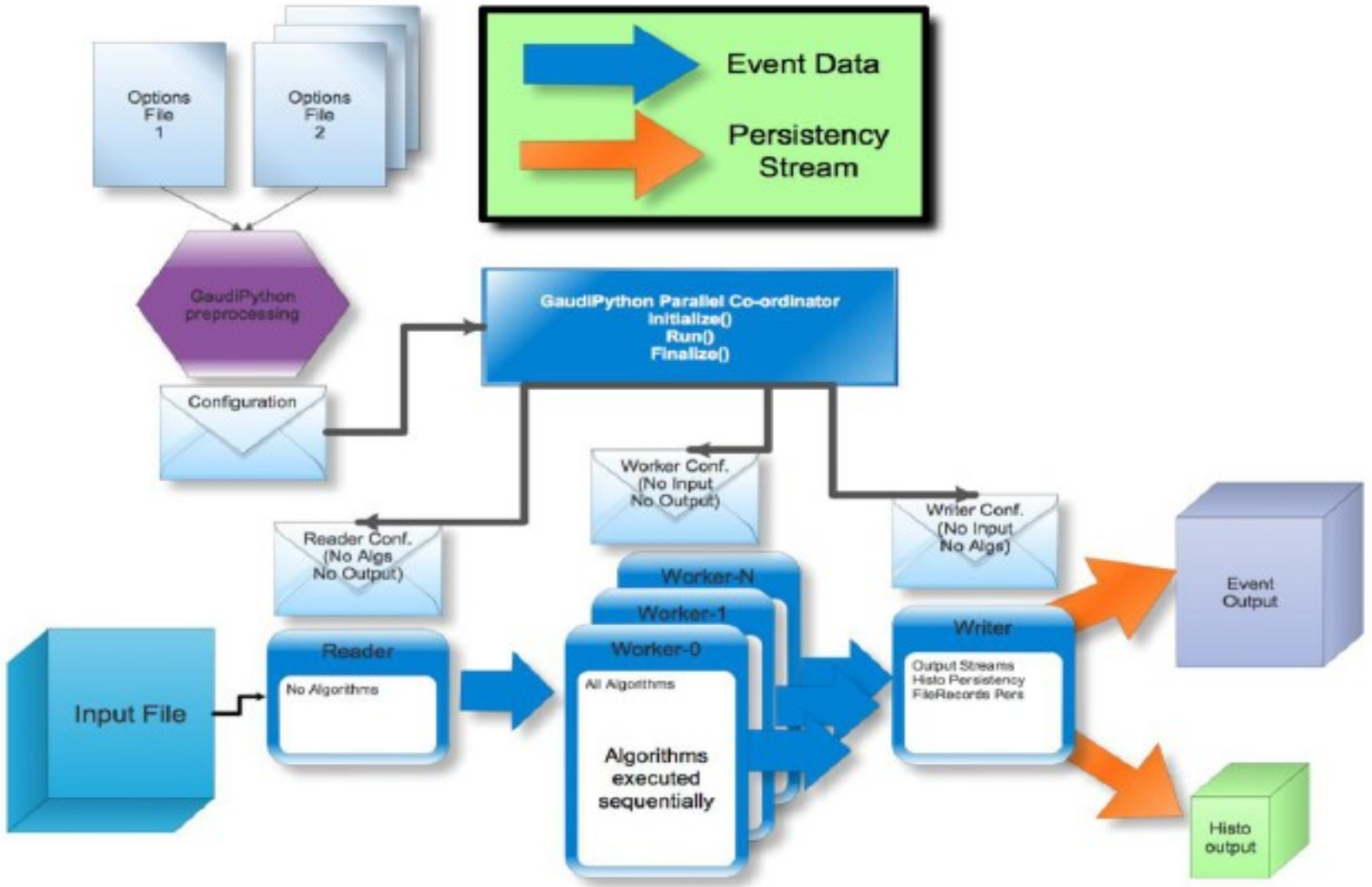
GaudiMP – performance- and KSM-measurements

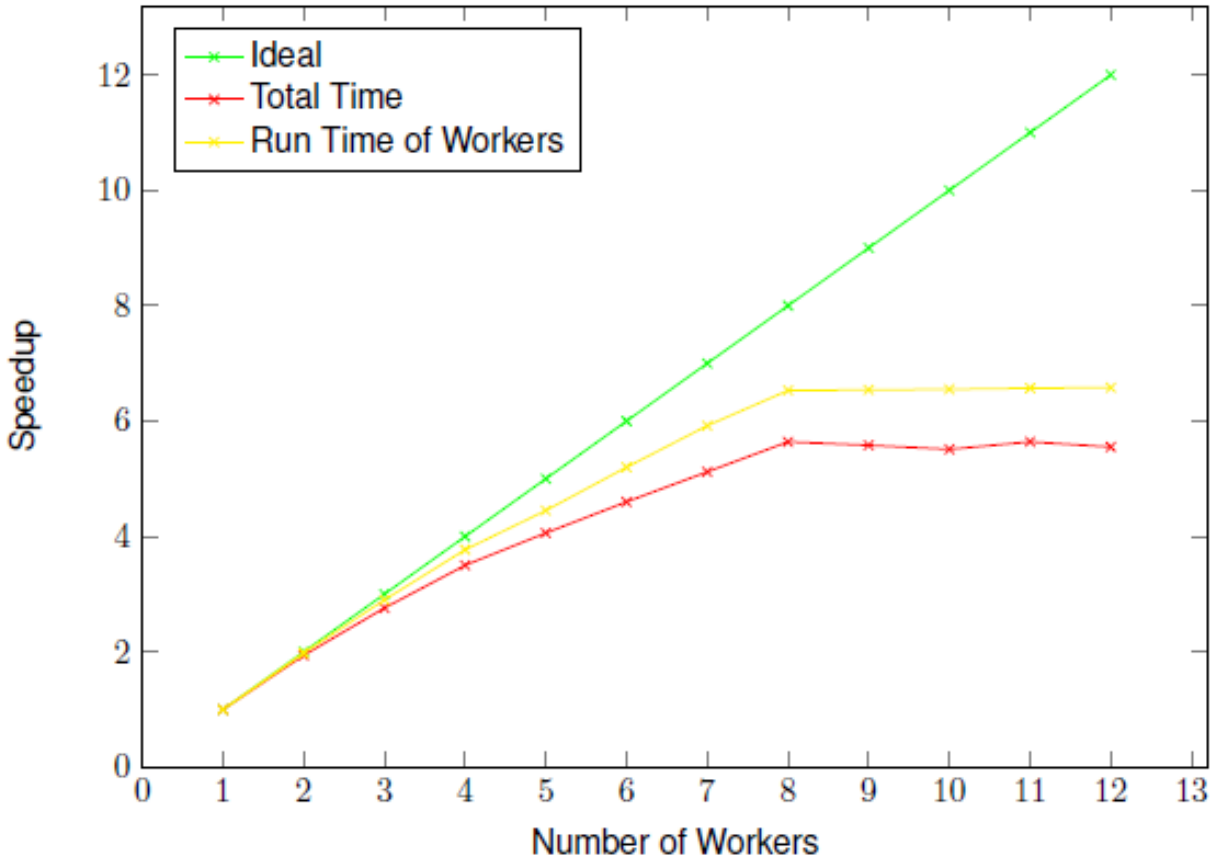
Nathalie Rauschmayr

Overview


Options File 1
Options File 2
Event Data
Persistency Stream
GaudiPython preprocessing
GaudiPython Parallel Co-ordinator
Initialize()
Run()
Finalize()
Configuration
Worker Conf.
(No Input
No Output)
Writer Conf.
(No Input
No Algs)
Reader Conf.
(No Algs
No Output)
Worker-N
Worker-1
Worker-0
Writer
Event Output
Input File
Reader
No Algorithms
All Algorithms
Algorithms executed sequentially
Output Streams
Histo Persistency
FileRecords Pers
Histo output

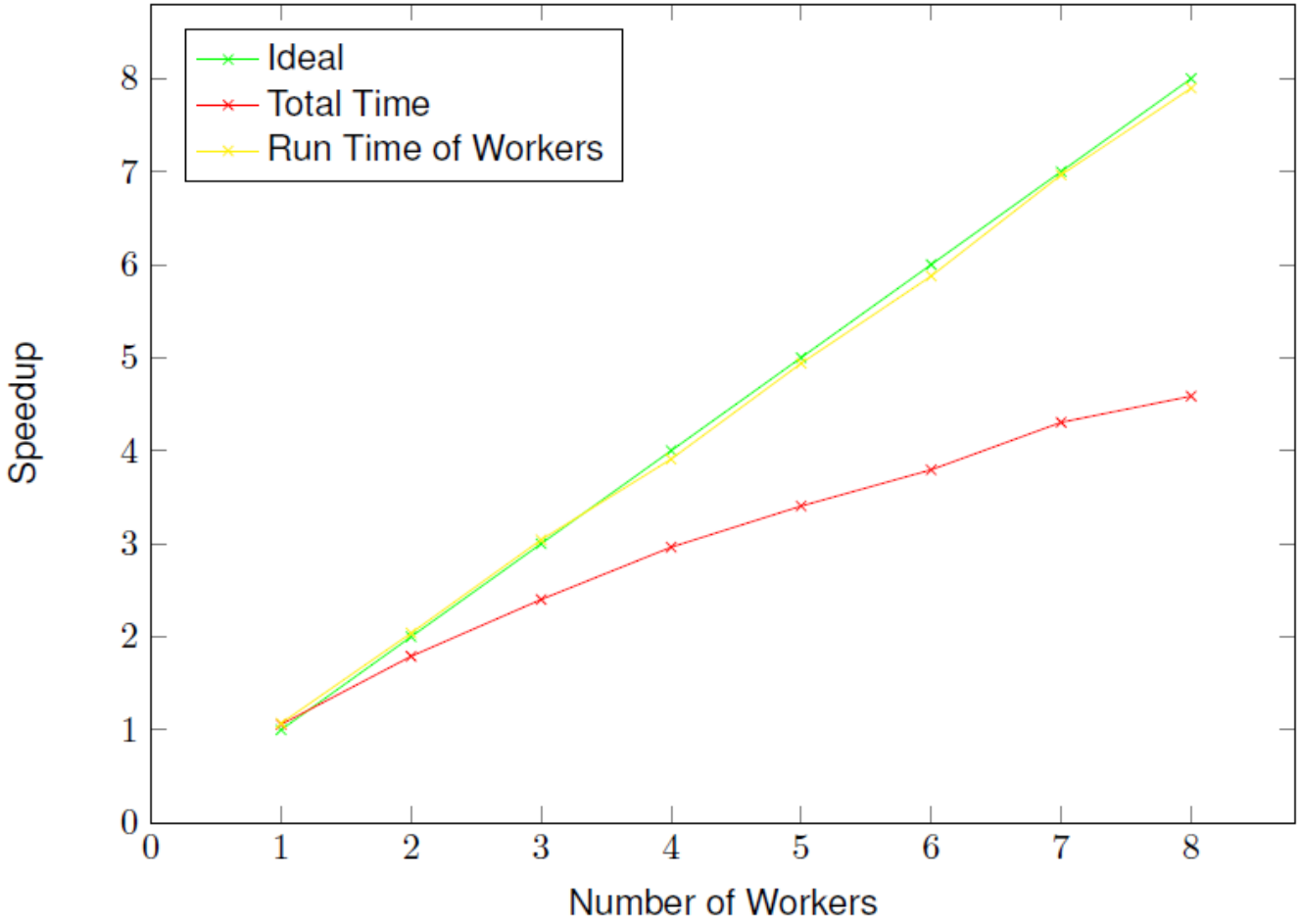
Speedup

- Reconstruction of 10000 Events



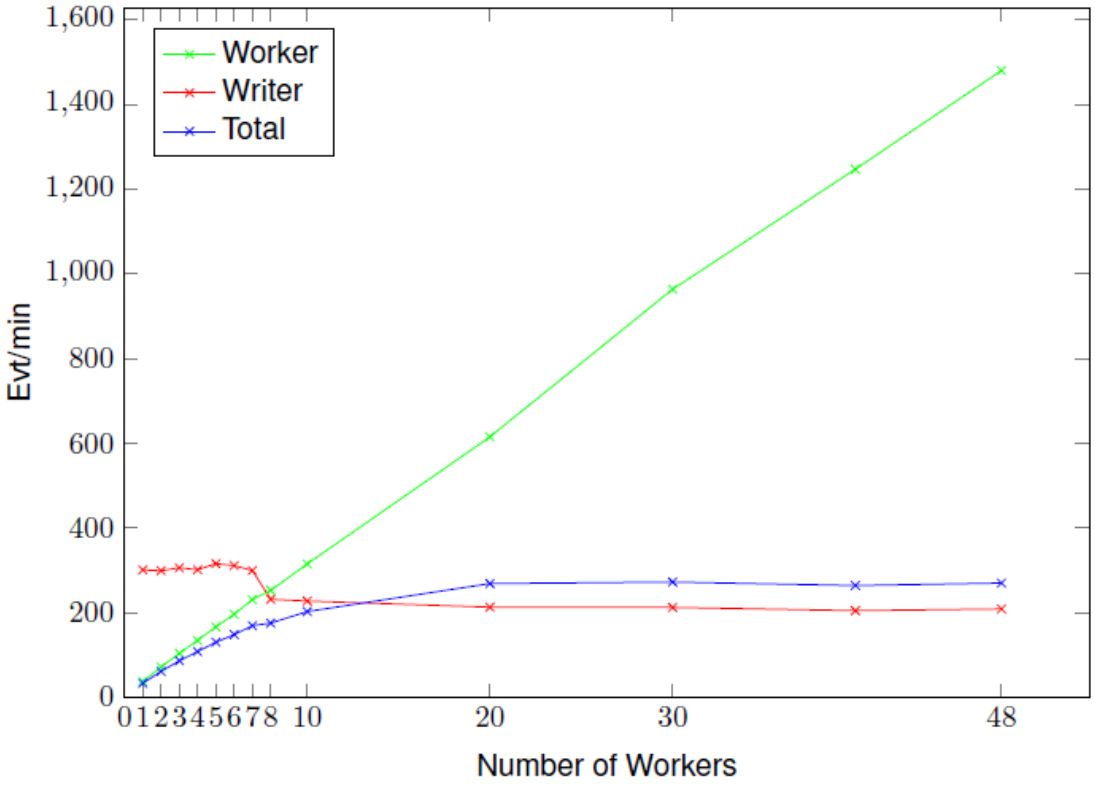
Speedup

- Simulation of 100 Events



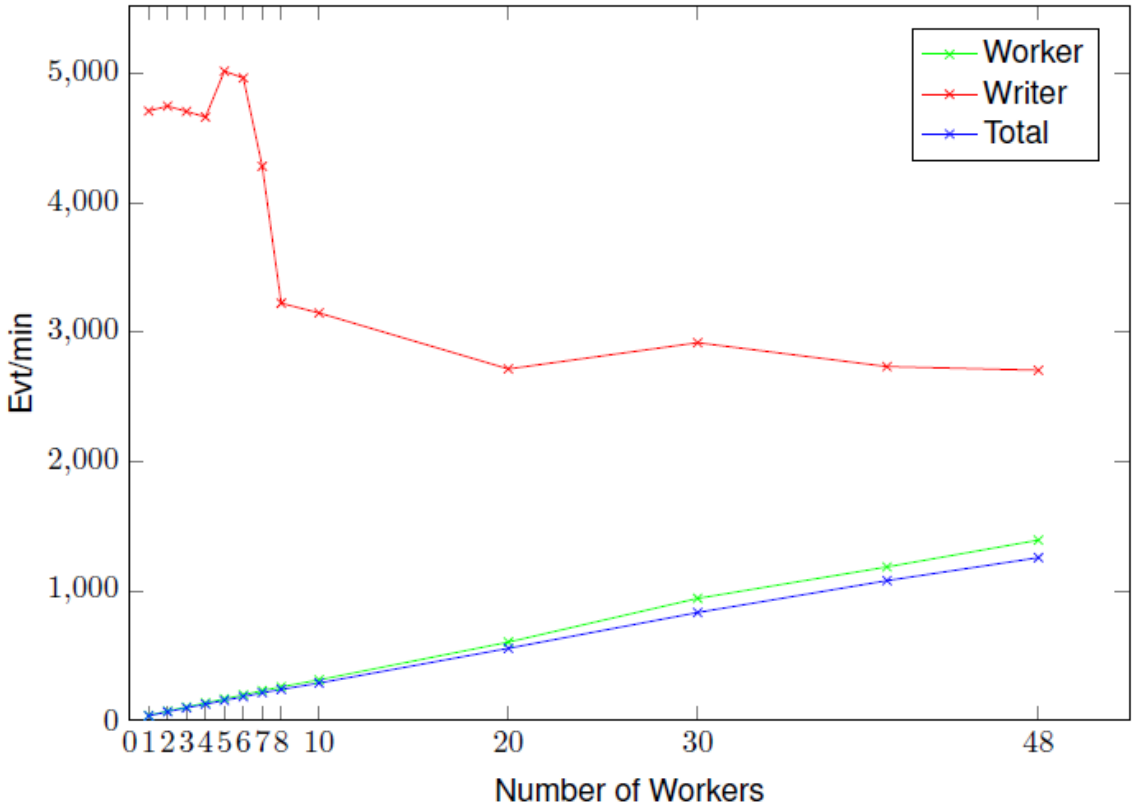
Limitations

- Problematic: when total event-throughput of workers reach the same value like writer
  - ~ factor 10



Limitations

- Change Root-compression
  - Writer throughput can be increased by factor 10

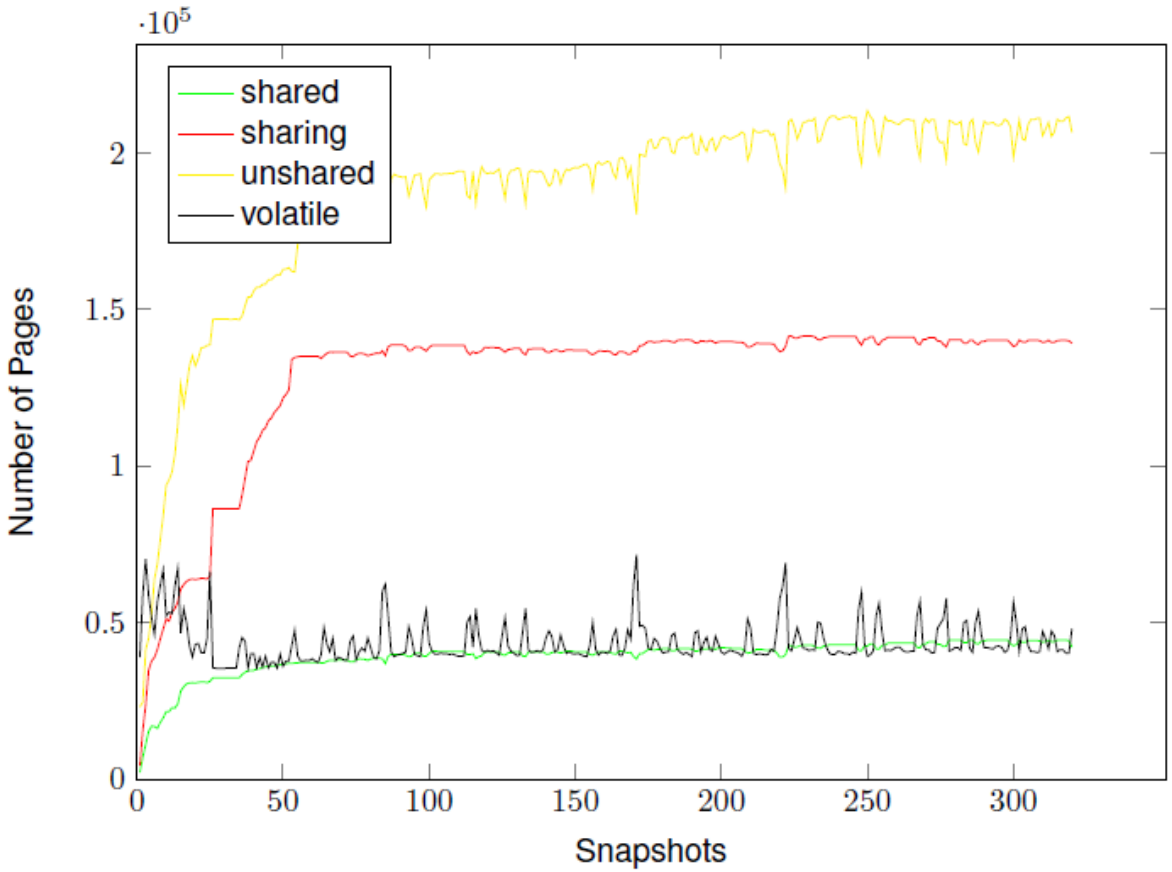


KSM-results

- *madvise*-call inside malloc-hook
- Monitoring of KSM-parameters
  - Pages shared
  - Pages sharing
  - Pages unshared
  - Pages volatile

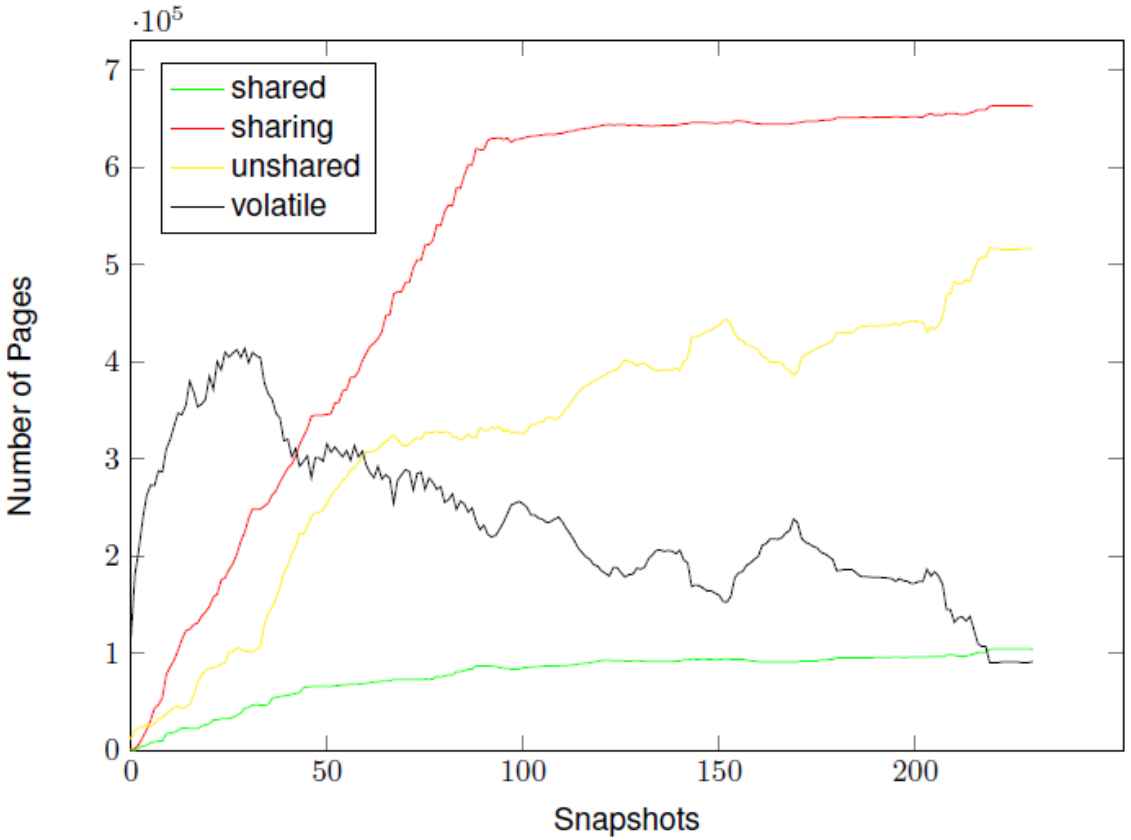
KSM-results

- 2 Workers, Reconstruction 1000



KSM-results

- Pages_volatile increases with the number of cores

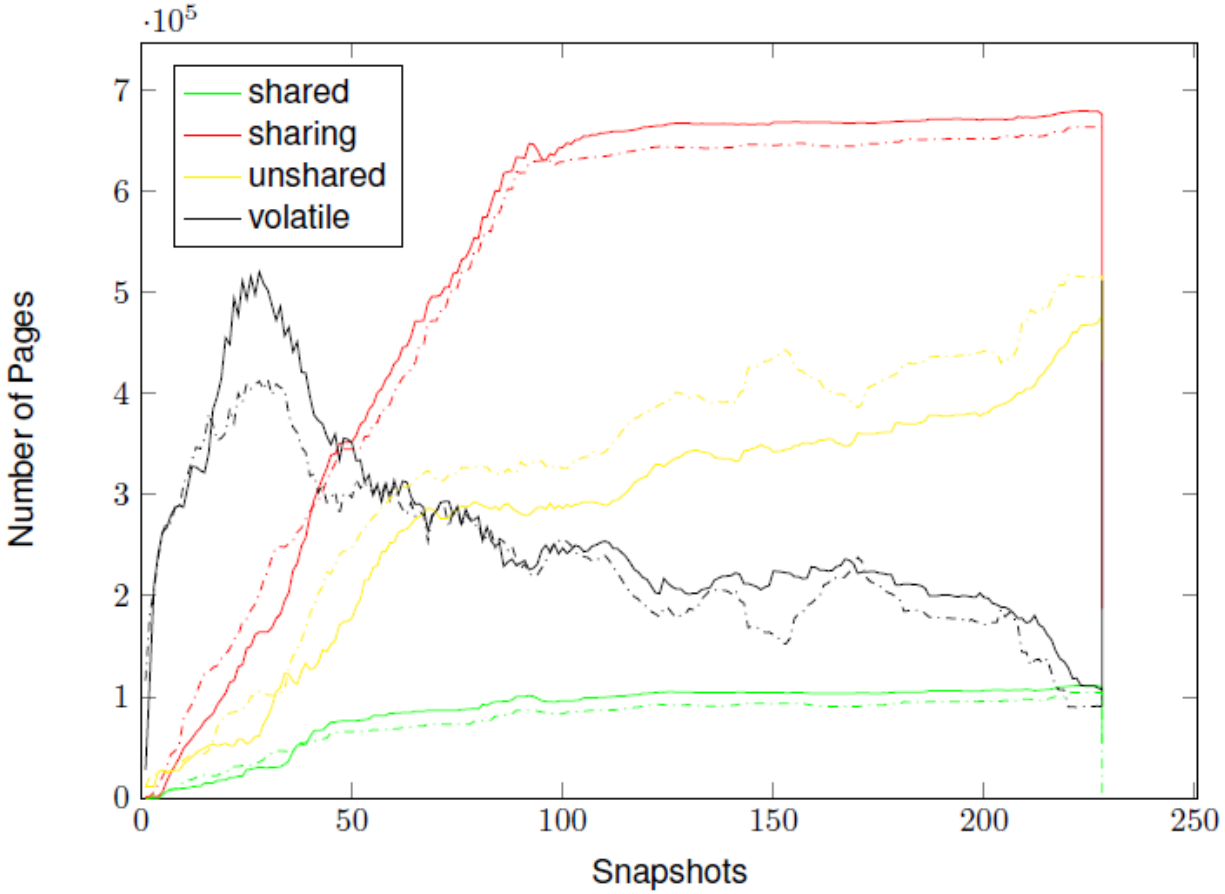


KSM-results

- Merging rate defined by:
  - Pages_to_scan
  - Time_to_sleep
- Modifying merging rate – example:
  - 8-core machine
  - worst case: analysis job
  - 40 MB/s * 8 processes
    - → 1640 Pages
    - → 20 ms
  - Decreasing CPU-consumption of KSM-thread

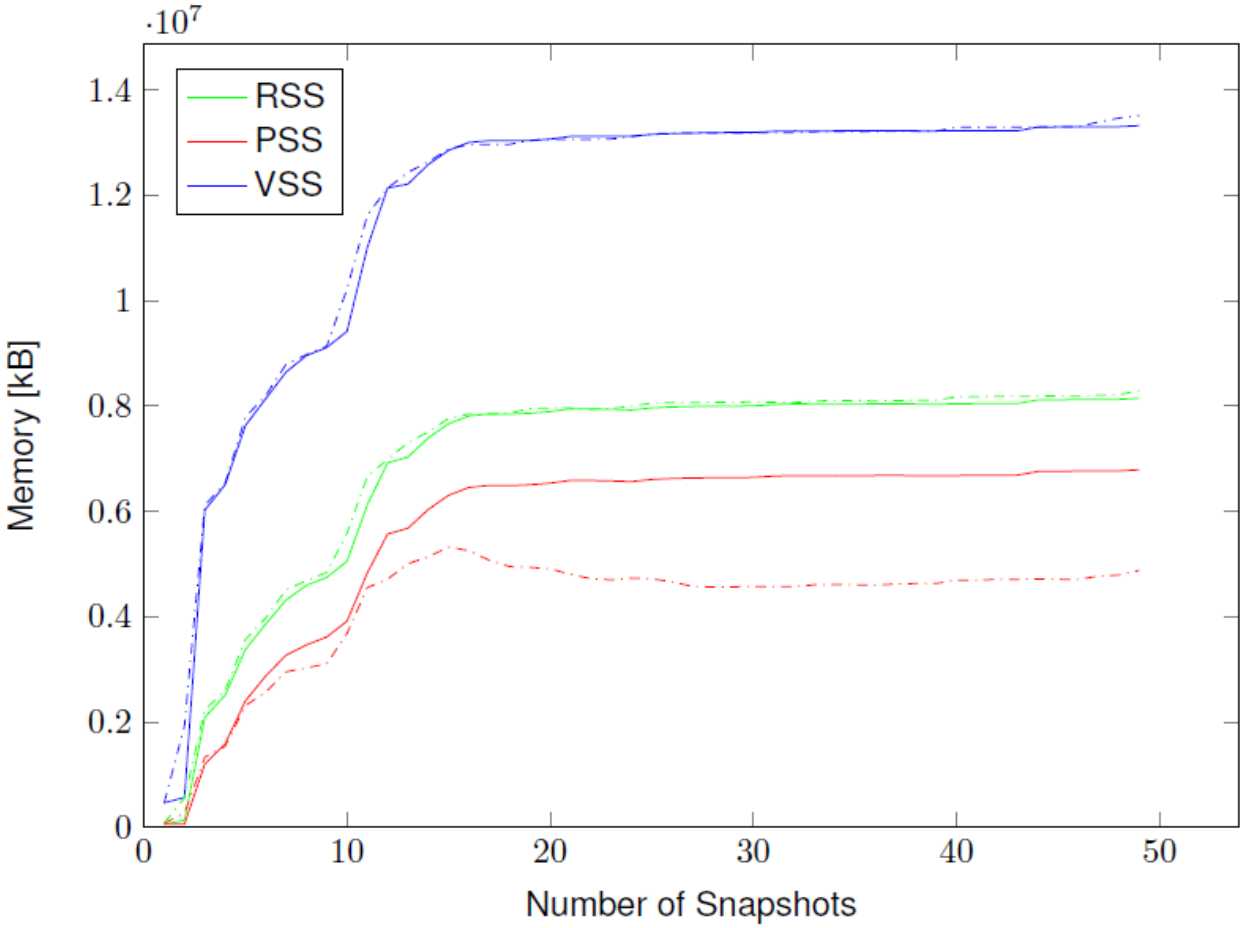
KSM-results

- Merging rate:
  - 190 GB/s versus 585 MB/s



KSM-results

- 8 Workers, Brunel Reconstruction 1000 Events



KSM-results

	serial mode	2 workers	4 workers	8 workers
Gauss	183 MB (22 %)	623 MB (33 %)	1275 MB (42 %)	2659 MB (48 %)
DaVinci	190 MB (10 %)	600 MB (17 %)	1577 MB (24 %)	3315 MB (27 %)
Brunel	94 MB (10 %)	465 MB (23%)	1112 MB (32 %)	1900 MB (31 %)

Caveats

- Merging rate must be adpated otherwise high CPU consumption by KSM-thread
- KSM does not work on the level of virtual memory
- *pages_volatile* becomes likely a bottleneck
- *madvise*-call inside application

Conclusion

- Without KSM: nearly no memory reduction
- GaudiMP scales well:
  - But: Optimization for the writer process necessary
- Future plans:
  - Find a solution for the writer process
  - Evaluation: is KSM a good replacement for late forking
  - Further memory optimzation: compression with compcache and zram