Scaling Score-P to the next level

Daniel Lorenz

Scalable Tools Workshop, Lake Tahoe, August 1, 2016

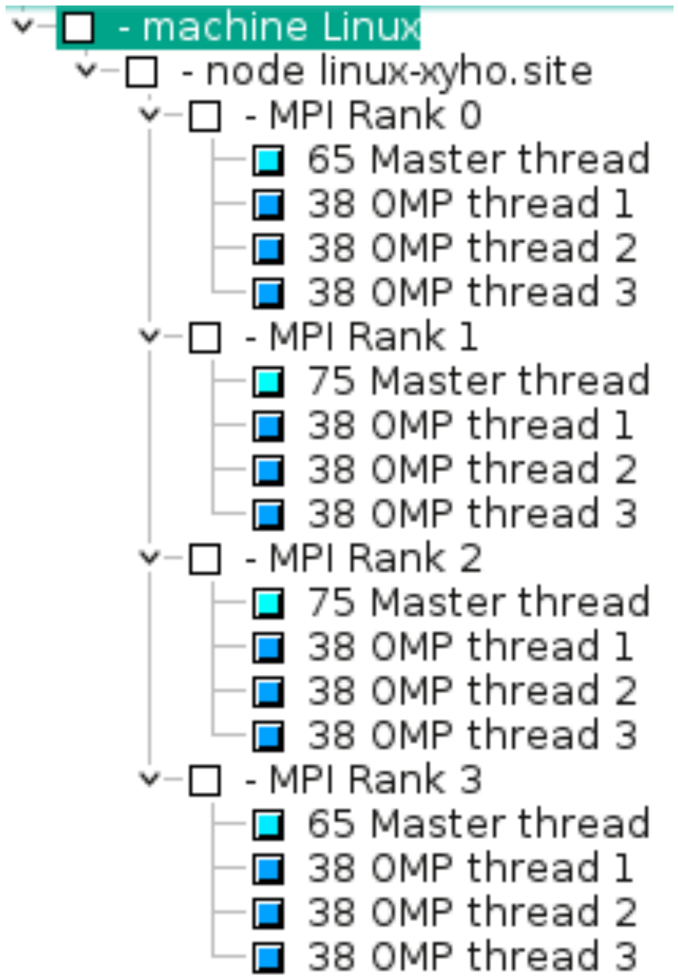
Approaches

Scalable system tree description

Automatic thread-level aggregation

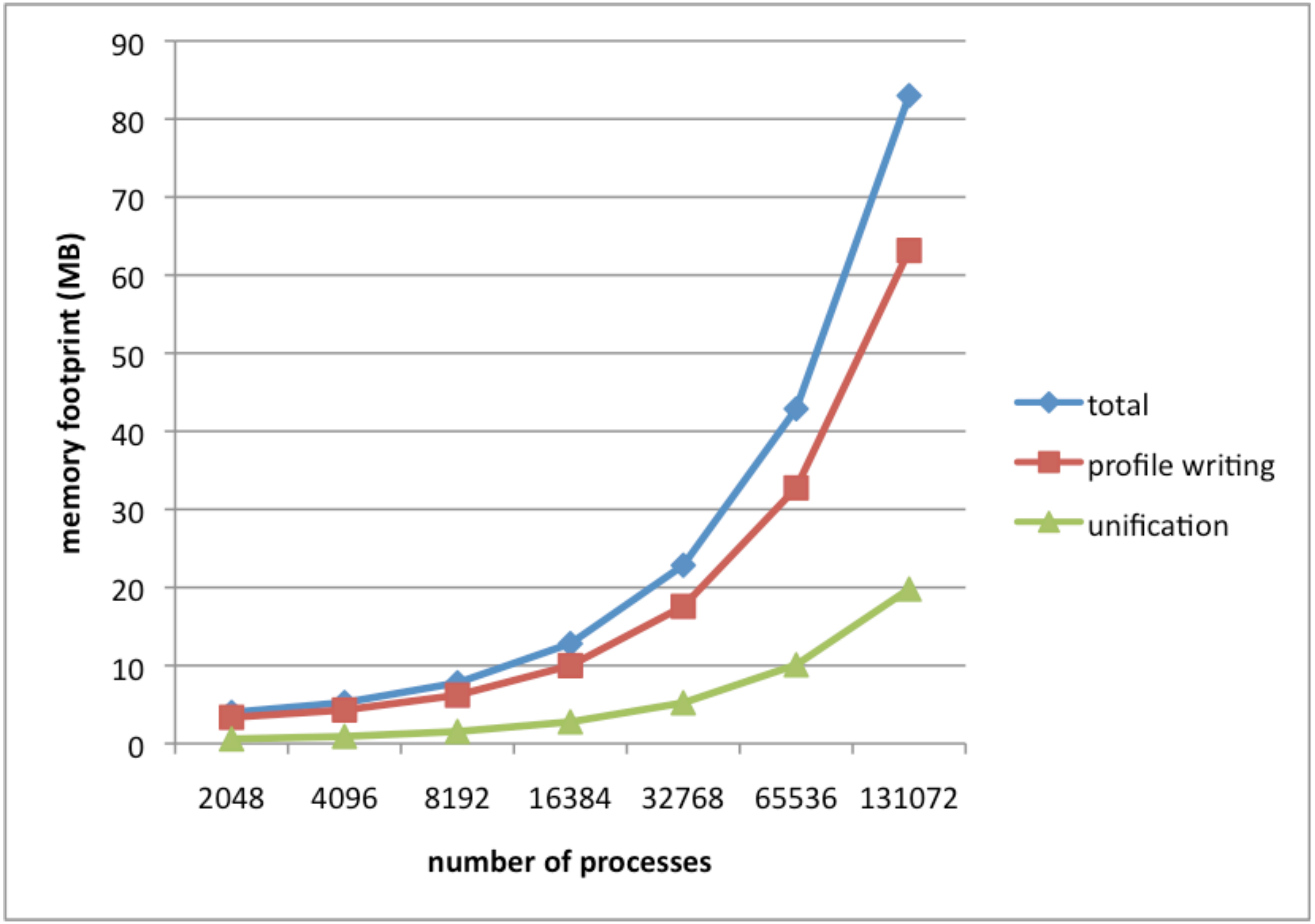
System tree definitions

Single-node definitions: One data record per system tree element



Score-P finalization memory footprint

TECHNISCHE UNIVERSITÄT DARMSTADT



The goal

A system tree description with a memory footprint that does not depend on the system size

An parallel algorithm to create the new system tree description from local information with constant memory footprint

Sequence definitions (1)

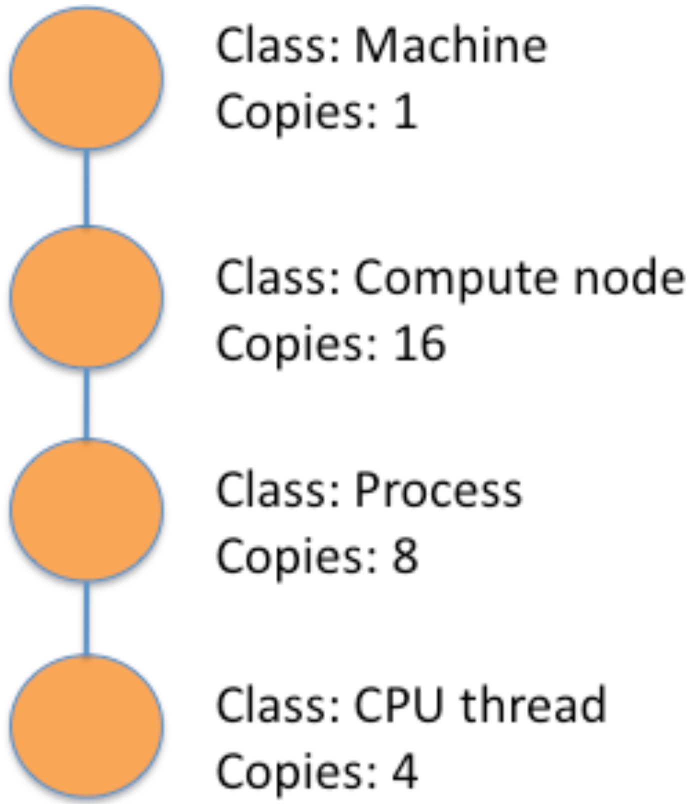
Based on the PERI-XML proposal

Exploit regularity of systems

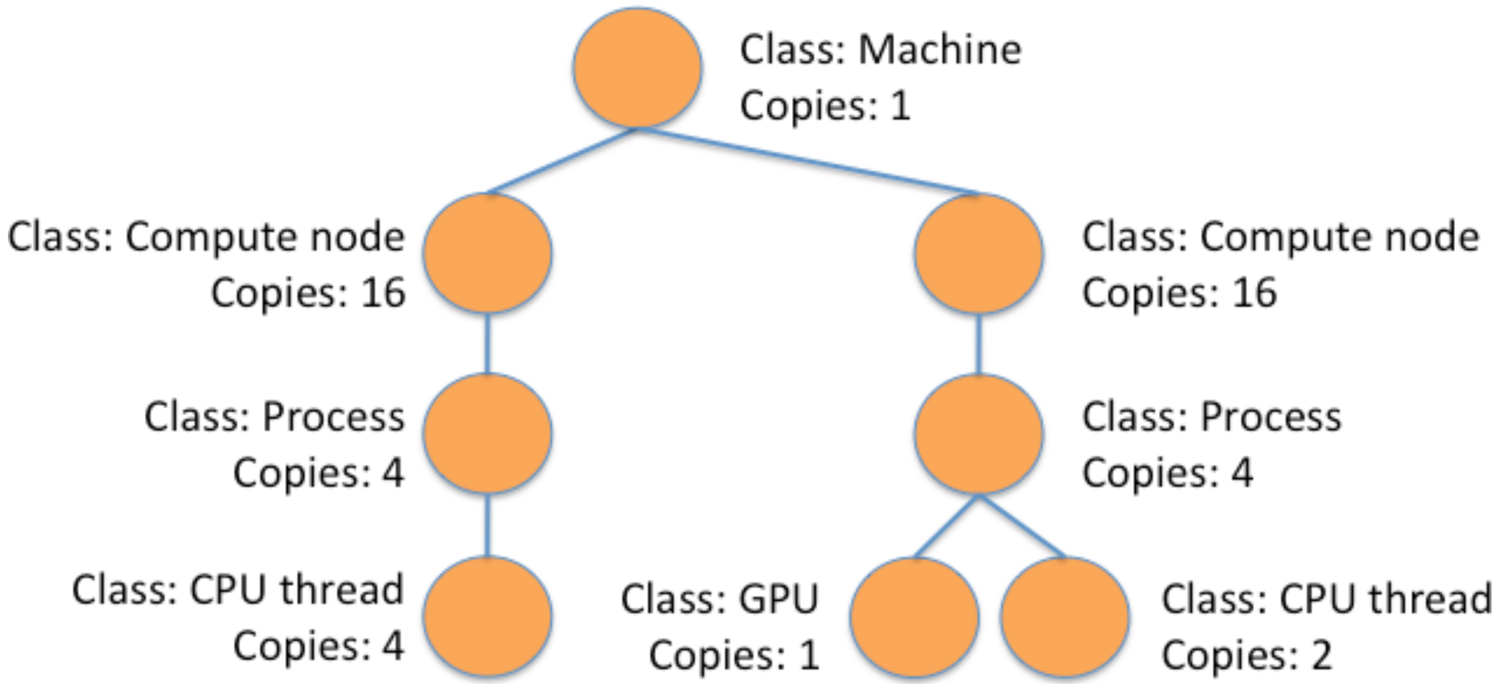
Constant size for regular systems

A depth-first traversal of the system tree provides enumeration

Can be used as index to reference individual nodes



Sequence definitions (2)



Memory footprint

O(size of definitions + size of communicator)

- For regular systems: Size of definitions in O(1)
- MPI communicators: Memory footprint can be O(1)
- Current implementations usually O(# processes)
- Expect MPI communicator implementations to scale to the system size

Computational complexity

TECHNISCHE UNIVERSITÄT DARMSTADT

$O(\log P * (t_{P2P}(P) + t_{Comm}(P)) * S * D)$

P: Number of processes

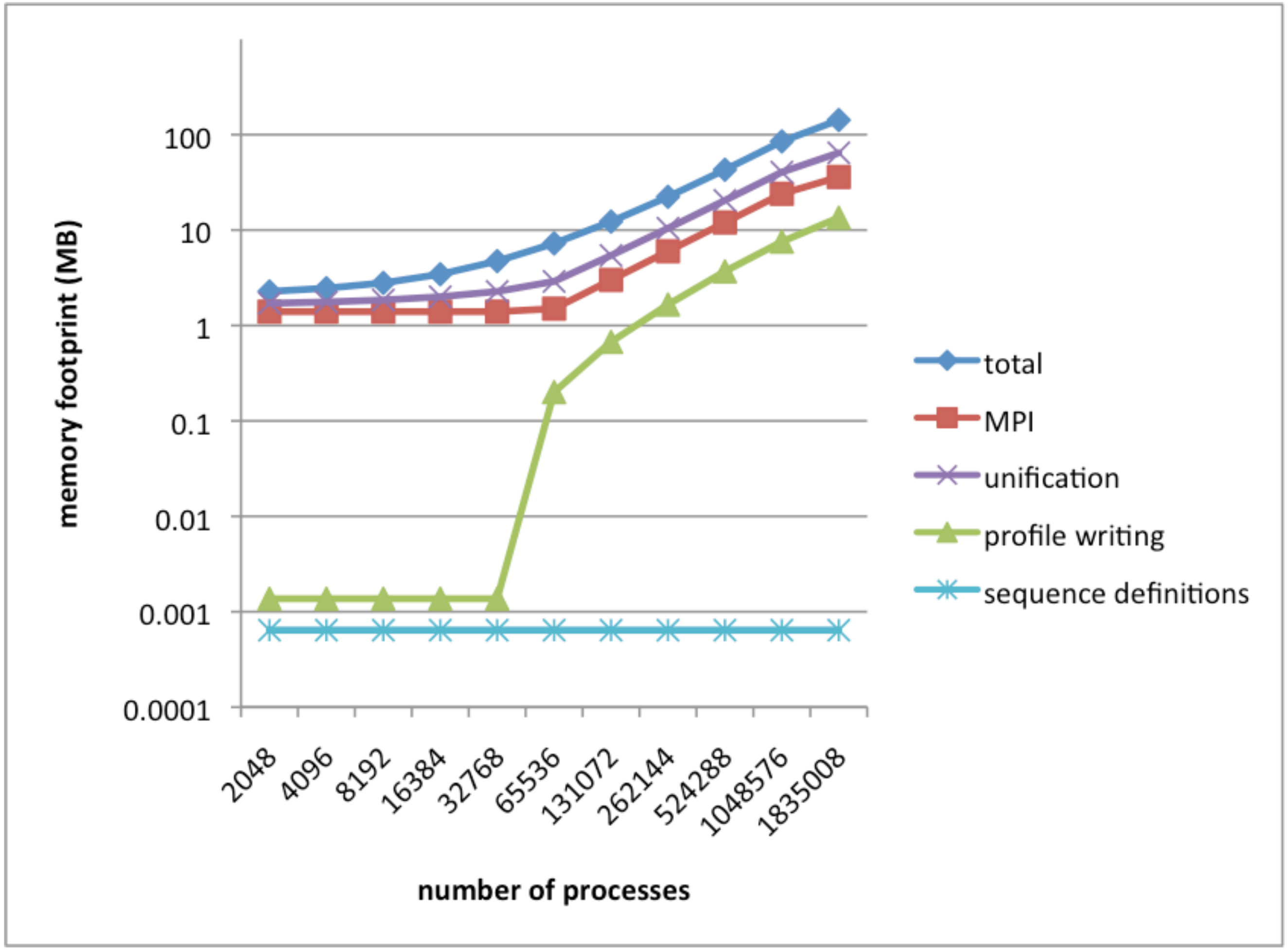
t_{P2P}: Time of a peer-to-peer communication

t_{Comm}: Time for communicator creation

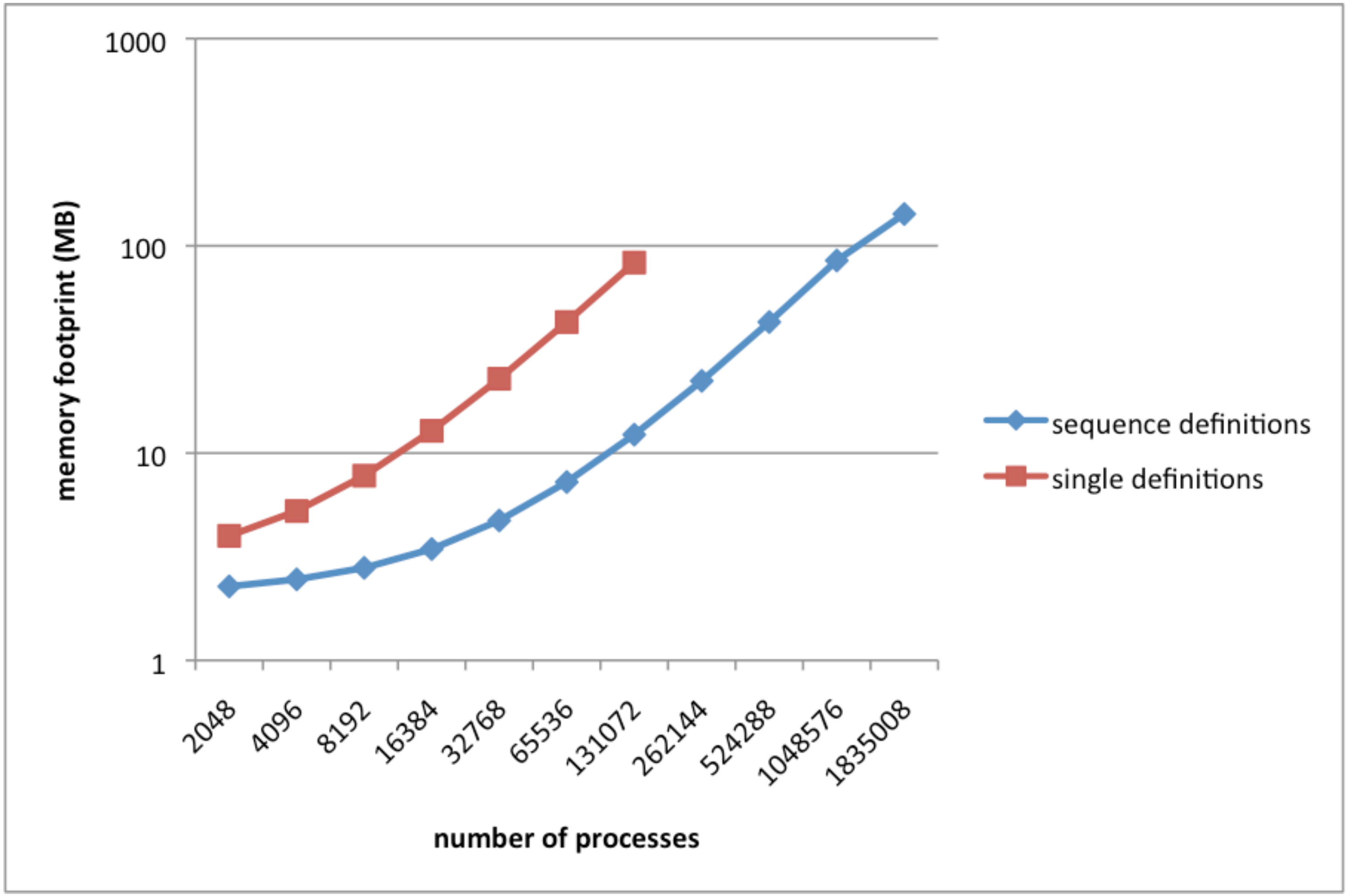
S: Size of the sequence definitions

D: Depth of the system hierarchy

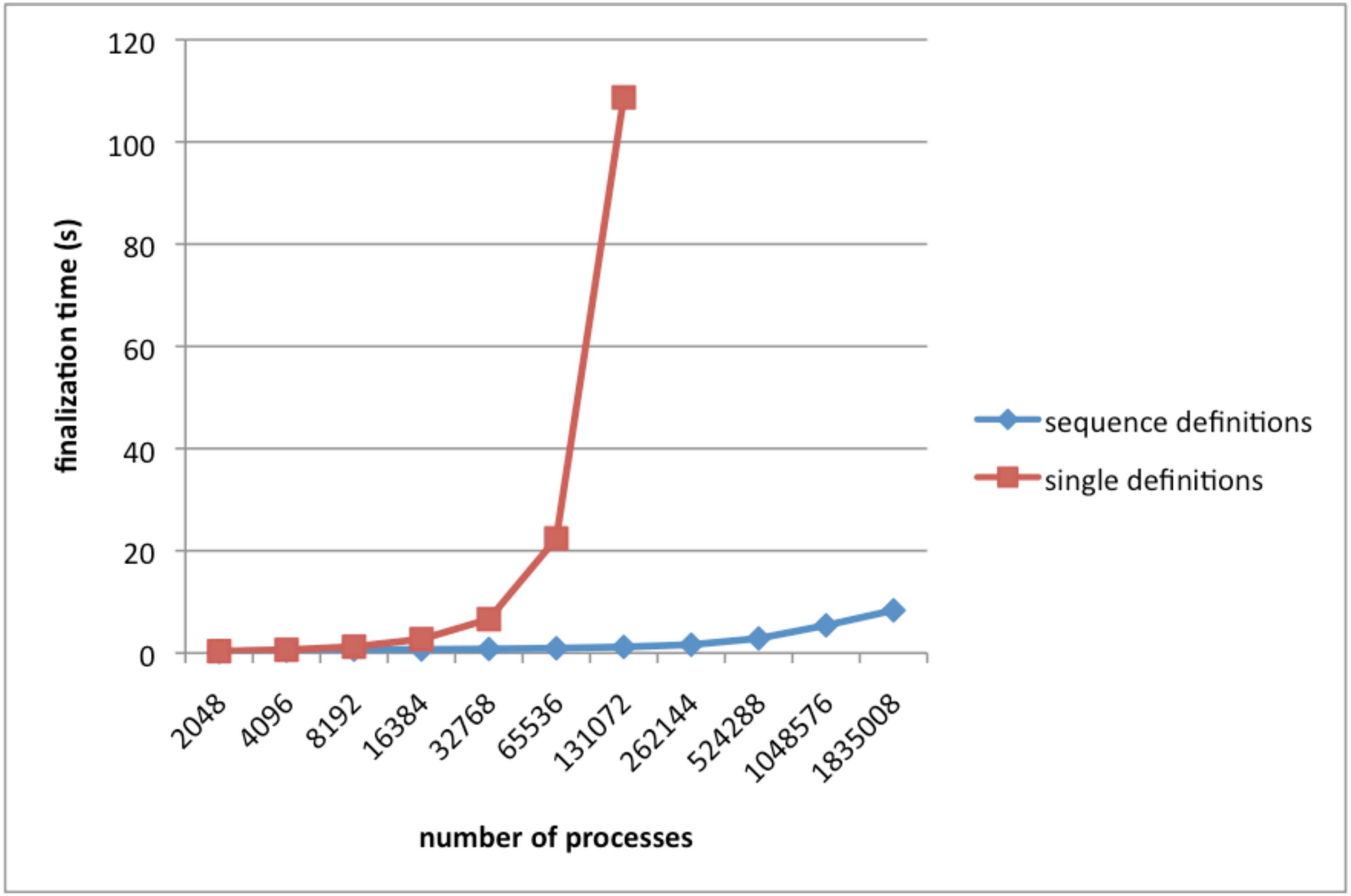
Finalization memory footprint (hello world)



Memory footprint comparison (hello world)



Finalization time (hello world)



Thread-level aggregation

Which data is needed?

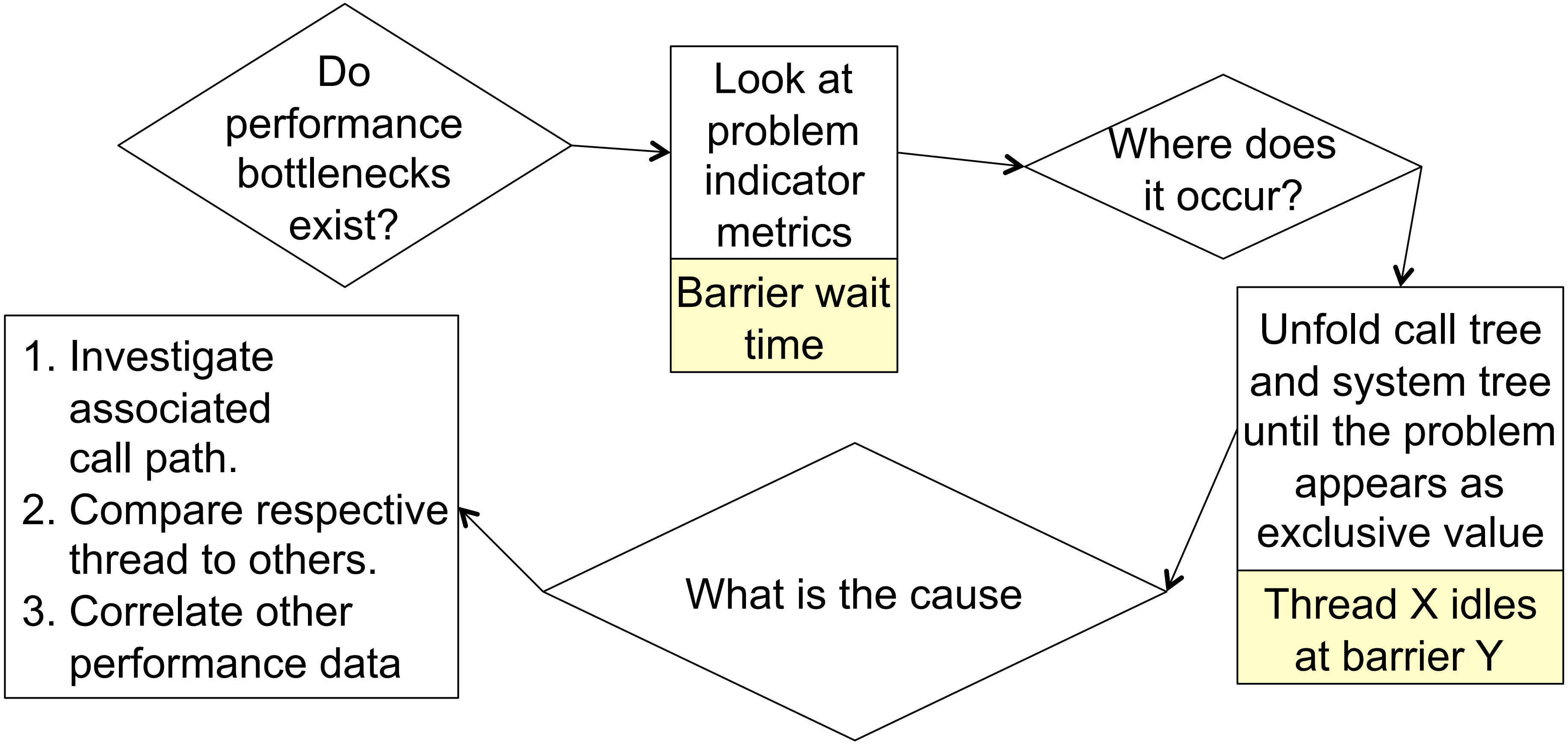
- Analysis workflow with CUBE

The compression strategies

Evaluation of the compression ratio

Evaluation of the information loss

What information is needed?



Aggregation strategies

SUM:

- Aggregates the data of all threads within a process

SET:

- Keeps also statistical data about the value distribution among the threads

KEY:

- Keeps the three so called key threads
- Aggregates all others

CALLPATH:

- Clusters threads that have the same call tree structure
- Aggregates all threads within a cluster.

SET

TECHNISCHE UNIVERSITÄT DARMSTADT

Contains:

- Sum
- Minimum
- Maximum
- Sum of squares (to calculate standard deviation)
- Number of threads

No correlations between call path and metrics possible

KEY

Need to improve the performance of slowest thread

You may want to compare it to the other extreme, the fastest thread

The initial thread plays often an distinct role

Aggregate other threads

- They can provide an average value for comparison

Slowest/Fastest calculation

- Classify regions
- Consider measured time in regions that are considered to do work

CALLPATH

Aggregate all threads that have the same call tree structure

- Both have at least one visit to a call path
- The number of visits is not compared

Number of resulting clusters depends on application

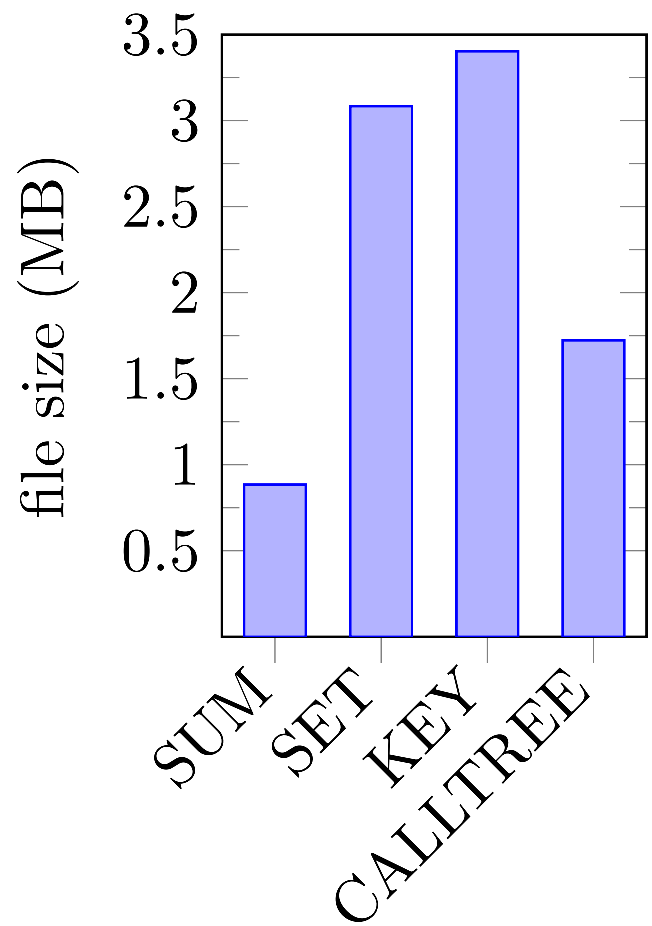
- Compression ratio may vary

Compression (1)

Constant files sizes

- Independent of number of threads per process
- Same number of locations stored

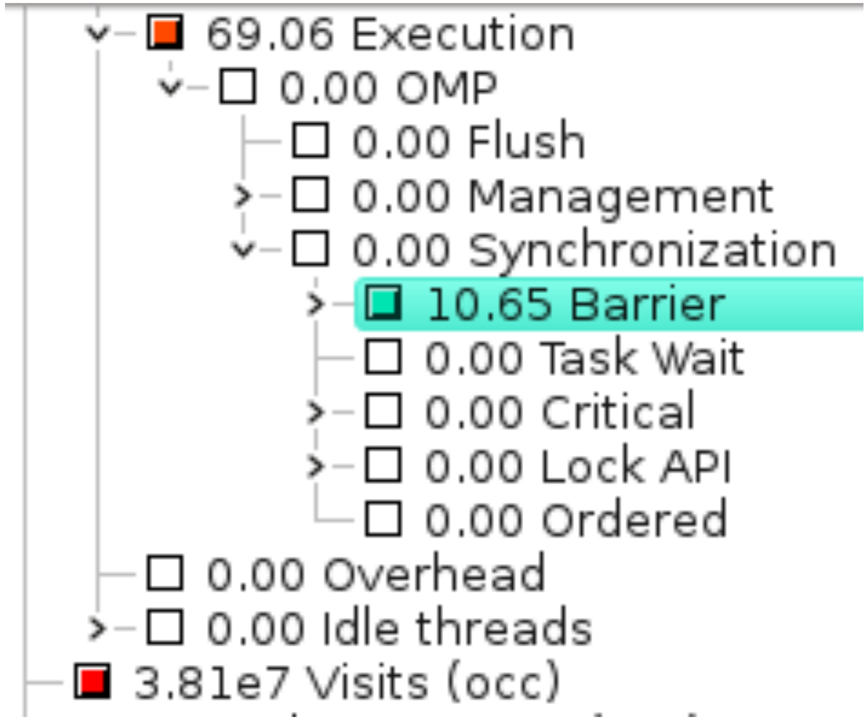
- Compression ratio varies with number of threads
- SET a little smaller than KEY because the CUBE record stores number of threads as 32 bit value.



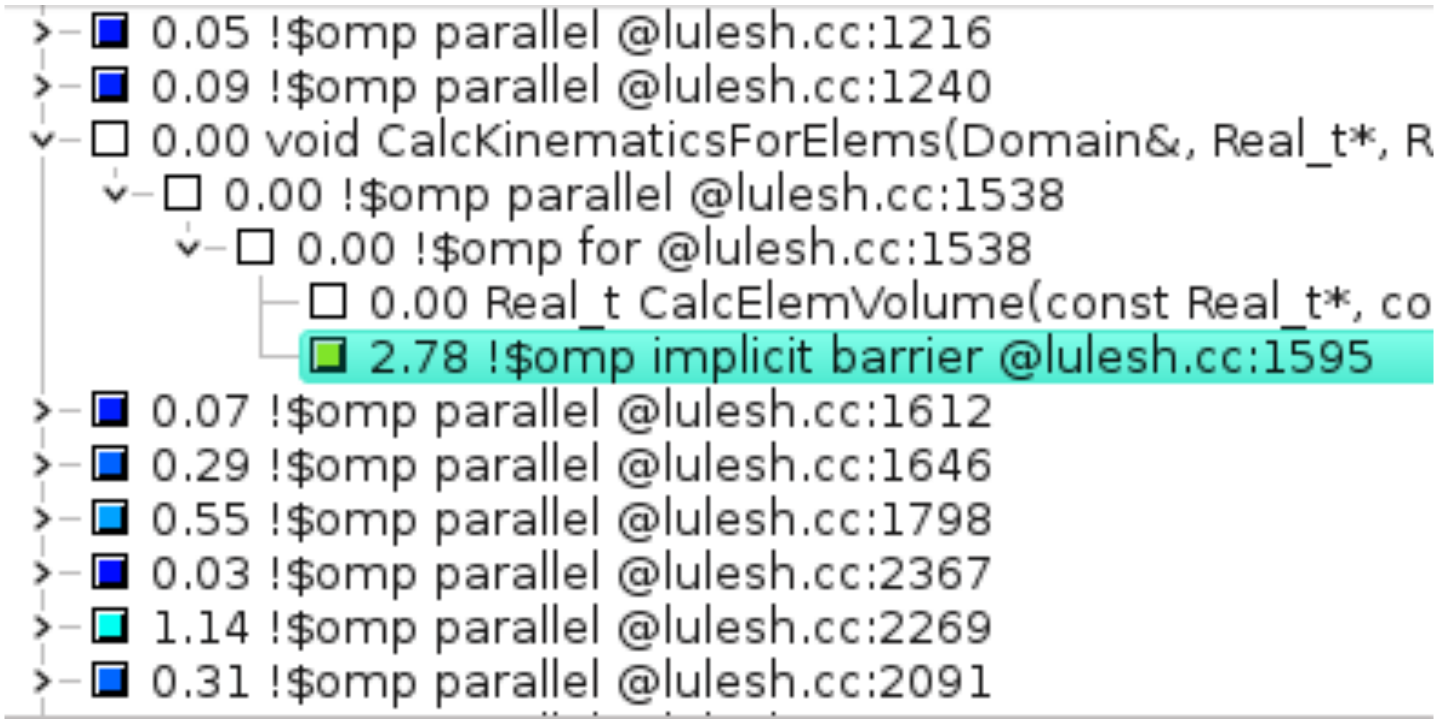
Imbalance – Test case

Lulesh 2.0

- Insert imbalance in a parallel region via too large schedule clause



Imbalance – Call tree



Imbalance – System trees

original

- machine Linux
  - node swat
    - Process
      - 0.10 Master thread
      - 0.02 OMP thread 1
      - 0.08 OMP thread 2
      - 0.01 OMP thread 3
      - 0.10 OMP thread 4
      - 0.51 OMP thread 5
      - 1.00 OMP thread 6
      - 0.96 OMP thread 7

KEY

- machine Linux
  - node swat
    - Process
      - 0.10 master thread
      - 1.00 fastest thread
      - 0.00 slowest thread
      - 1.63 aggregated worker threads

SUM

- machine Linux
  - node swat
    - 2.94 Process

CALLTREE

- machine Linux
  - node swat
    - Process
      - 0.14 cluster 0
      - 2.75 cluster 1

SET

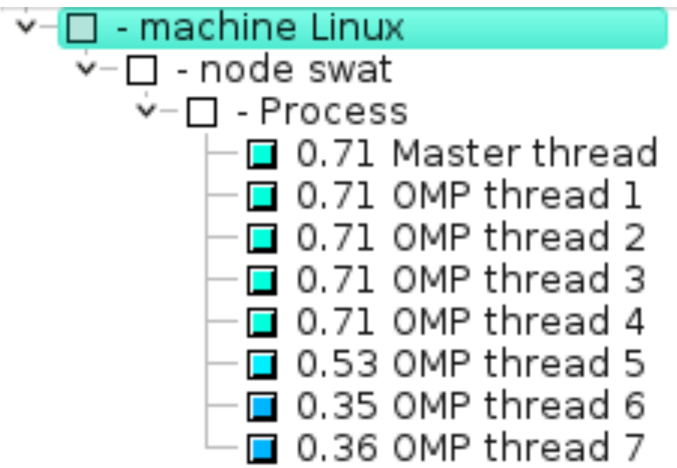
- machine Linux
  - node swat
    - (8,3590848,1019191267):3.64175e+08,3.94046e+08 Process

Imbalance – Loop Body

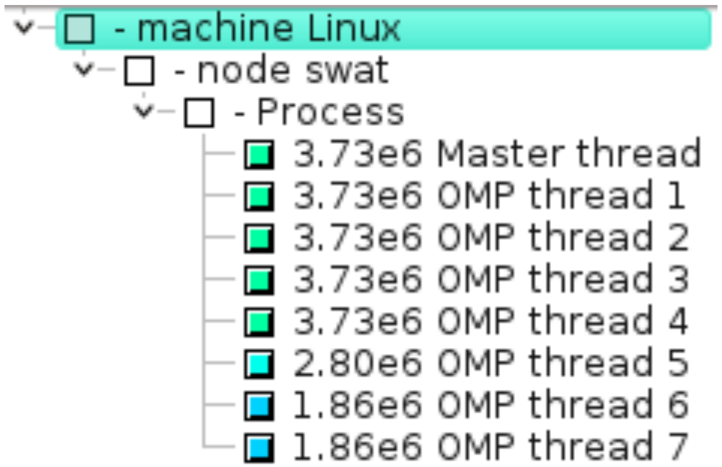
Correlate visits and execution time for the loop body

- Some threads have less iterations
- Same threads spend less time in loop

Execution time



Visits



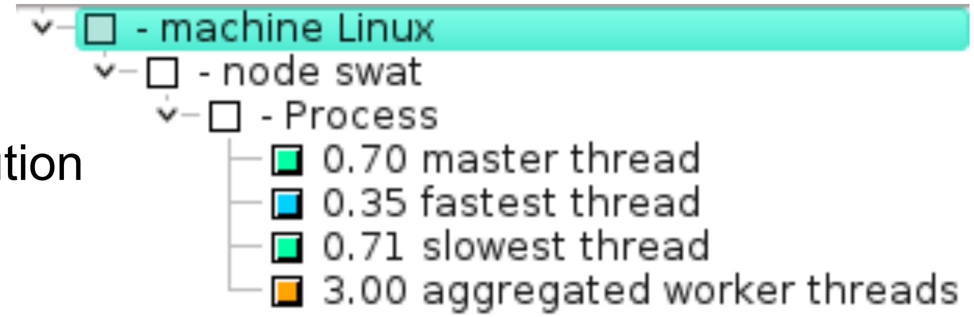
Imbalance – Loop Body

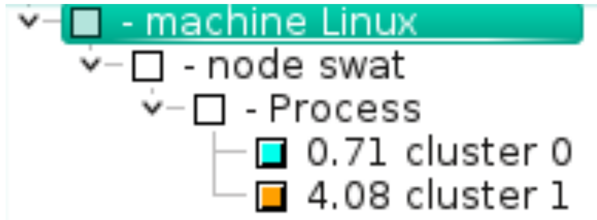
TECHNISCHE UNIVERSITÄT DARMSTADT

KEY

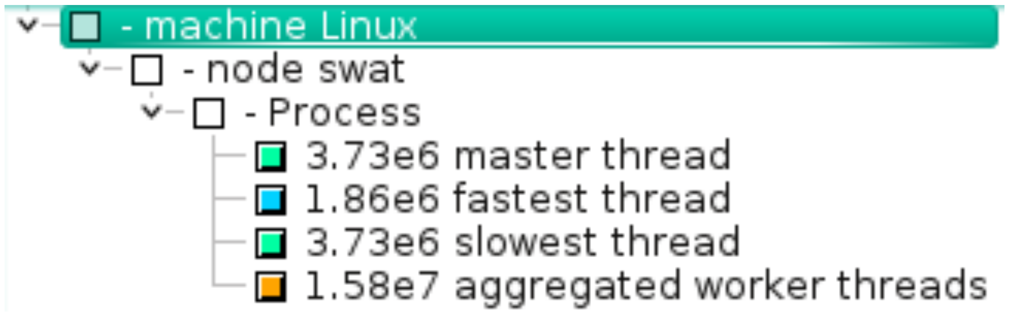
CALLTREE

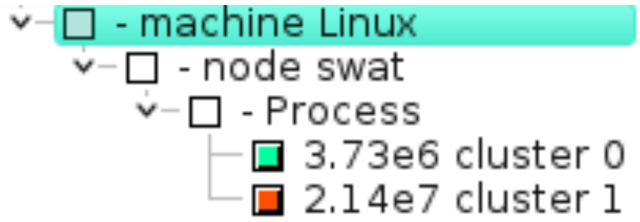
Execution time





Visits





Other test cases

TECHNISCHE UNIVERSITÄT DARMSTADT

Task granularity

Lock contention

Memory bandwidth saturation

Per thread resolution less important

Imbalance is the hard case for thread aggregation

Conclusion

SUM:

- Best compression
- Good for analysis where thread resolution is not necessary

KEY:

- Possibility to find the most limiting bottleneck

SET:

- Similar compression to KEY
- Less correlation possibilities than KEY

CALLPATH:

- Non-optimal cluster criteria
- Promising approach

Thank you for your attention!

This material is based upon work supported by the US Department of Energy under Grant No. DE-SC0015524