly only present on subset of processors
- Example: Memory accesses local for one NUMA node, remote on another

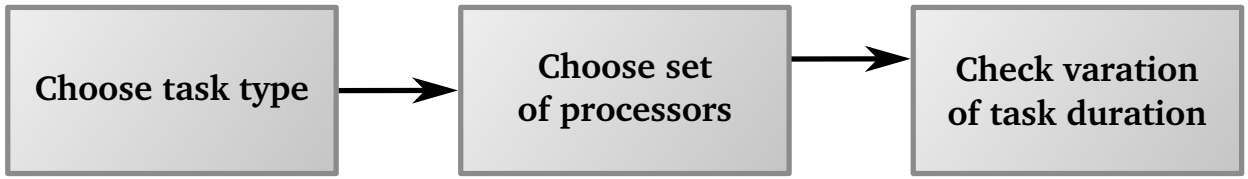
Per-interval analysis of parallelism & overhead



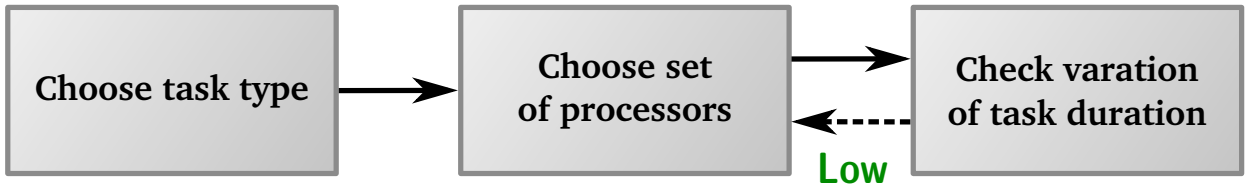
Per-interval analysis of parallelism & overhead



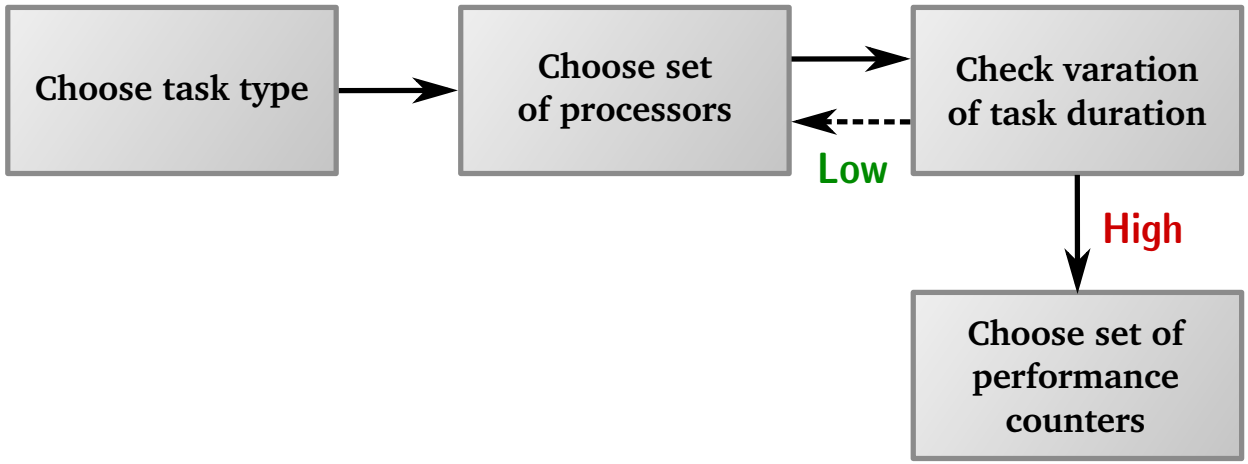
Per-interval analysis of parallelism & overhead



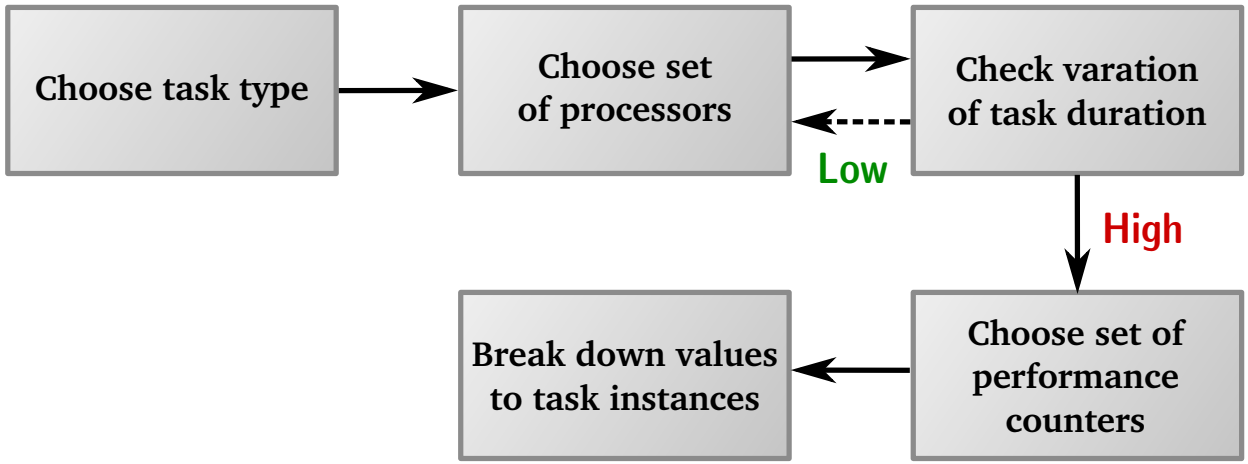
Per-interval analysis of parallelism & overhead



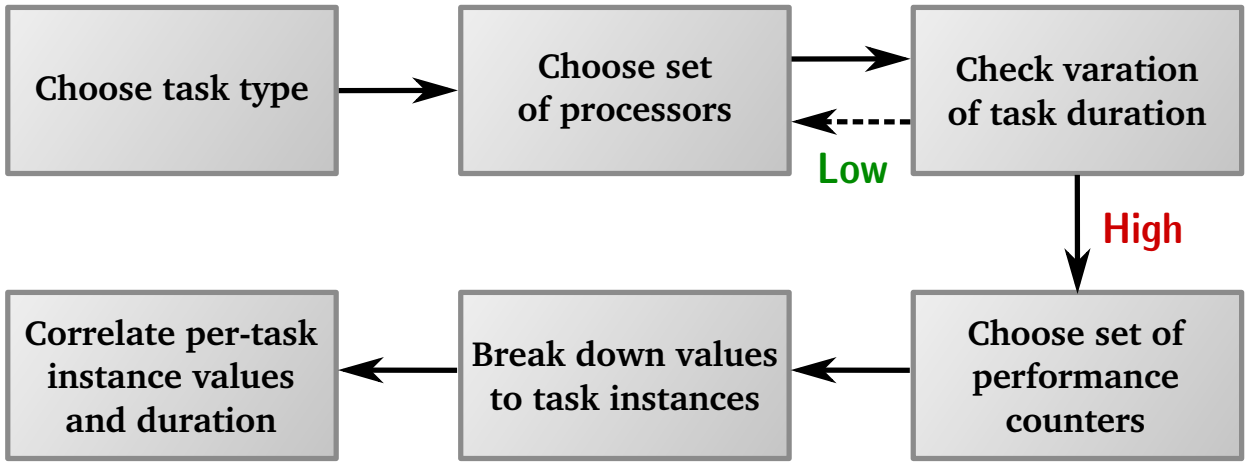
Per-interval analysis of parallelism & overhead



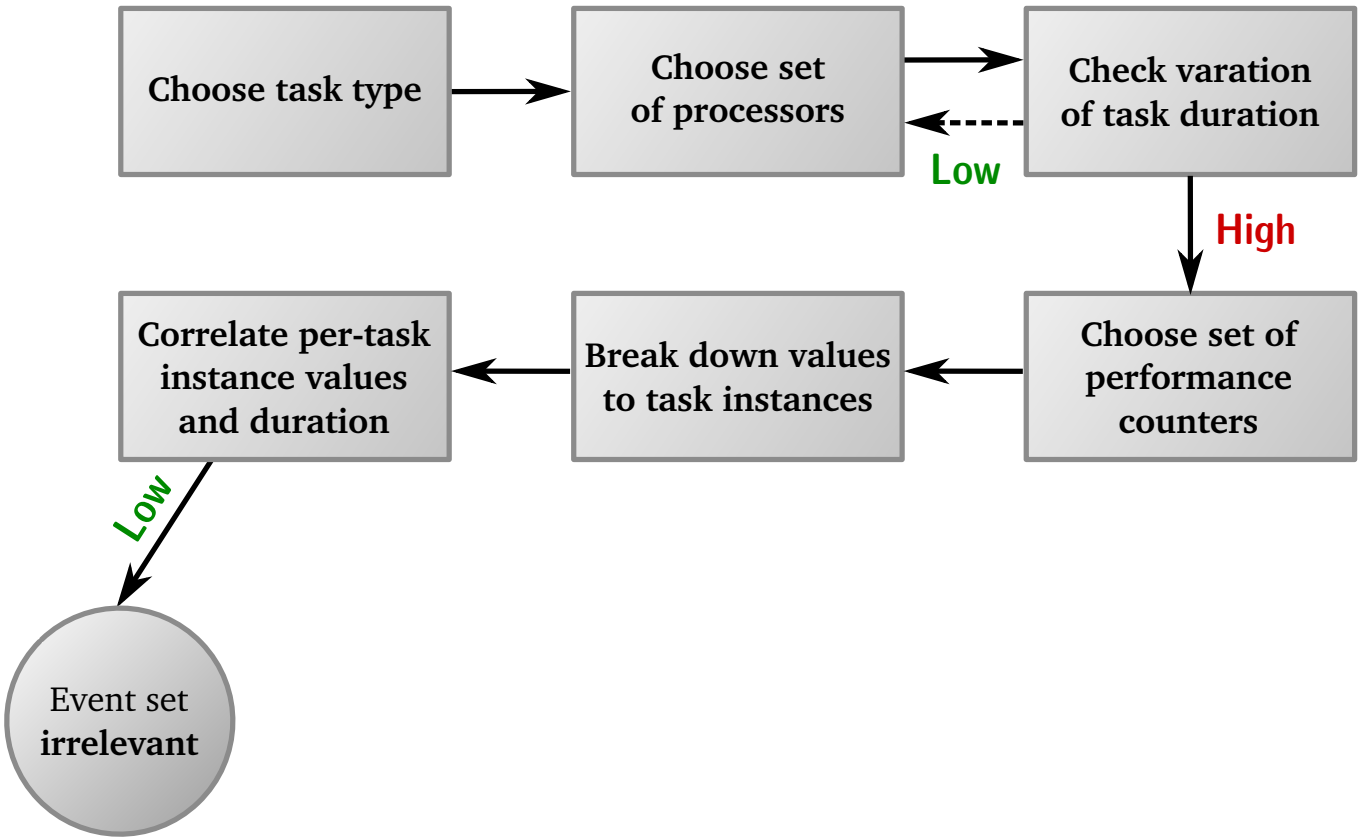
Per-interval analysis of parallelism & overhead



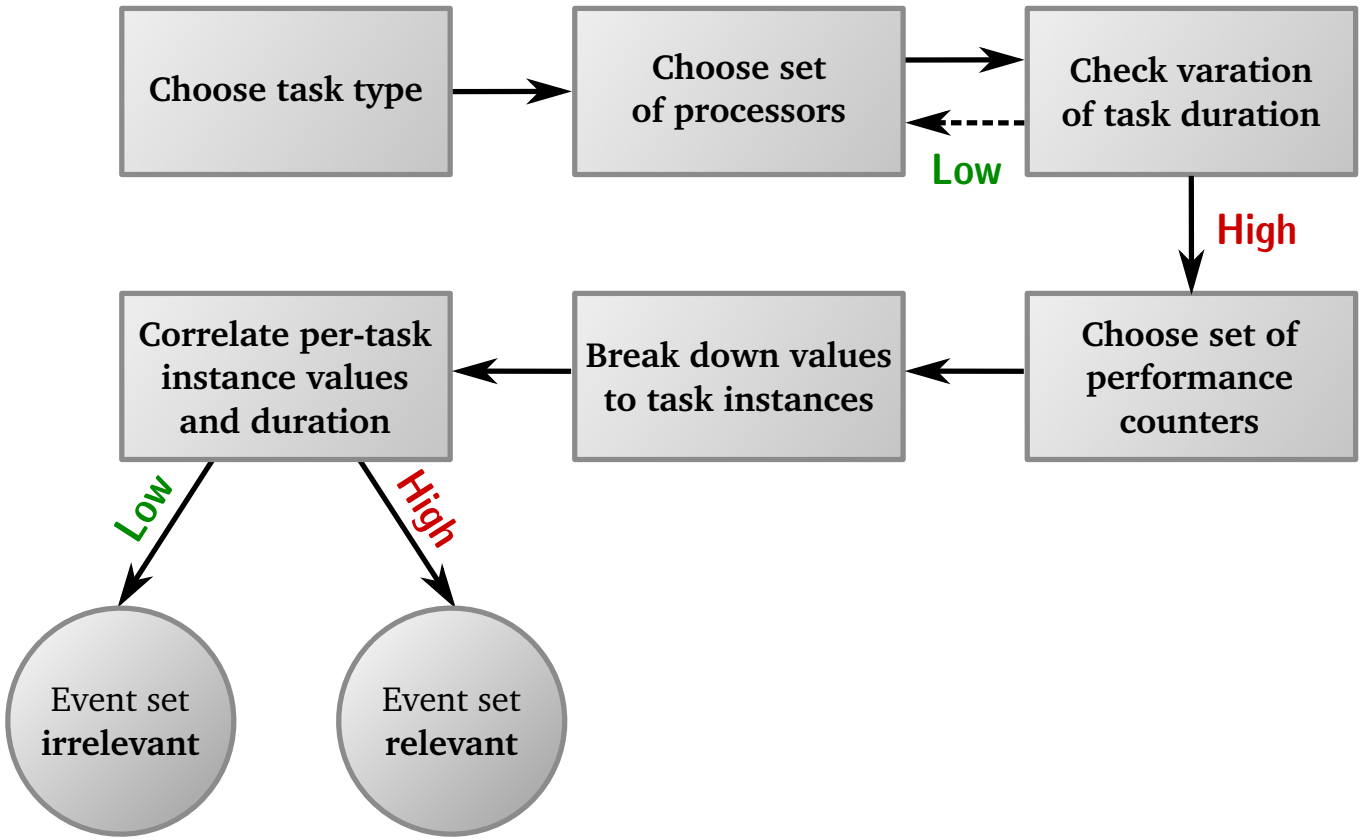
Per-interval analysis of parallelism & overhead



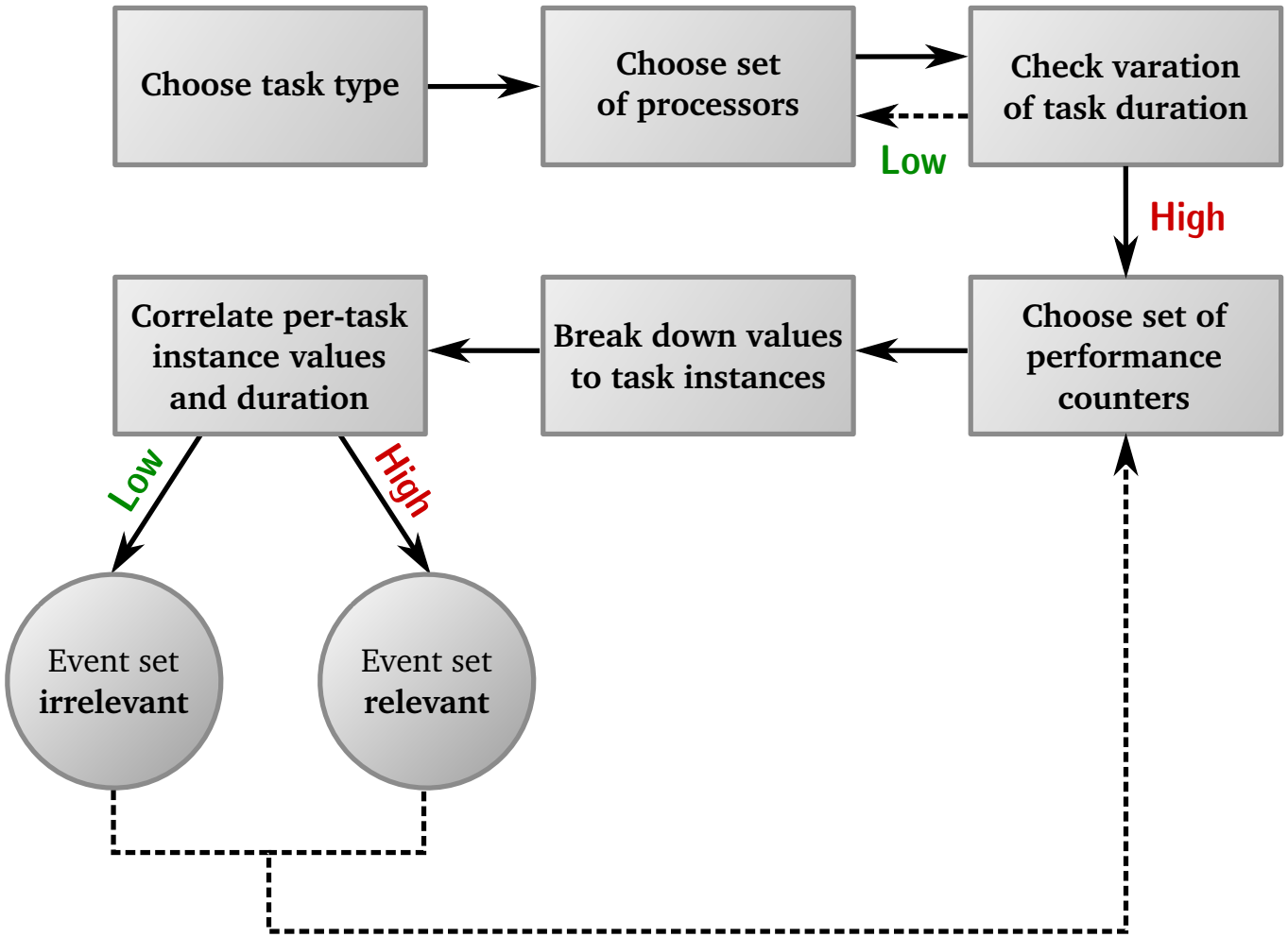
Per-interval analysis of parallelism & overhead



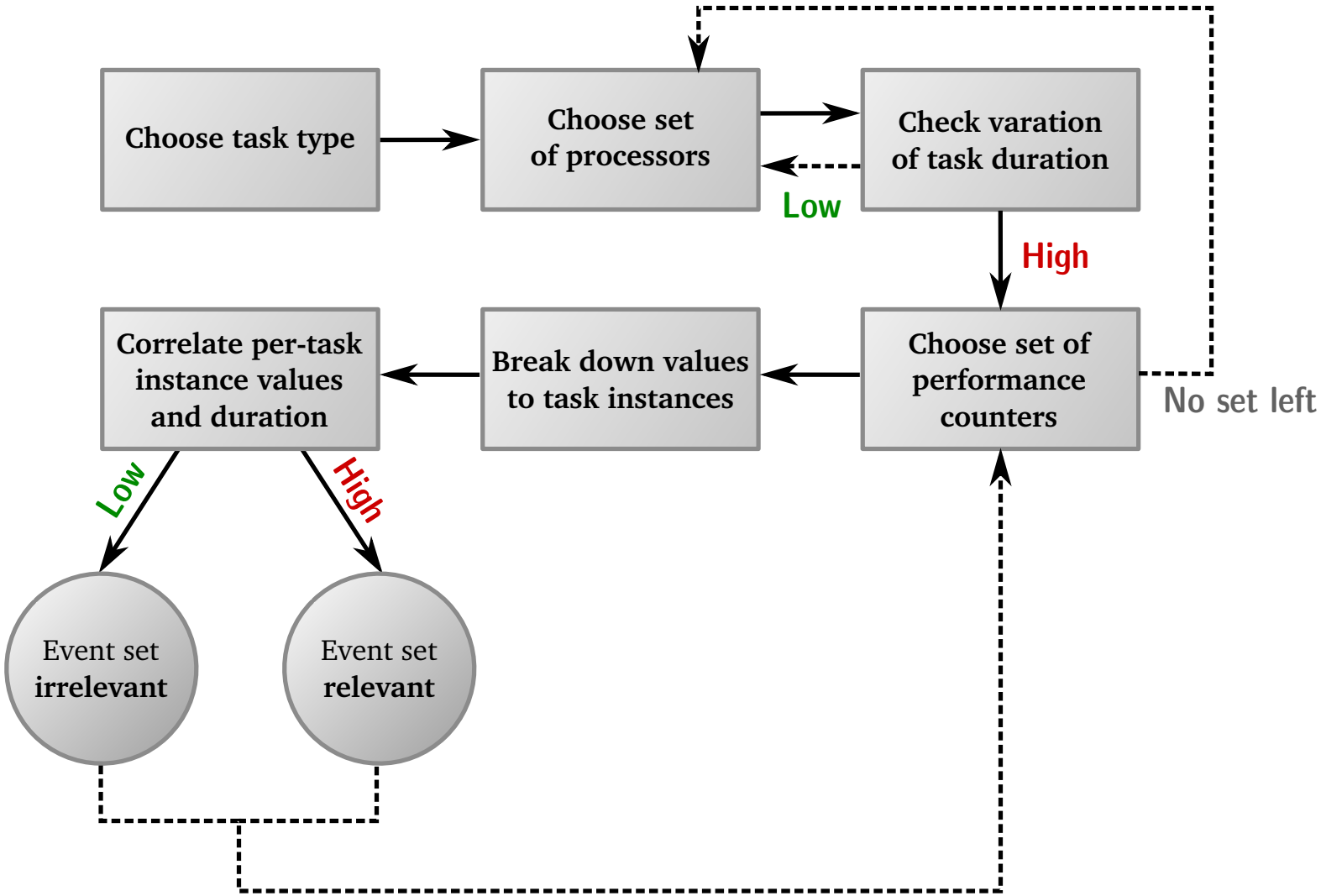
Per-interval analysis of parallelism & overhead



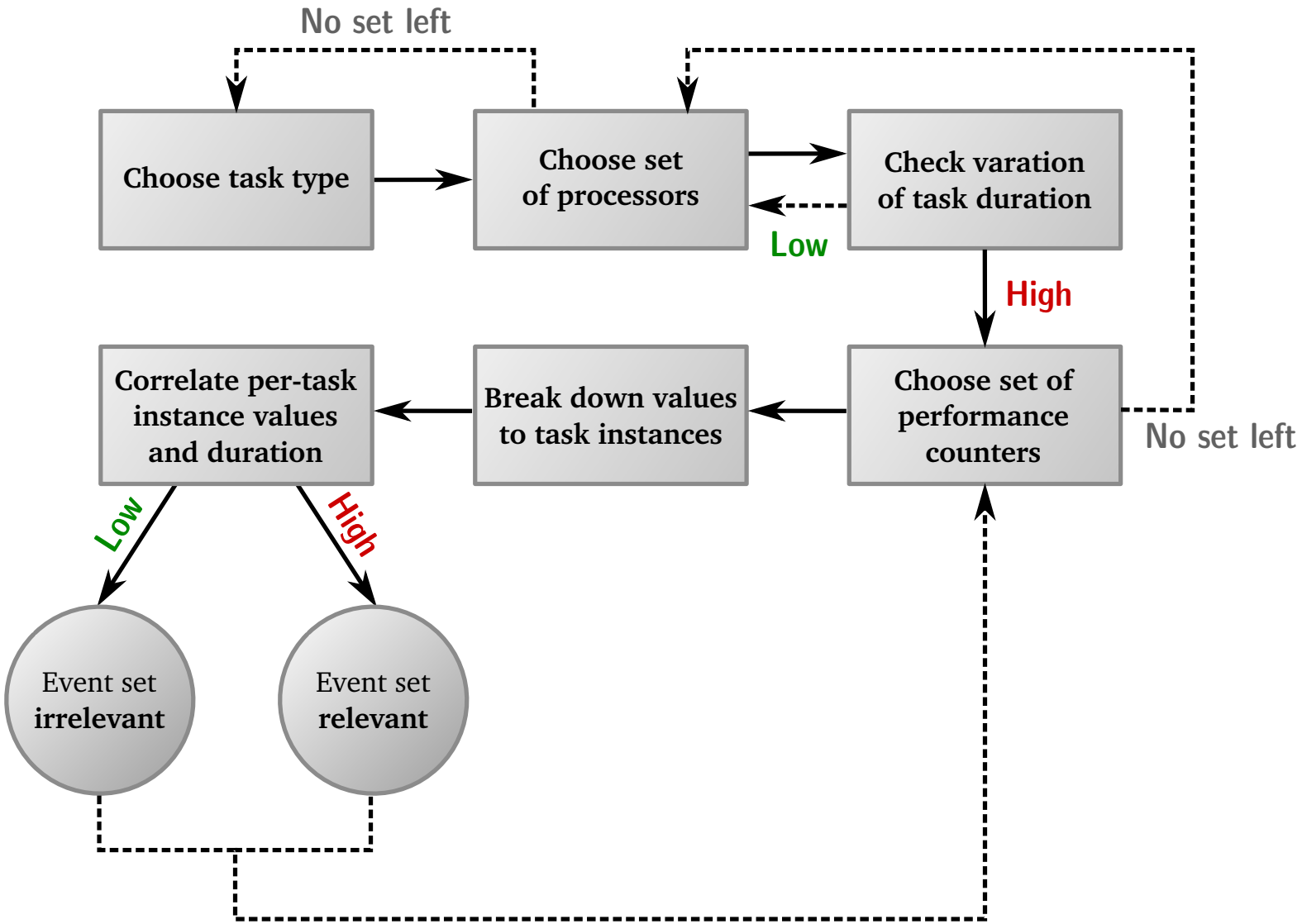
Per-interval analysis of parallelism & overhead



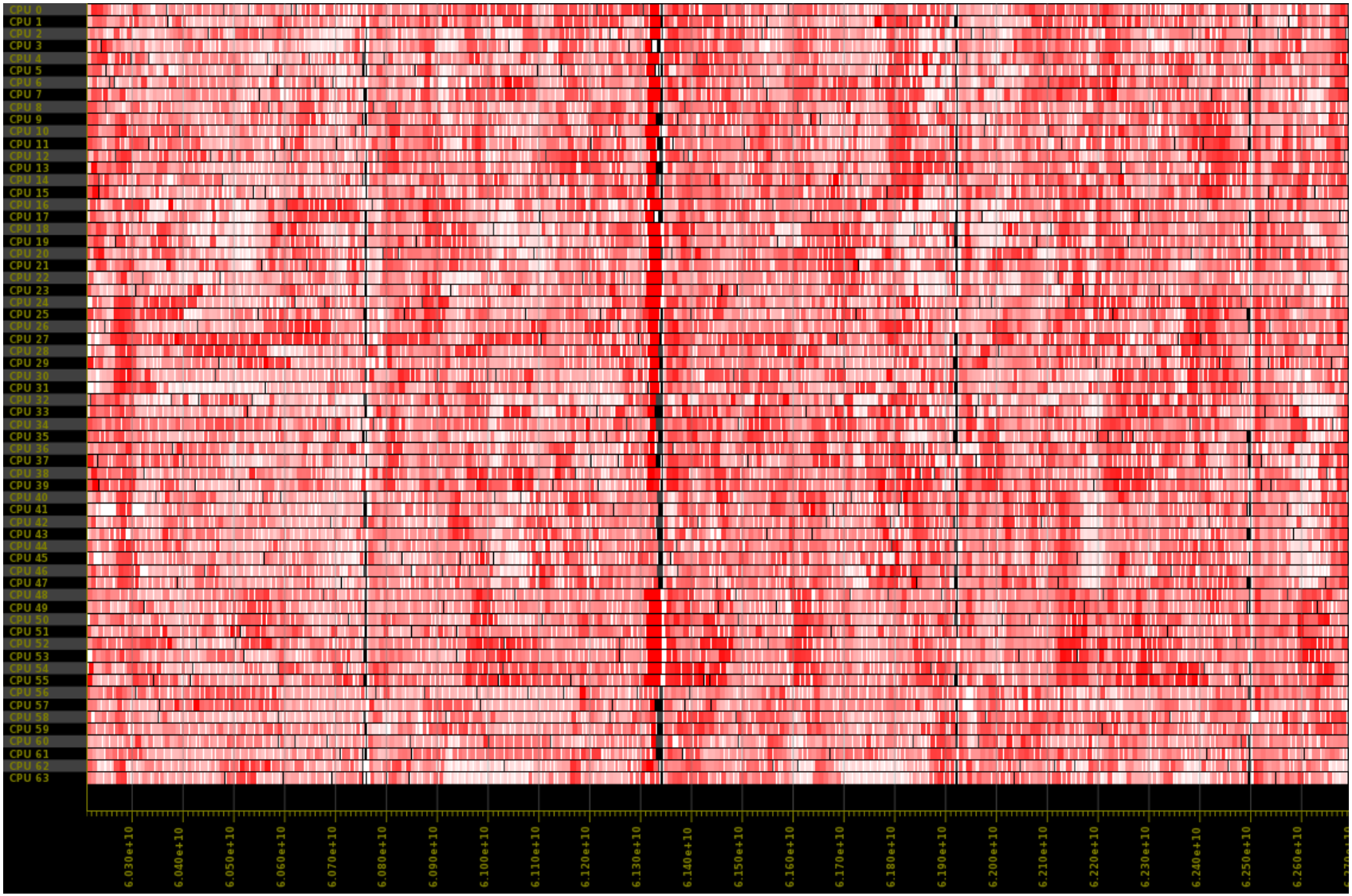
Per-interval analysis of parallelism & overhead



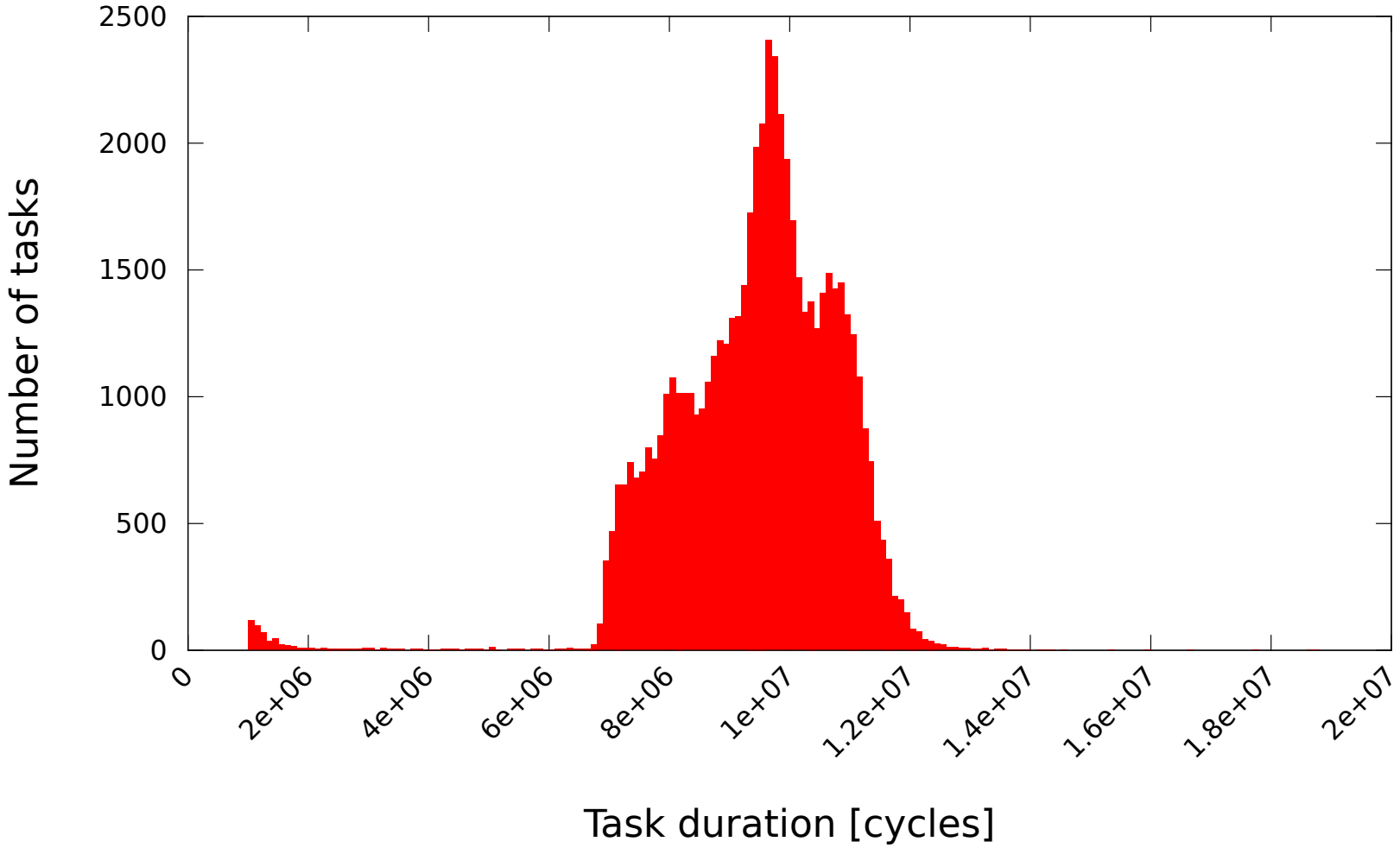
Per-interval analysis of parallelism & overhead



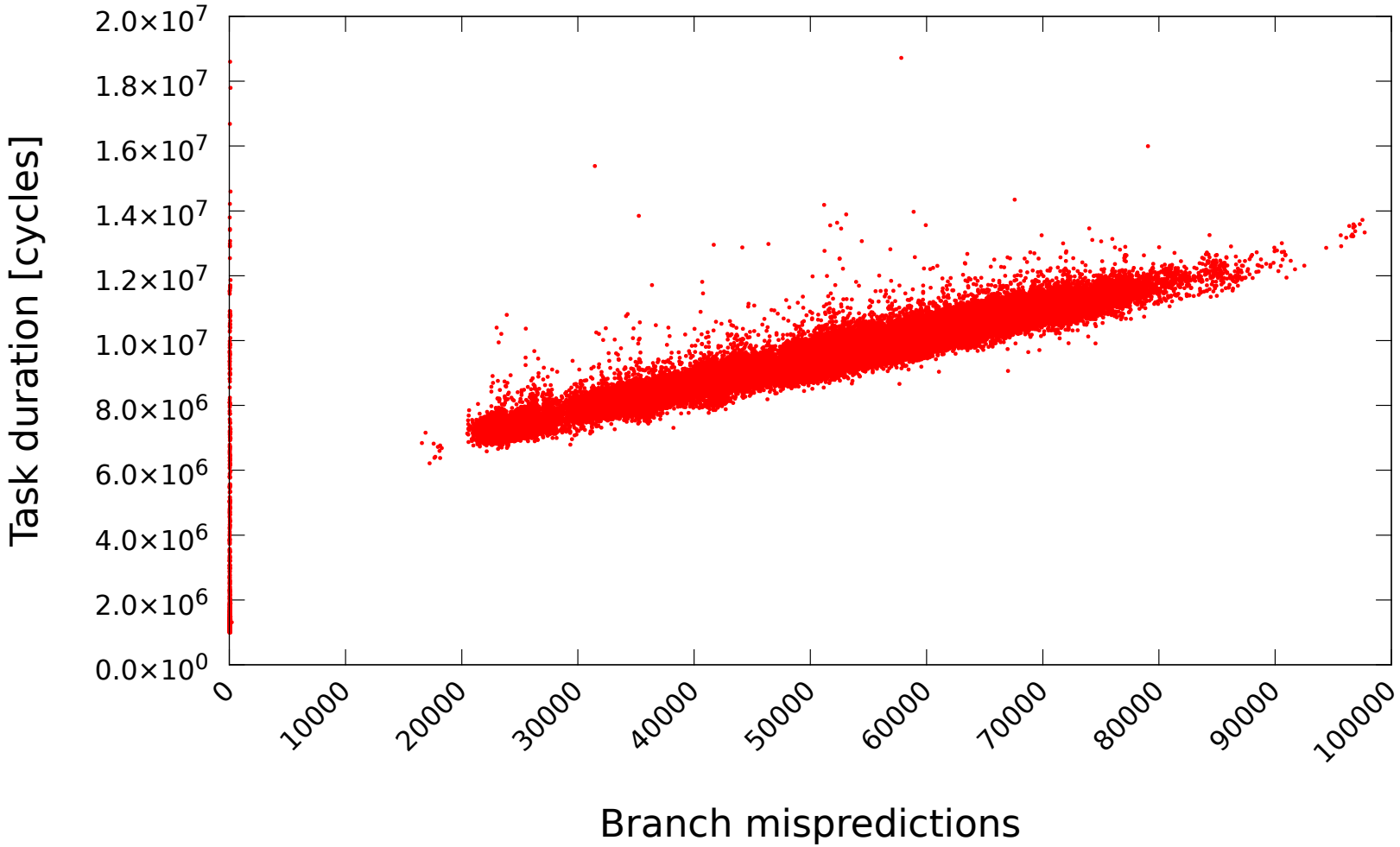
Example: K-means branch misprediction



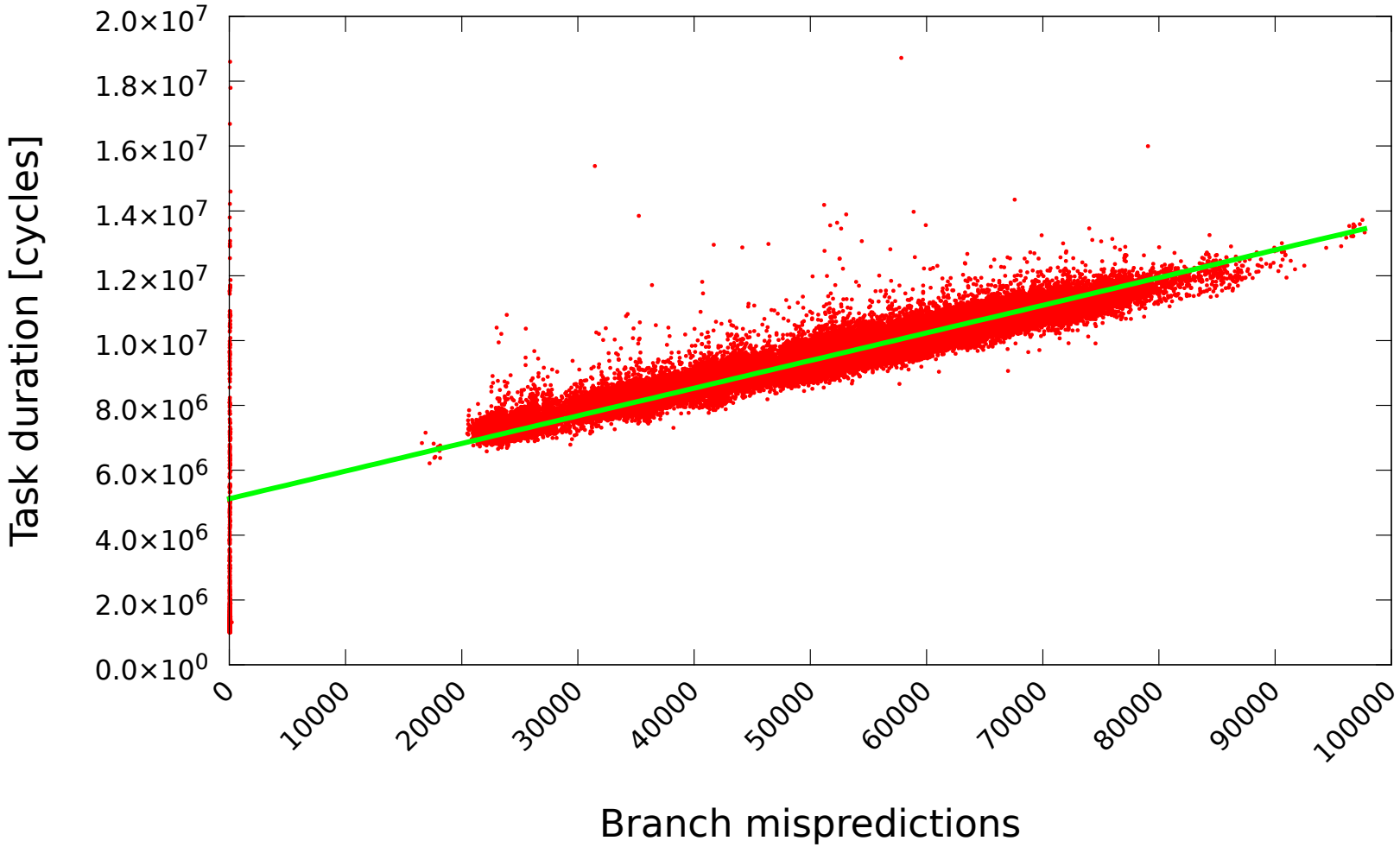
Example: K-means branch misprediction



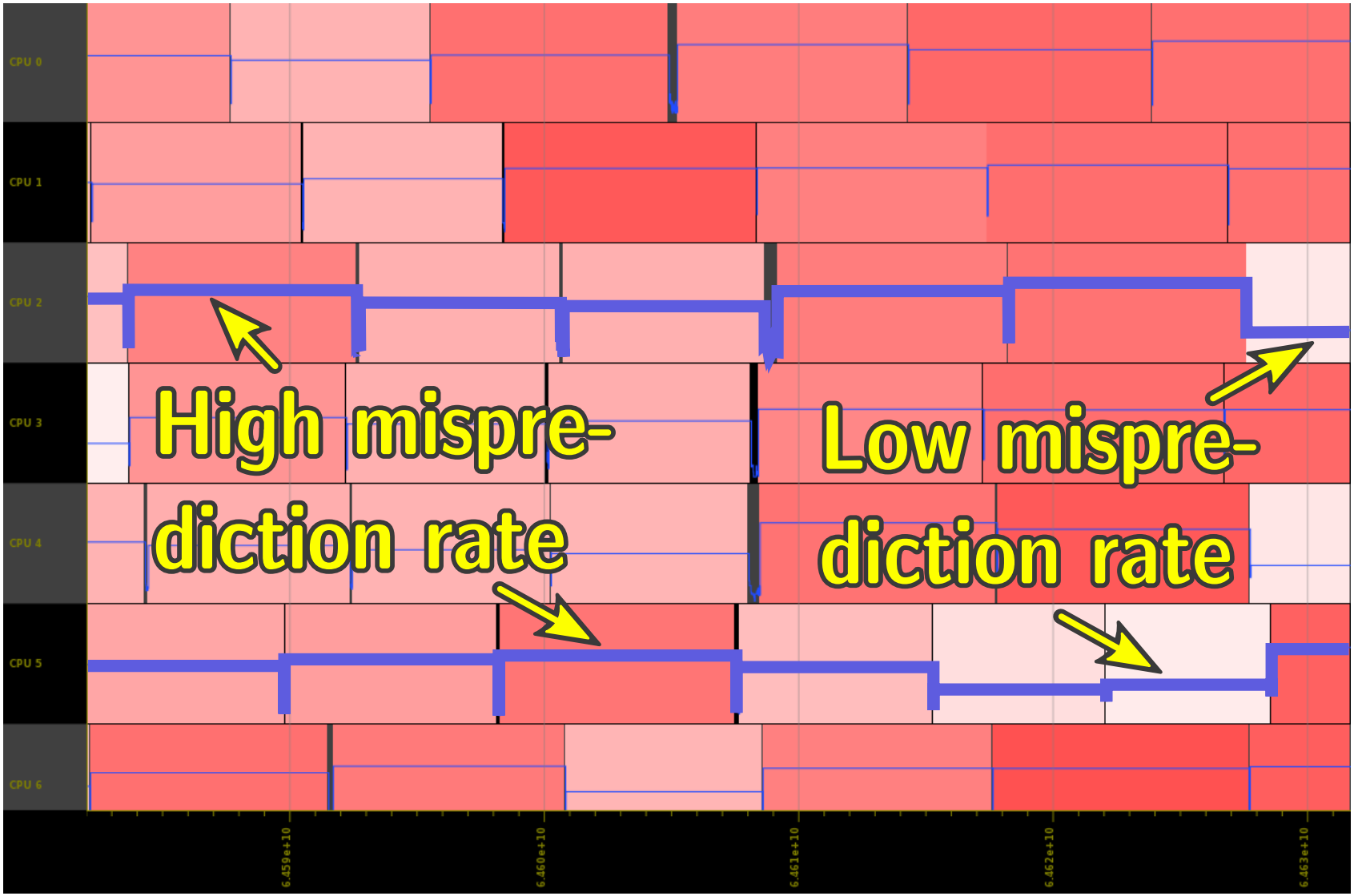
Example: K-means branch misprediction



Example: K-means branch misprediction



Example: K-means branch misprediction



Work in progress: Status

Analysis of parallelism

- Per-interval analysis of time spent on task execution
- Per-interval analysis of time spent in run-time states

Work in progress: Status

Analysis of parallelism

- Per-interval analysis of time spent on task execution
- Per-interval analysis of time spent in run-time states
- Support for thresholds
- Loop performing analysis on set of intervals

Work in progress: Status

Analysis of parallelism

- Per-interval analysis of time spent on task execution
- Per-interval analysis of time spent in run-time states
- Support for thresholds
- Loop performing analysis on set of intervals

Correlation of performance indicators

- Support for performance counters
- Task duration histogram

Work in progress: Status

Analysis of parallelism

- Per-interval analysis of time spent on task execution
- Per-interval analysis of time spent in run-time states
- Support for thresholds
- Loop performing analysis on set of intervals

Correlation of performance indicators

- Support for performance counters
- Task duration histogram
- Analysis of the variation of task durations
- Breaking down performance counter values to task instances
- Linear regression

Summary

Aftermath

- Tool for trace-based analysis of task-parallel programs
- Currently provides only support for manual analysis
- Available at `http://openstream.info/aftermath`

Summary

Aftermath

- Tool for trace-based analysis of task-parallel programs
- Currently provides only support for manual analysis
- Available at `http://openstream.info/aftermath`

Automatic analysis of parallelism based on thresholds

- Amount of time spent on task execution sufficiently high?
- If not, perform subsequent threshold-based analysis for states associated with overhead of the run-time system

Summary

Aftermath

- Tool for trace-based analysis of task-parallel programs
- Currently provides only support for manual analysis
- Available at `http://openstream.info/aftermath`

Automatic analysis of parallelism based on thresholds

- Amount of time spent on task execution sufficiently high?
- If not, perform subsequent threshold-based analysis for states associated with overhead of the run-time system

Automatic correlation of performance indicators

- Indicate which events are relevant
- Break down counter evolution to task instances
- Correlate with task duration