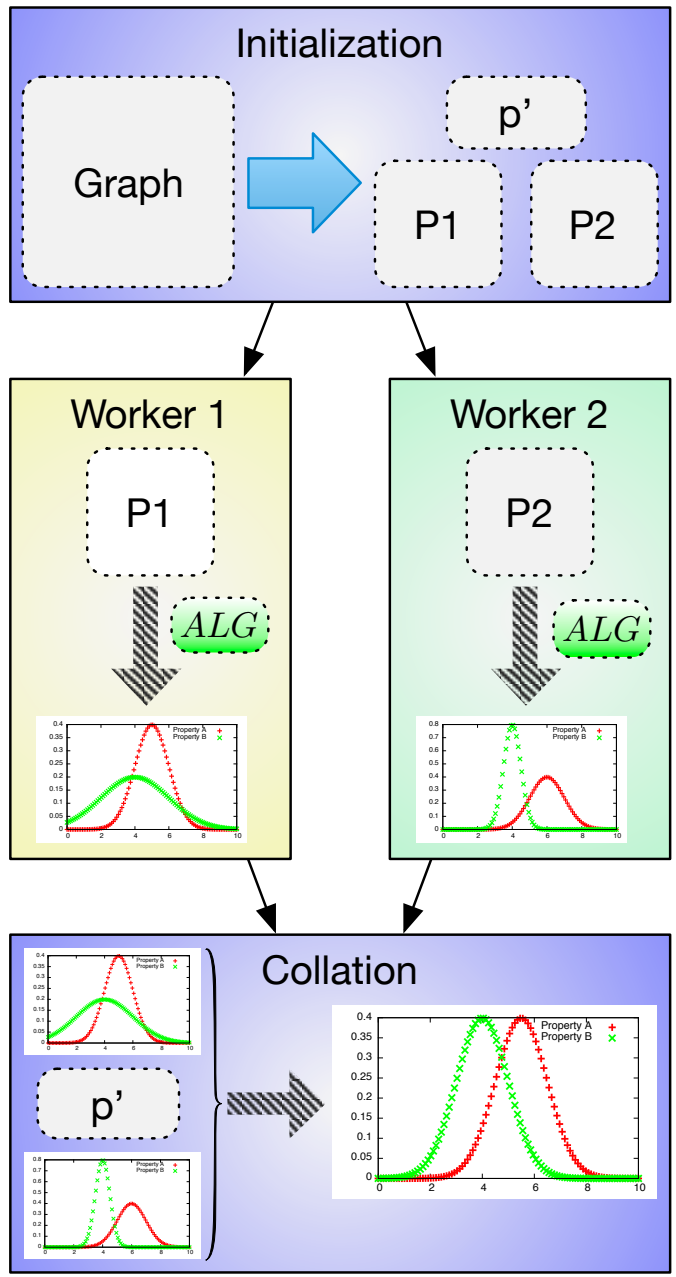
Parallel Stream-based Graph Analysis

Benjamin Schiller, Datenschutz und Datensicherheit, TU Dresden

Approach



partitioning as initialization

batches

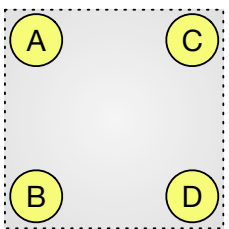
how to compute is up to each thread

after processing batch

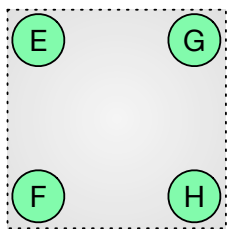
collation could be parallelized

Partition Type

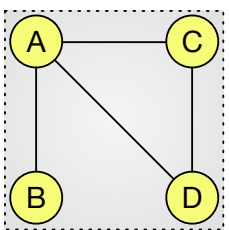
Non Overlapping



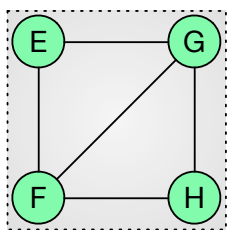
P1 = {A, B, C, D}



P2 = {E, F, G, H}



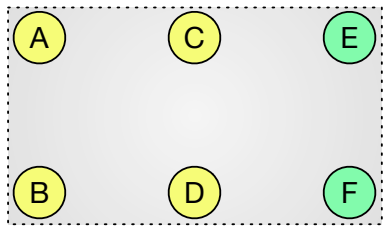
G_P1



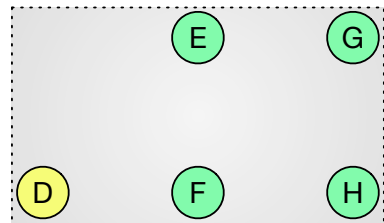
G_P2

+ no duplication of data
- collation complexity

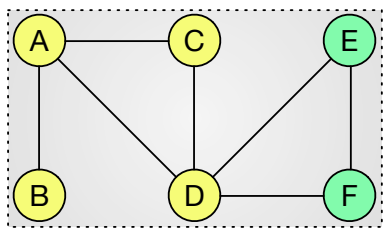
Overlapping



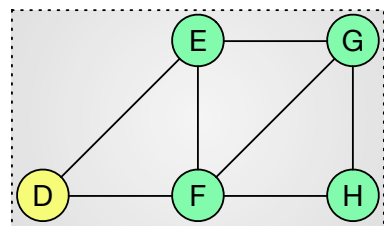
N(P1) = {A, B, C, D, E, F}



N(P2) = {D, E, F, G, H}



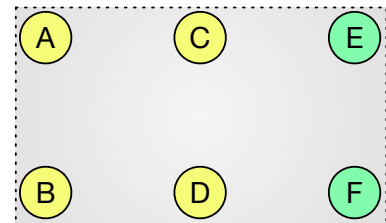
G_N(P1)



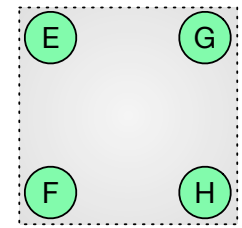
G_N(P2)

+ collation complexity
- memory overhead
- computation overhead

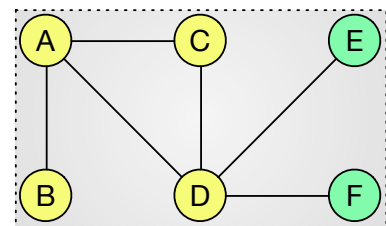
Node Cut



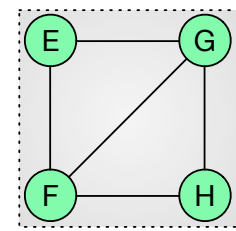
N(P1) = {A, B, C, D, E, F}



P2 = {E, F, G, H}



G_N(P1)



G_P2

+ collation complexity
- no memory overhead
- computation overhead

Algorithm Types

- Snapshot-based
- Batch-based
- Stream-based

- Snapshot-based (overlapping)
- Batch-based (overlapping)
- Stream-based (overlapping)

Partitioning Scheme

- BFS
- DFS
- EQUAL_SIZE
- LPA
- RANDOM

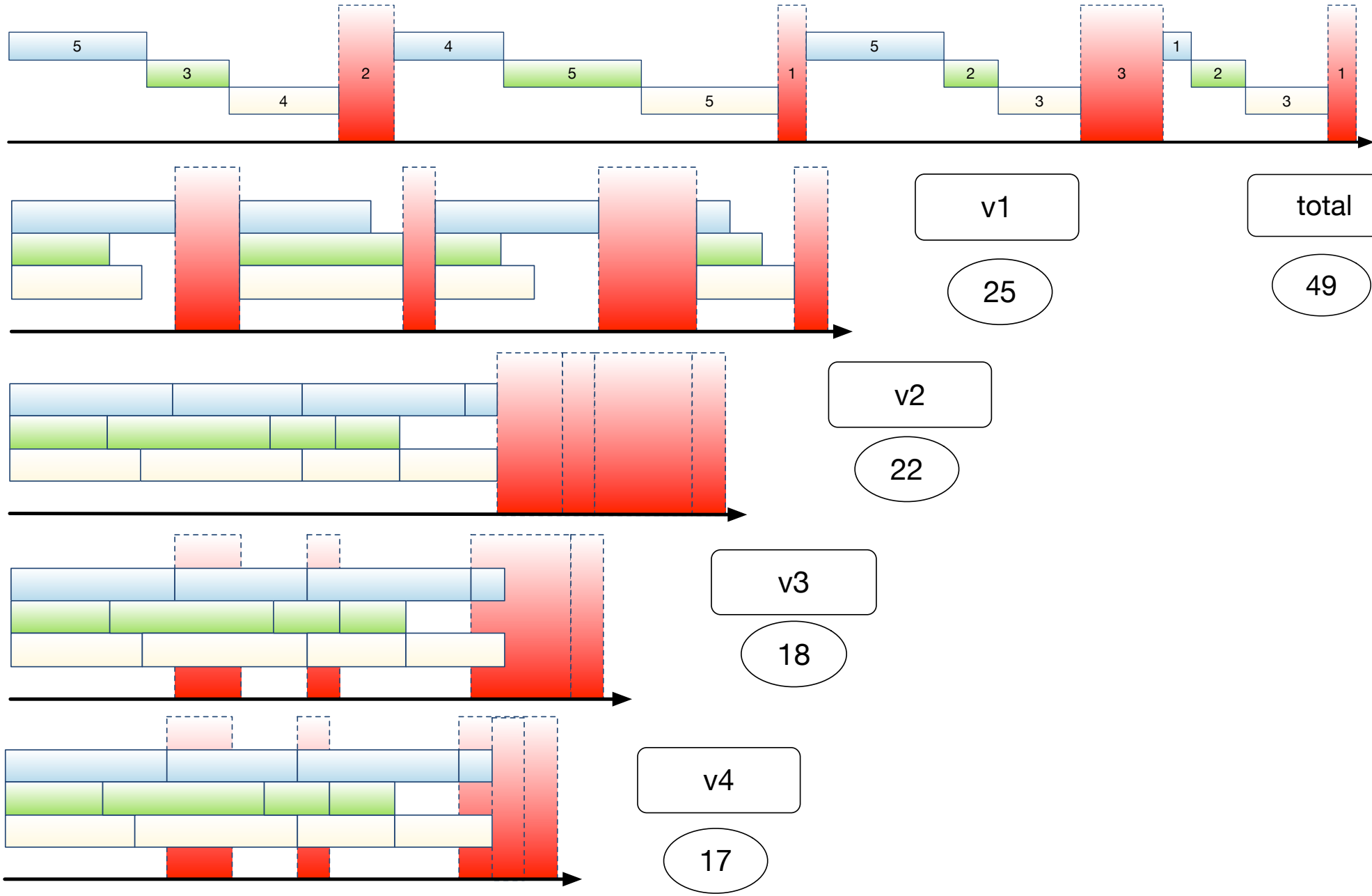
Node Assignment

- FIRST_EDGE
- RANDOM
- ROUND_ROBIN

Evaluaton so far

- Statistics (all per thread / per batch)
  - Analysis runtimes
  - |V|, |E|
  - Collation runtimes
  - Total runtimes (v{0..4})
- Statistics (depending on partitioning type)
  - NON_OVERLAPPING: external edges
  - OVERLAPPING: auxiliary nodes, auxiliary edges
  - NODE_CUT: cut nodes

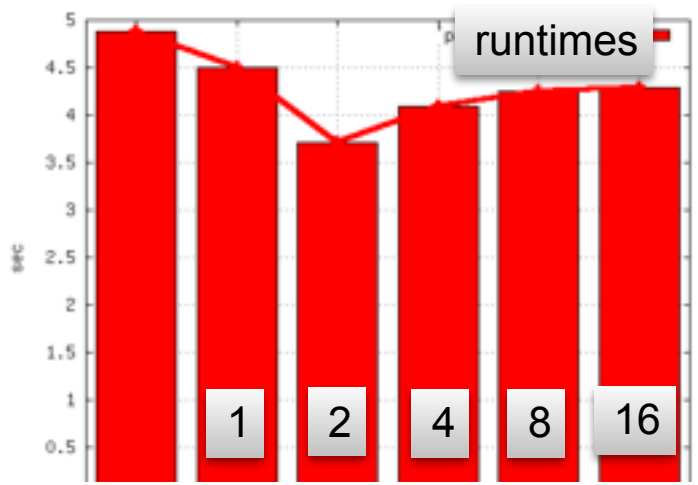
Sequential Evaluation / Approximation



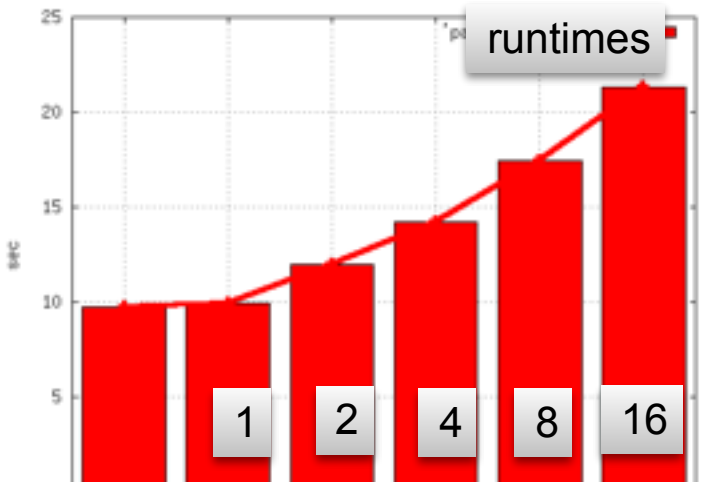
First Results

Example: Densifying Graph, Clustering Coefficient

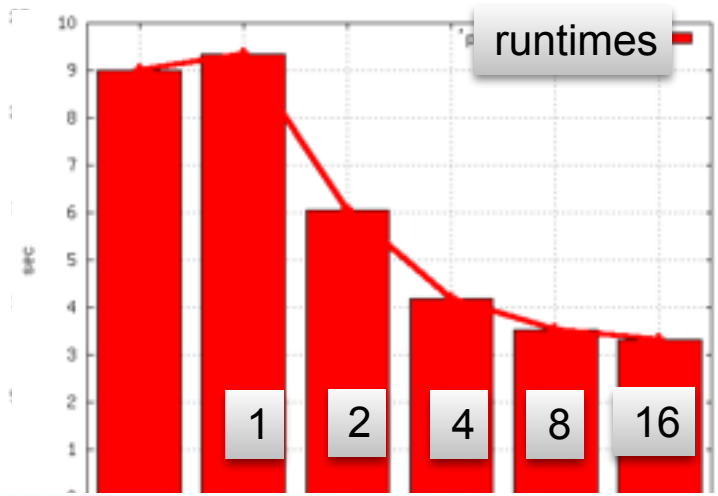
Non-Overlapping

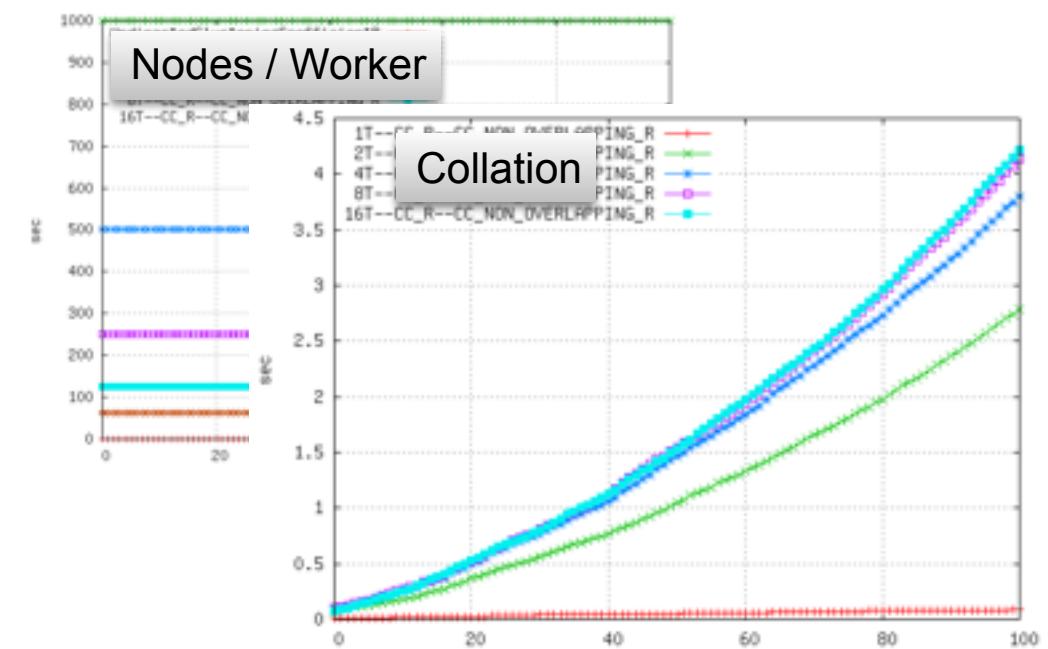


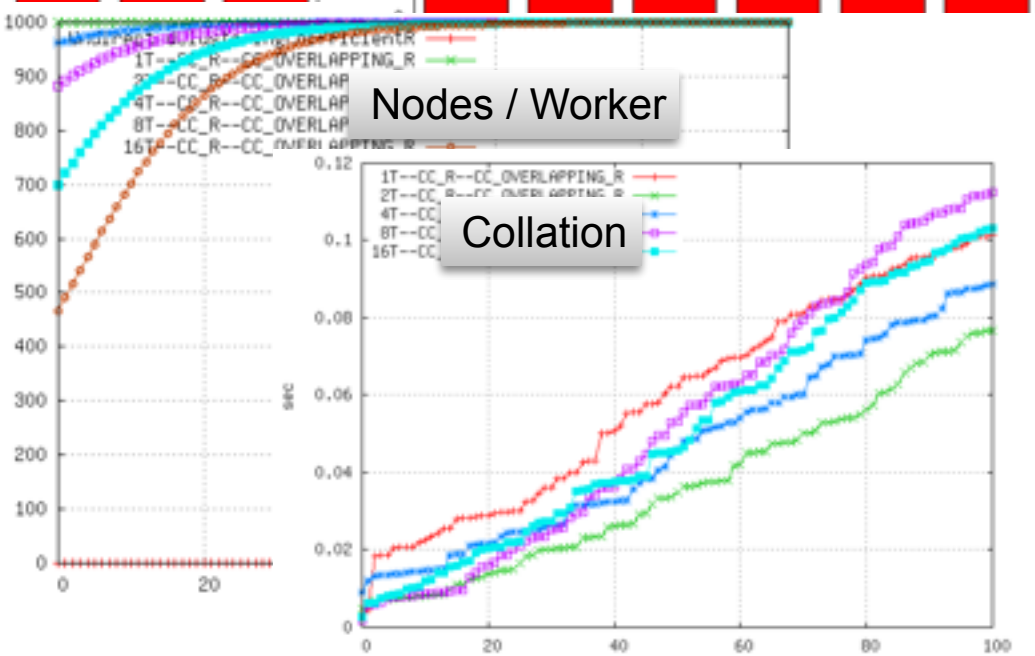
Overlapping



Overlapping (Alg*)







General Questions

- Approach
  - Partitioning Type?
  - Partitioning Scheme?
  - Node Assignment?
  - Re-Partitioning?
- Related Work
  - Is there a niche for this approach?
- Evaluation
  - How to “simulate”?
  - Which metrics to use?
  - Which datasets to use?