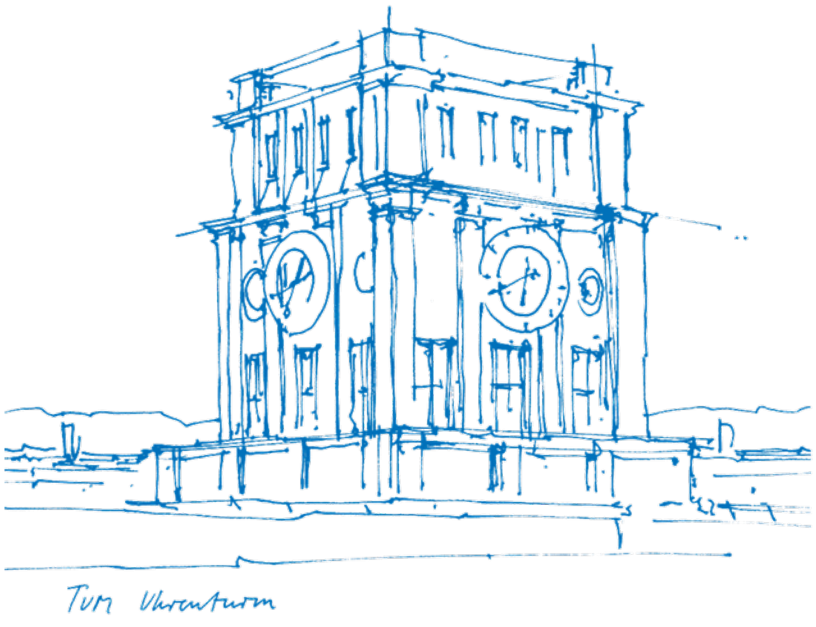
Implementing an Efficient Shuffle Operator for Streaming Database Systems

Bachelor Thesis

Author: Jonas Ladner
Supervisor: Maximilian Rieger, M.Sc.
Examiner: Prof. Dr. Thomas Neumann



Garching, 11.03.2025

Technical University of Munich

Problem Setting

Streaming Shuffle Simulation:

1. **Tuple Generation:** Randomly generated tuples with 32-bit keys and optional data fields.
2. **Data Shuffle:** Tuples stored in partition buckets using slotted pages.
3. **Storing on Slotted Pages:** Thread-local vs. shared (locking/lock-free) write-out strategies.

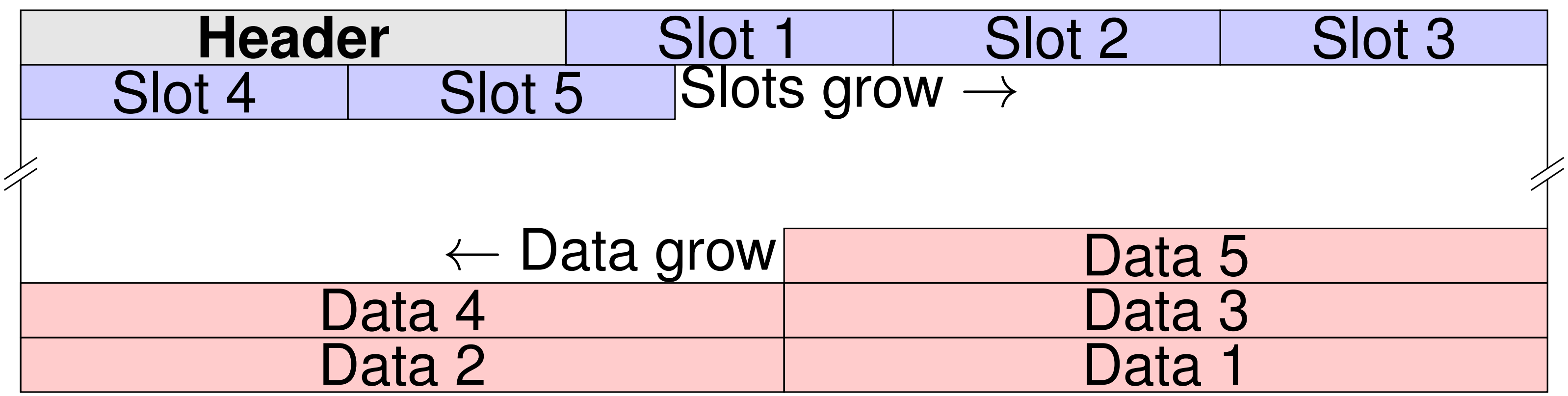


Figure: Slotted Page grow visualization

Key Contribution: Efficient, multithreaded shuffle operator implementations.

Naive approach: *OnDemand*

OnDemand:

- Tuples are directly written to the partition buckets.

Problems:

- Each tuples causes a write-out to a shared slotted page.
- Very high contention on partition buckets.

Naive approach: *OnDemand*

OnDemand:

- Tuples are directly written to the partition buckets.

Problems:

- Each tuples causes a write-out to a shared slotted page.
- Very high contention on partition buckets.

Optimized approach: *Smb*

Software Managed Buffers (SMBs):

- Cacheline-sized, thread-local buffers for each partition.
- Flush partition when buffer is full.

Problems:

- High contention on partition buckets.

Histogram-based approach: *Radix*

Materialize large chunk of incoming tuples:

- Assign each thread a memory region within this chunk.
- Memory region can be reused for next chunk.

Local Histogram:

- Create thread-local histogram (frequency map) for each partition.

Global Histogram:

- Merge local histograms into a global histogram.
- Pre-allocate slotted pages based on global histogram.
- Assign exclusive write-out locations based on local histograms.

Problems:

- Synchronization overhead during histogram merging and page allocation.

Histogram-based approach: *Hybrid*

Materialize small chunk of incoming tuples:

- Assign each thread a memory region within this chunk.
- Memory region can be reused for next chunk.

Local Histogram:

- Create thread-local histogram (frequency map) for each partition.

Ad-hoc page allocation:

- Assign exclusive write-out locations based on local histograms.
- Allocate slotted page for local histogram if not already allocated.

Problems:

- Synchronization overhead during page allocation.

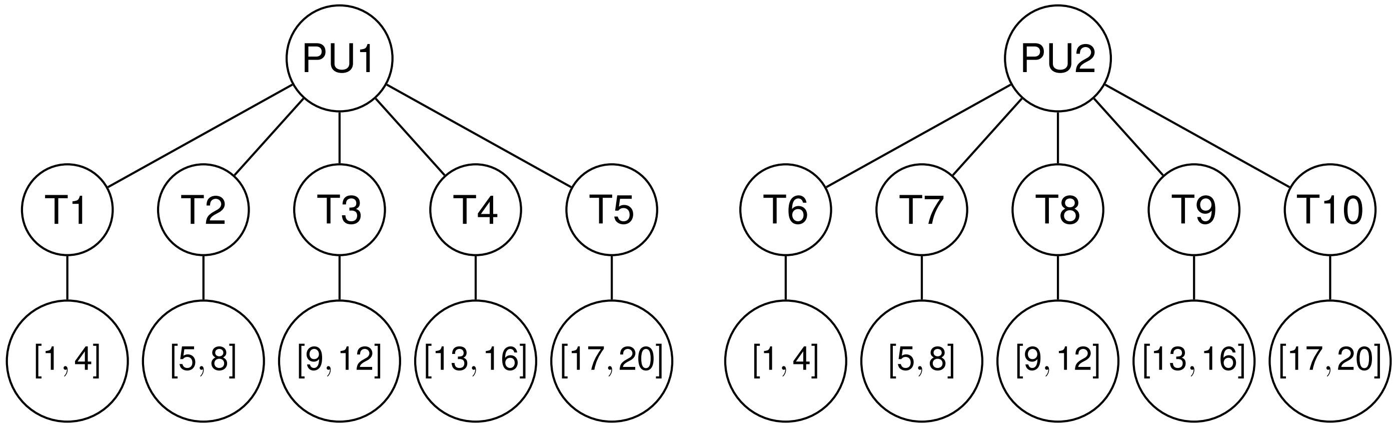
Reducing contention: *CmpProcessingUnits*

Processing Units:

- Partition threads into *Processing Units*
- Within each Processing Unit:
  - Each thread is assigned an exclusive partition range
  - No overlap between partition ranges
  - Only a single thread writes to a given partition

Problems:

- Each tuple must be processed by all threads of a Processing Unit



Avoiding contention: *LocalPagesAndMerge*

Thread-local Pages:

- Each thread has its own slotted pages
- Avoids synchronization

Merging Phase:

- All non-full pages have to be merged
- We assign each thread a group of partitions to merge
- Each thread merges the pages of its assigned partitions without synchronization

Problems:

- Huge initial memory consumption

Evaluation – 16 Byte Tuple and 2/8 Partitions

Intel i9-7900X, 1x20 threads, 1 NUMA node, 128 GB memory

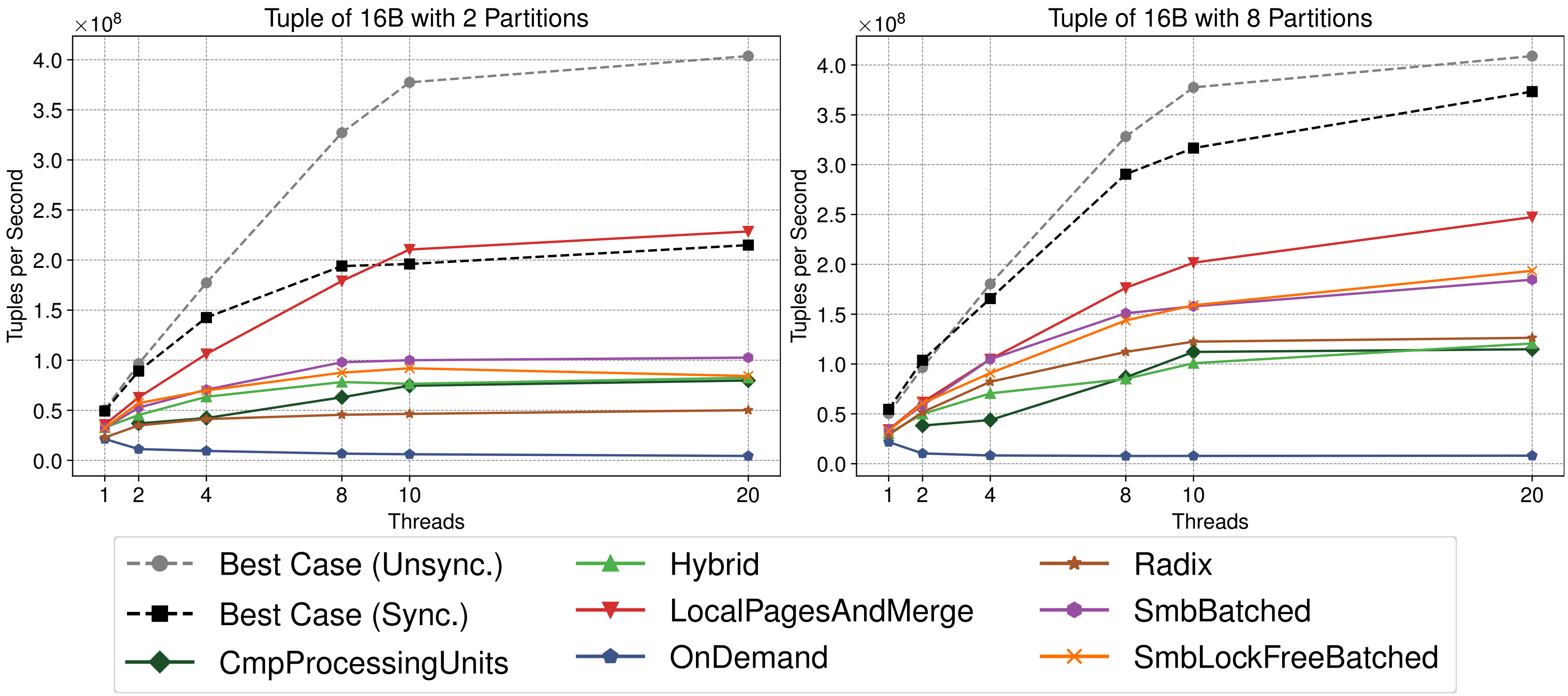


Figure: Benchmark Plots for Tuple of 16B with 2 and 8 Partitions

Evaluation – 16 Byte Tuple and 32/1024 Partitions

Intel i9-7900X, 1x20 threads, 1 NUMA node, 128 GB memory

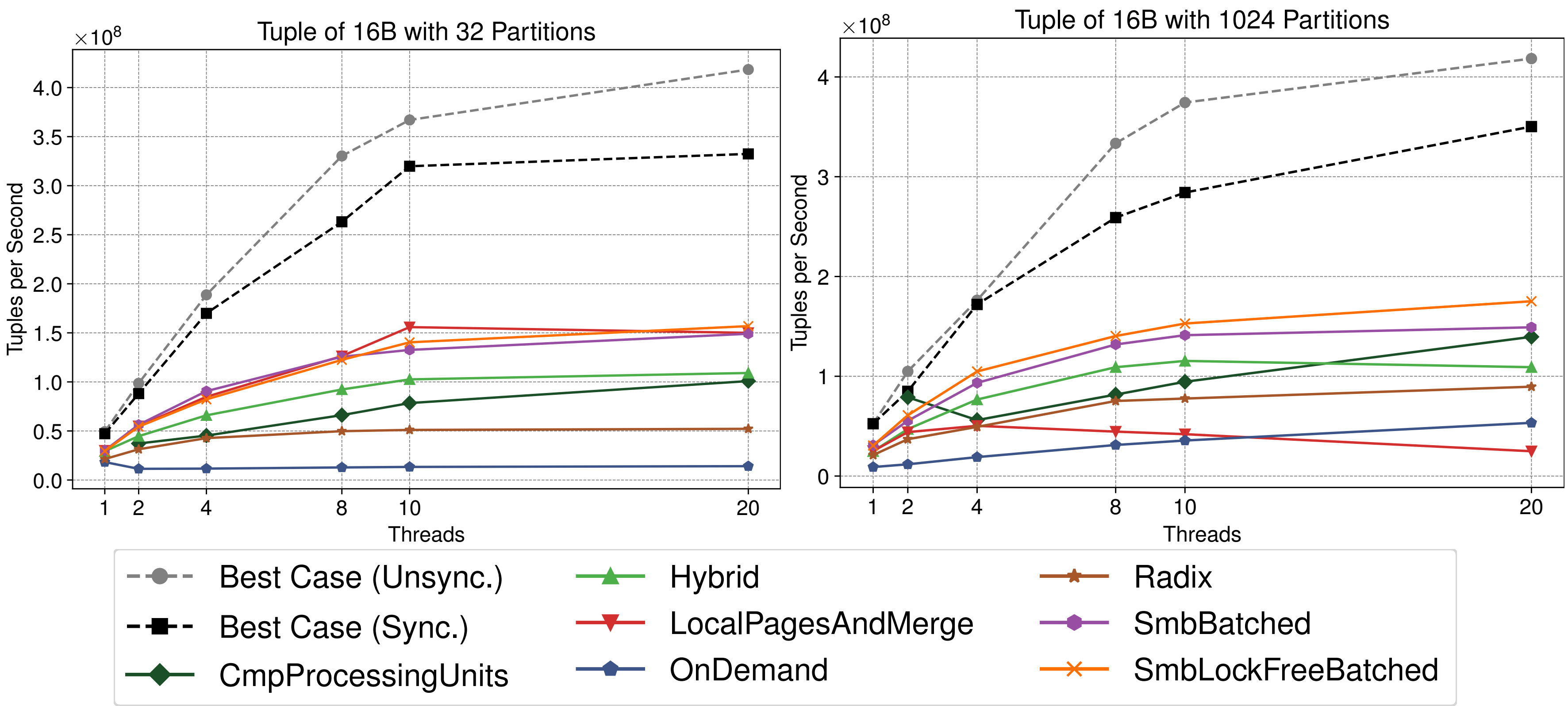


Figure: Benchmark Plots for Tuple of 16B with 32 and 1024 Partitions

Peak Heap Memory

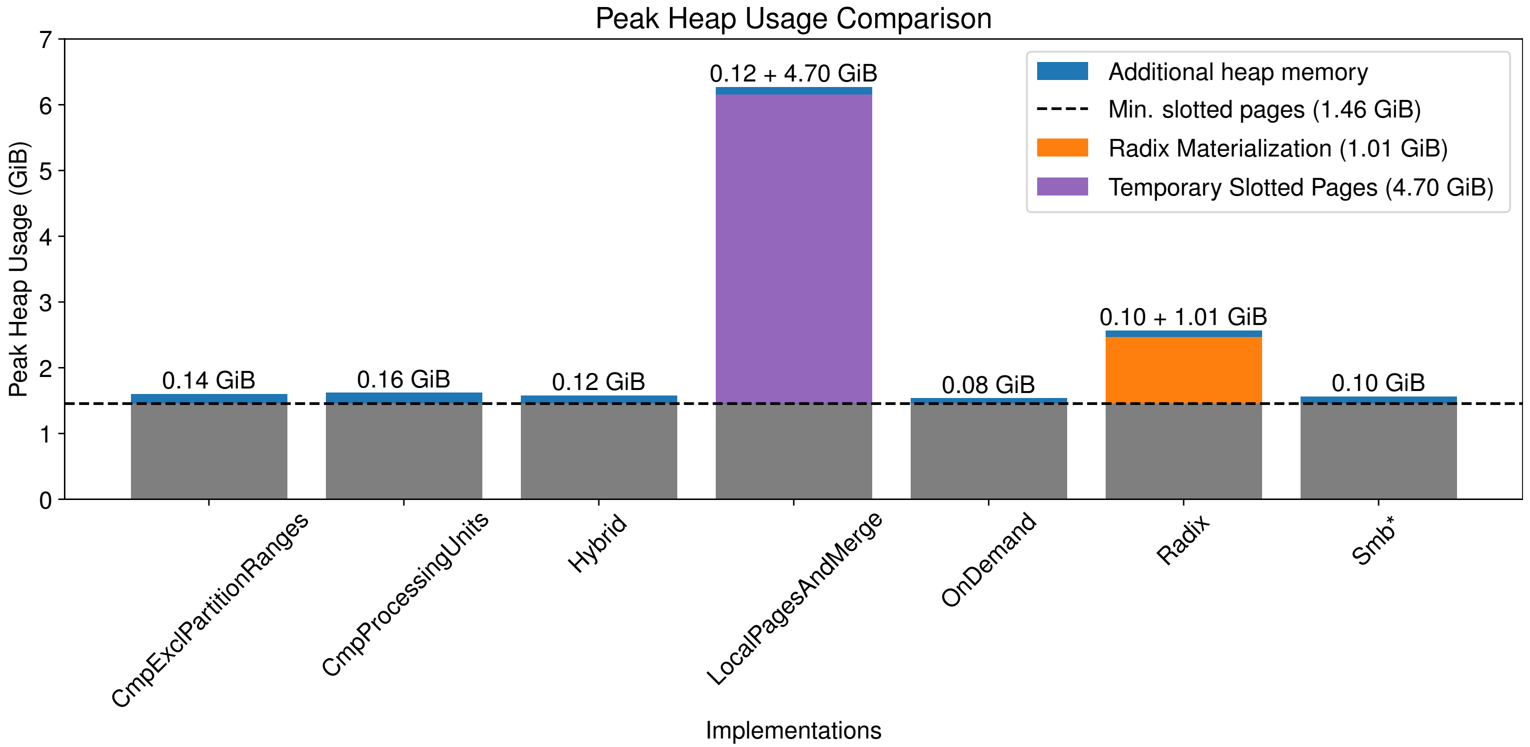


Figure: Peak Heap Usage when using 32 Partitions, 40 Threads and 67.2 Mio. 16B Tuples (1 GiB)

Comparison with Apache Flink

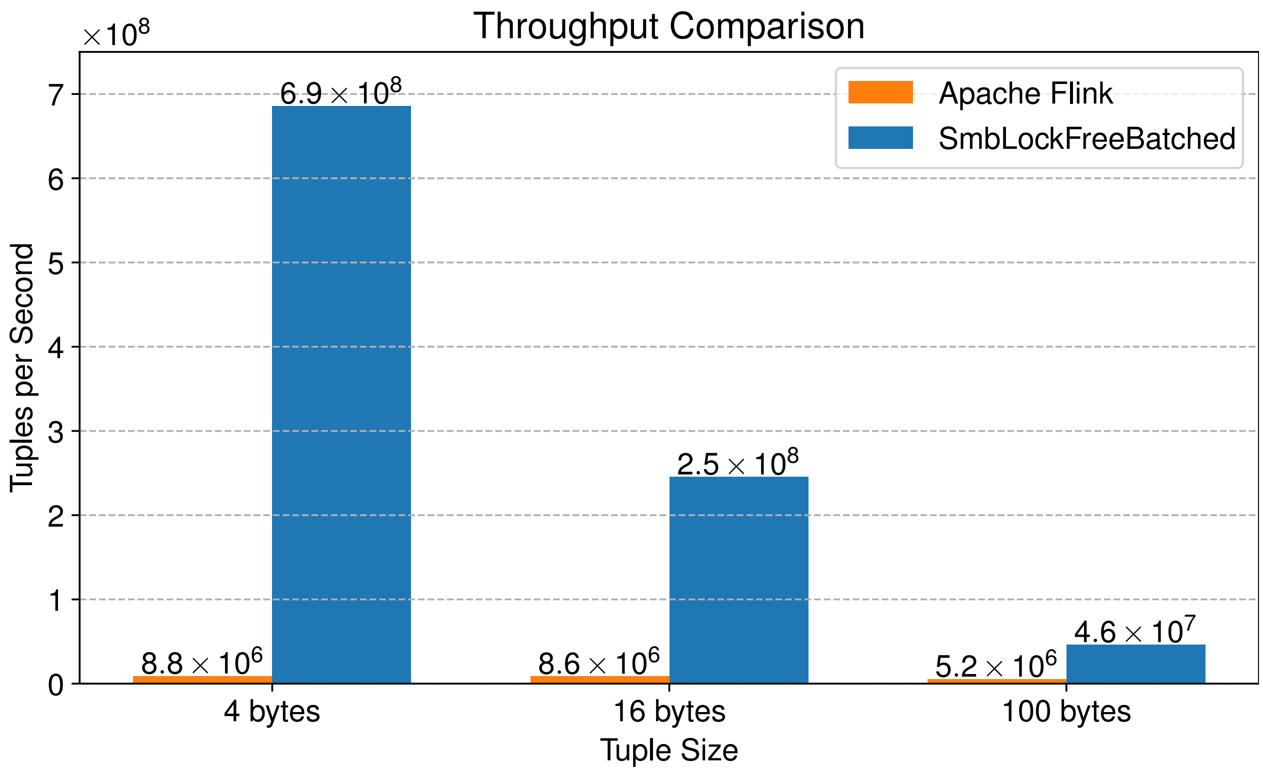


Figure: Tuples per Second Comparison when using 1024 Partitions

Evaluation – 16 Byte Tuple and 4/32 Partitions

AMD EPYC 7713, 2x64 threads, 2 NUMA nodes, 1 TB memory

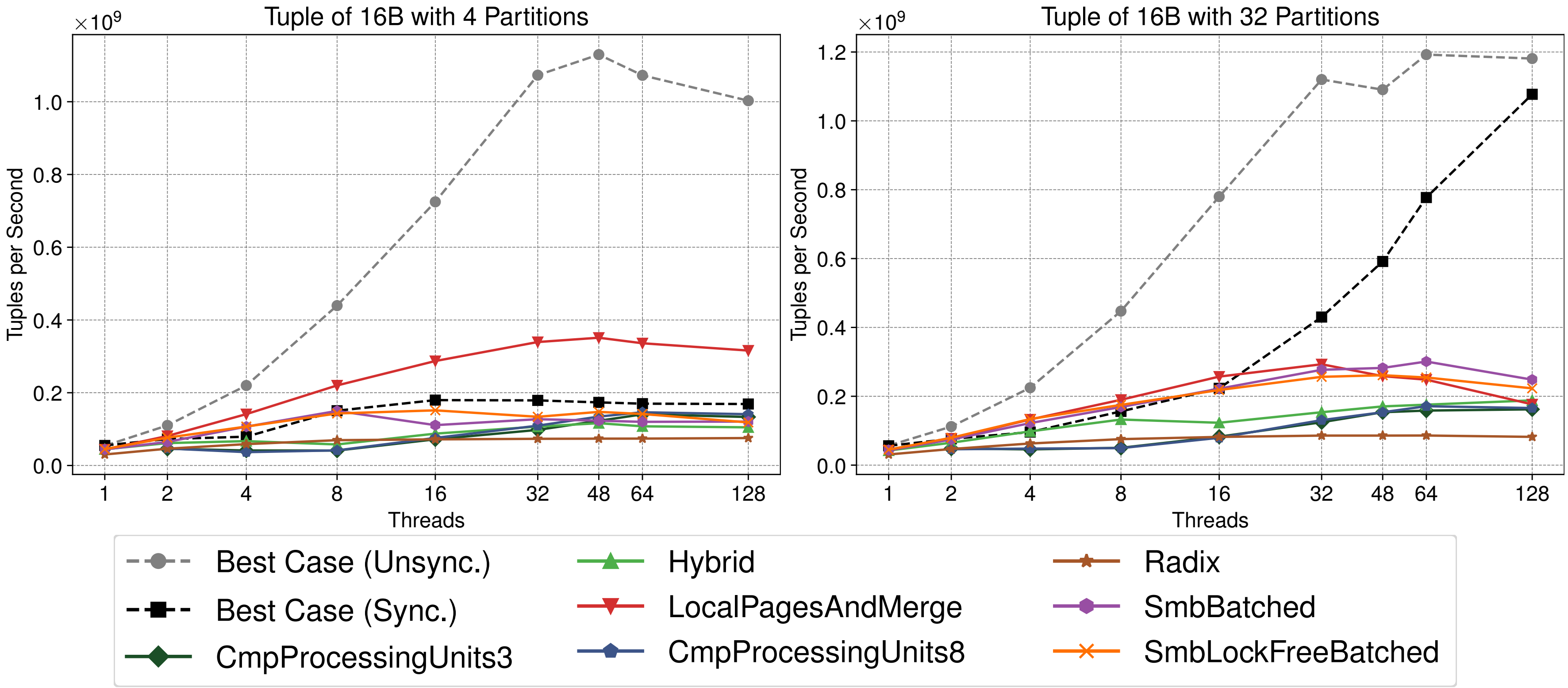


Figure: Benchmark Plots for Tuple of 16B with 4 and 32 Partitions

Conclusion

- Thread-local slotted pages are optimal for low partition counts (<32).
- SMB-based methods scale best for high partition counts (>32).

Future Work

- Further reducing contention on machines with 20+ cores.
- Evaluate the implementations in a real-world streaming system.
- Investigate the impact of data skew.

Evaluation – 4 Byte Tuple and 32/1024 Partitions

Intel i9-7900X, 1x20 threads, 1 NUMA node, 128 GB memory

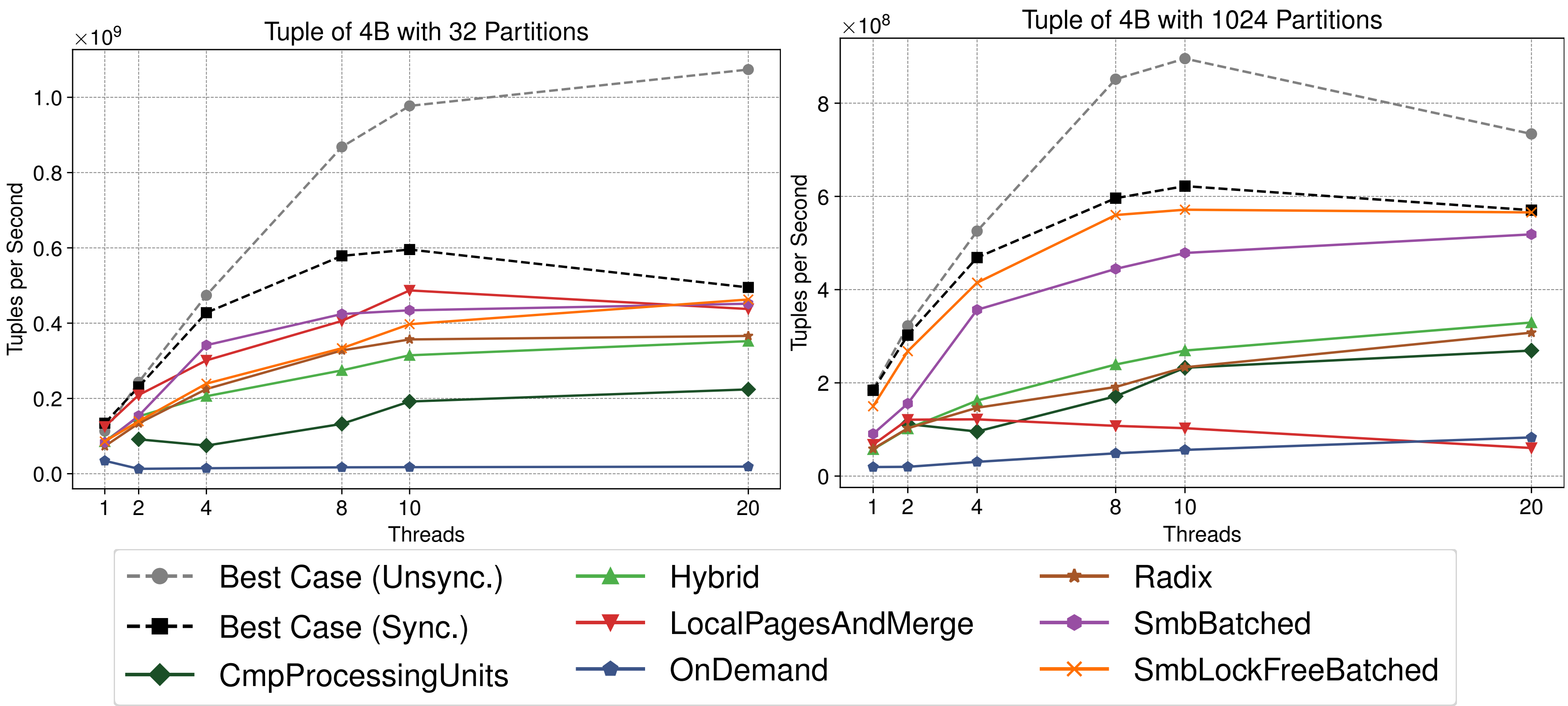


Figure: Benchmark Plots for Tuple of 4B with 32 and 1024 Partitions

Evaluation – 100 Byte Tuple and 32/1024 Partitions

Intel i9-7900X, 1x20 threads, 1 NUMA node, 128 GB memory

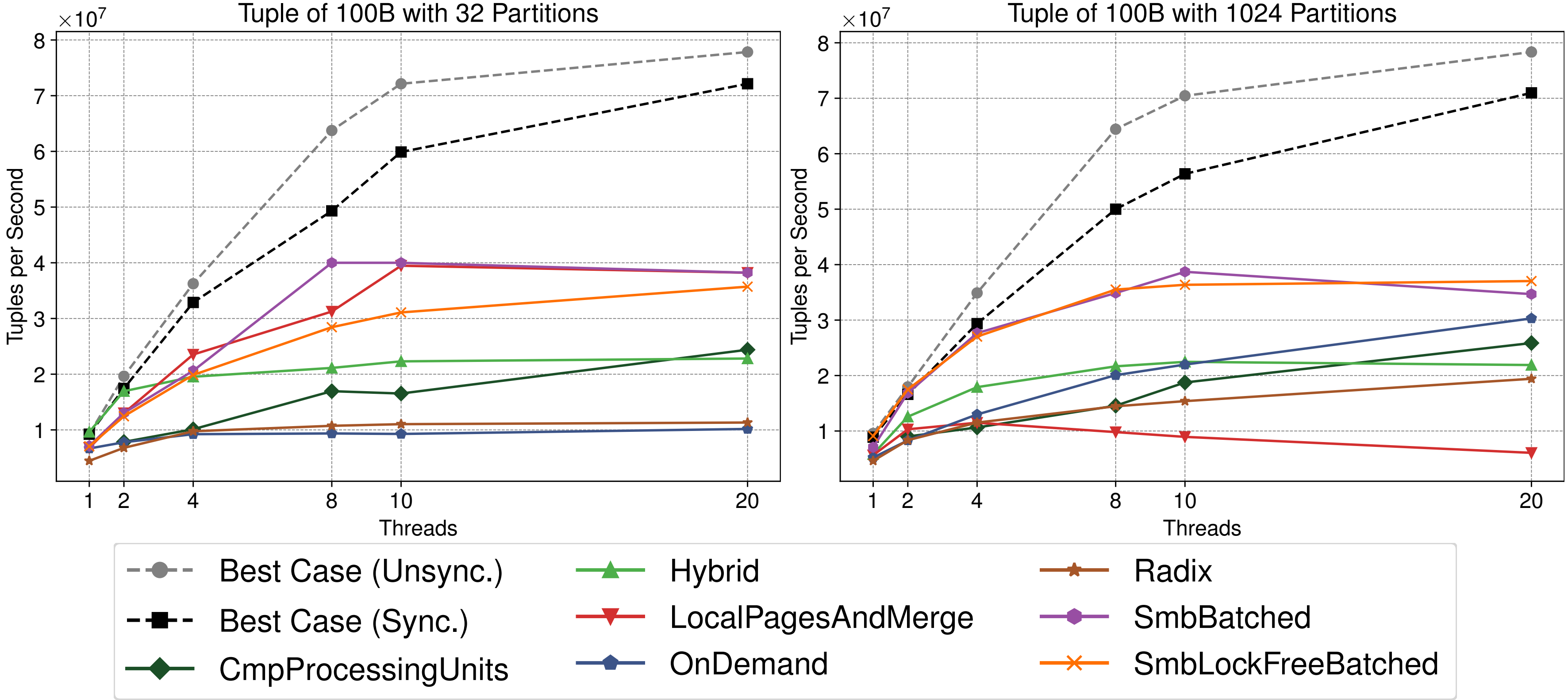


Figure: Benchmark Plots for Tuple of 100B with 32 and 1024 Partitions