Morsel-Driven Parallelism: A NUMA-Aware Query Evaluation Framework for the Many-Core Age

Viktor Leis, Peter Boncz*, Alfons Kemper, Thomas Neumann

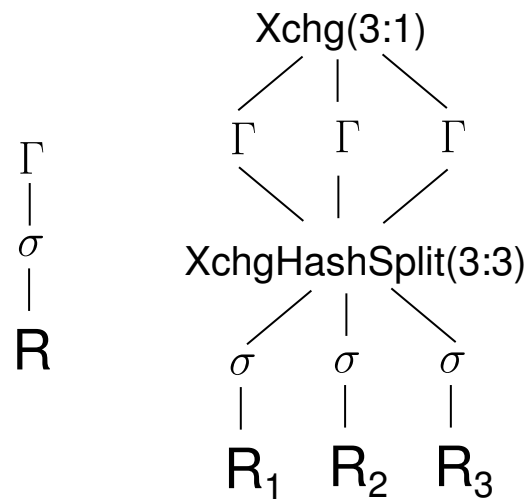
Technische Universität München *CWI

Introduction

- Number of CPU cores keeps growing:
  4-socket Ivy Bridge EX with 60 cores, 120 threads, 1TB RAM (50,000$)
- These systems support terabytes of NUMA RAM: disk is not a bottleneck
- For analytic workloads intra-query parallelization is necessary to utilize such systems
- We present an architectural blueprint for a query engine incorporating the following
  - Morsel-driven query execution (work is distributed between threads dynamically using work stealing)
  - Set of fast parallel algorithms for the most important relational operators
  - Systematic approach to integrating NUMA-awareness into database systems

Related Work: Volcano-Style Parallelism (1)

- *Encapsulation of Parallelism in the Volcano Query Processing System*, Goetz Graefe, SIGMOD 1990
  SIGMOD Test of Time Award 2000
- *Plan-driven* approach:
  - optimizer statically determines at query compile time how many threads should run
  - instantiates one query operator plan for each thread
  - connects these with exchange operators, which encapsulate parallelism and manage threads
- Elegant model which is used by many systems

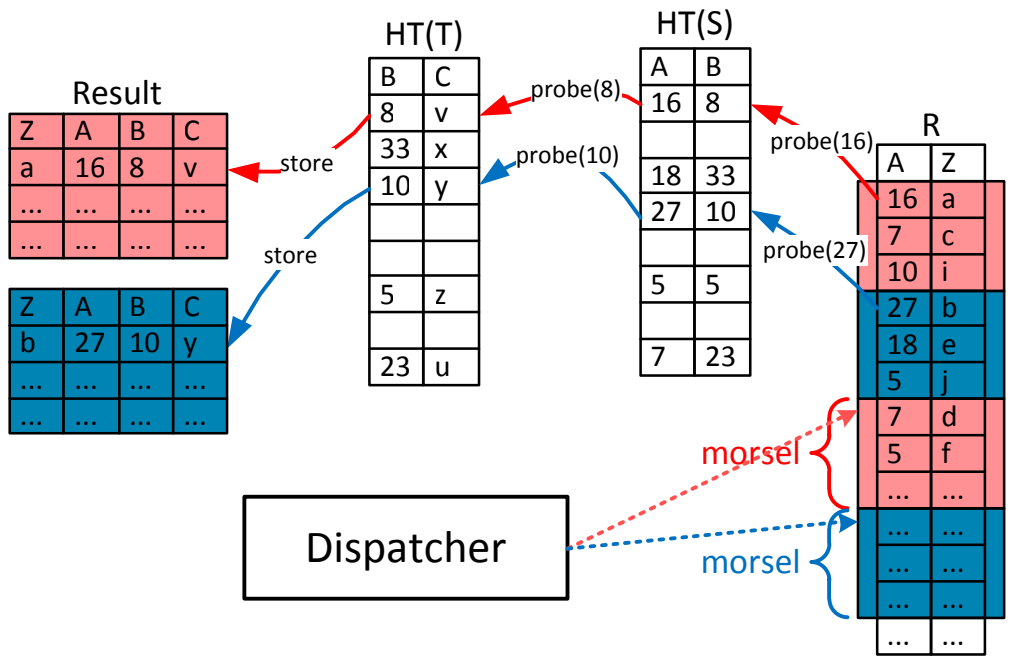


Volcano-Style Parallelism (2)

+ Operators are largely oblivious to parallelism
– Static work partitioning can cause load imbalances
– Degree of parallelism cannot easily be changed mid-query
– Not NUMA aware
– Overhead:
  - Thread oversubscription causes context switching
  - Hash re-partitioning often does not pay off
  - Exchange operators create additional copies of the tuples

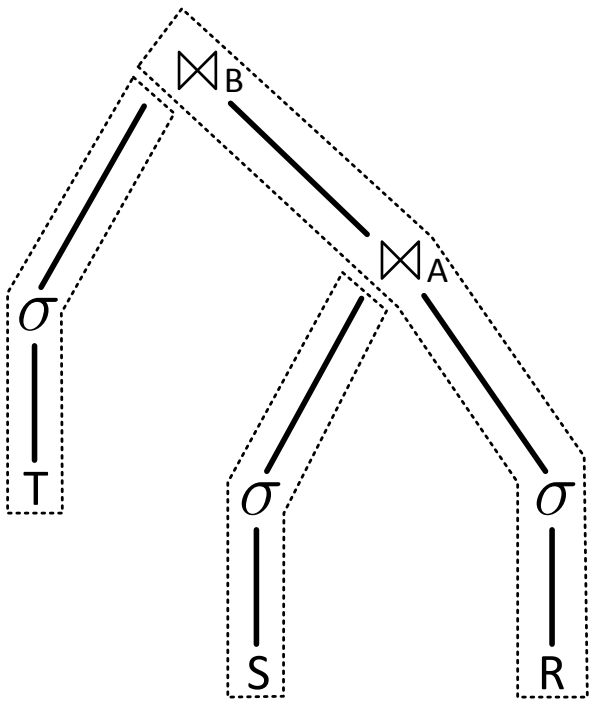
Morsel-Driven Query Execution (1)

- Break input into constant-sized work units ("morsels")
- Dispatcher assigns morsels to worker threads
- # worker threads = # hardware threads
- Operators are designed for parallel execution



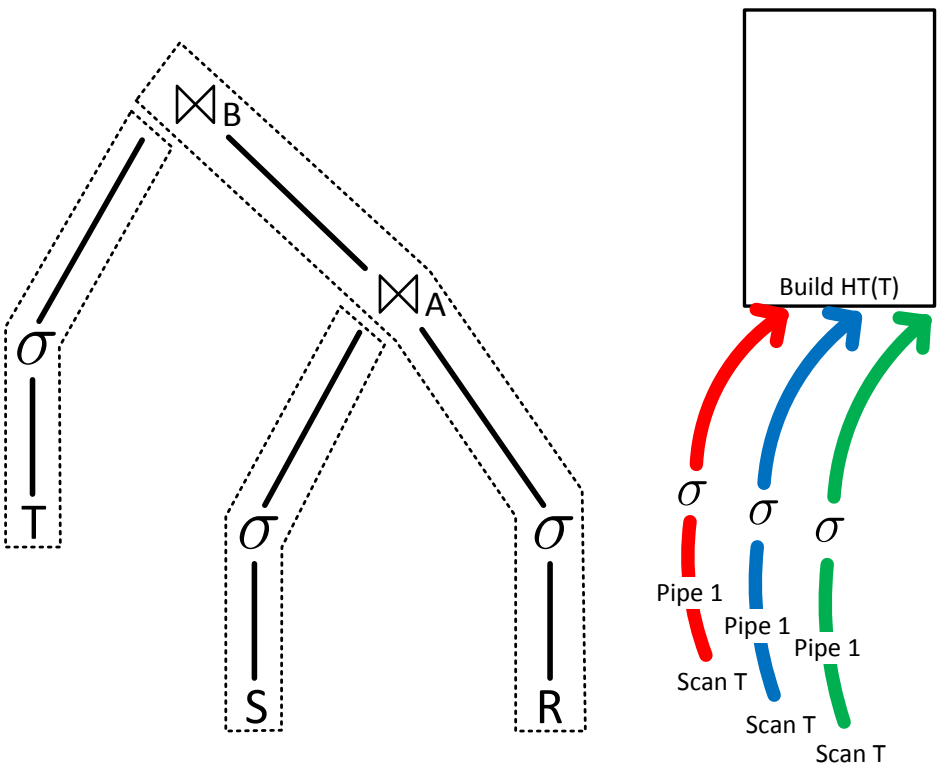
Morsel-Driven Query Execution (2)

- Each pipeline is parallelized individually using all threads



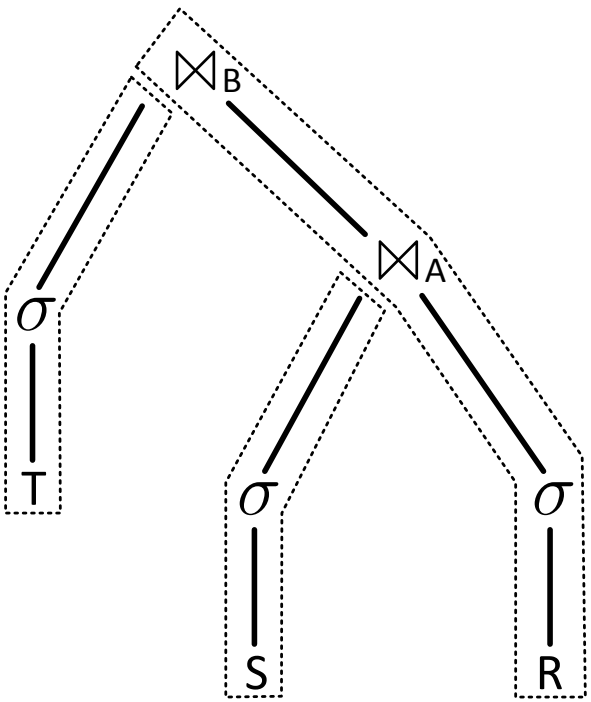
Morsel-Driven Query Execution (2)

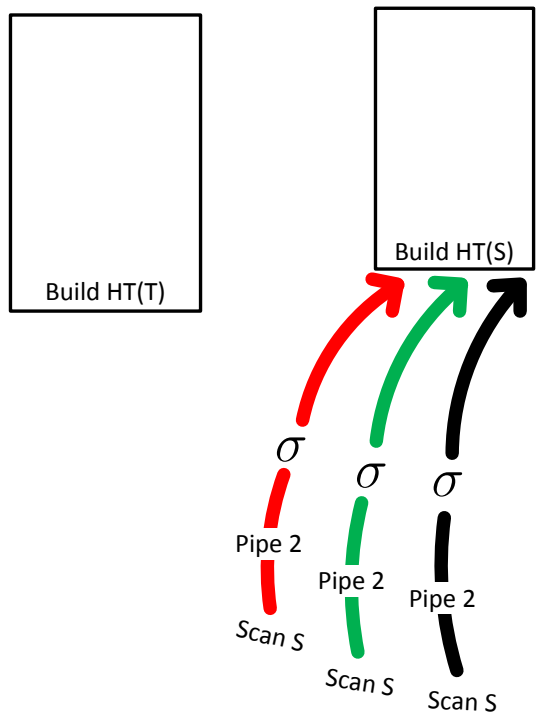
- Each pipeline is parallelized individually using all threads



Morsel-Driven Query Execution (2)

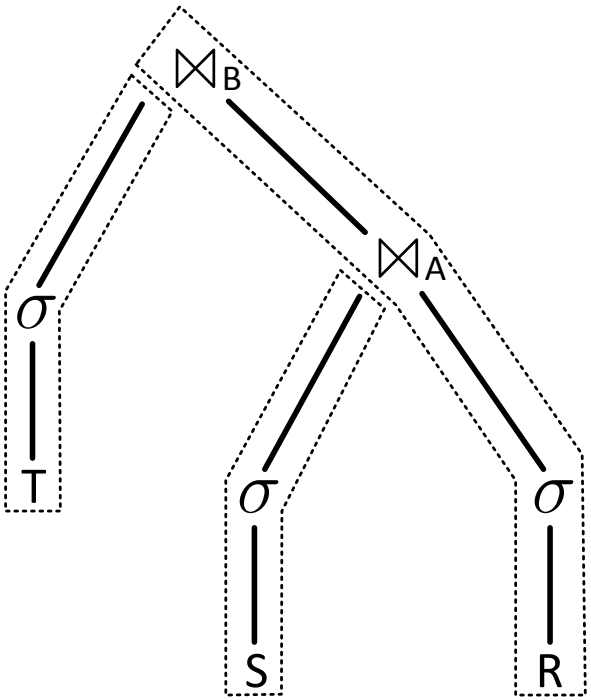
- Each pipeline is parallelized individually using all threads

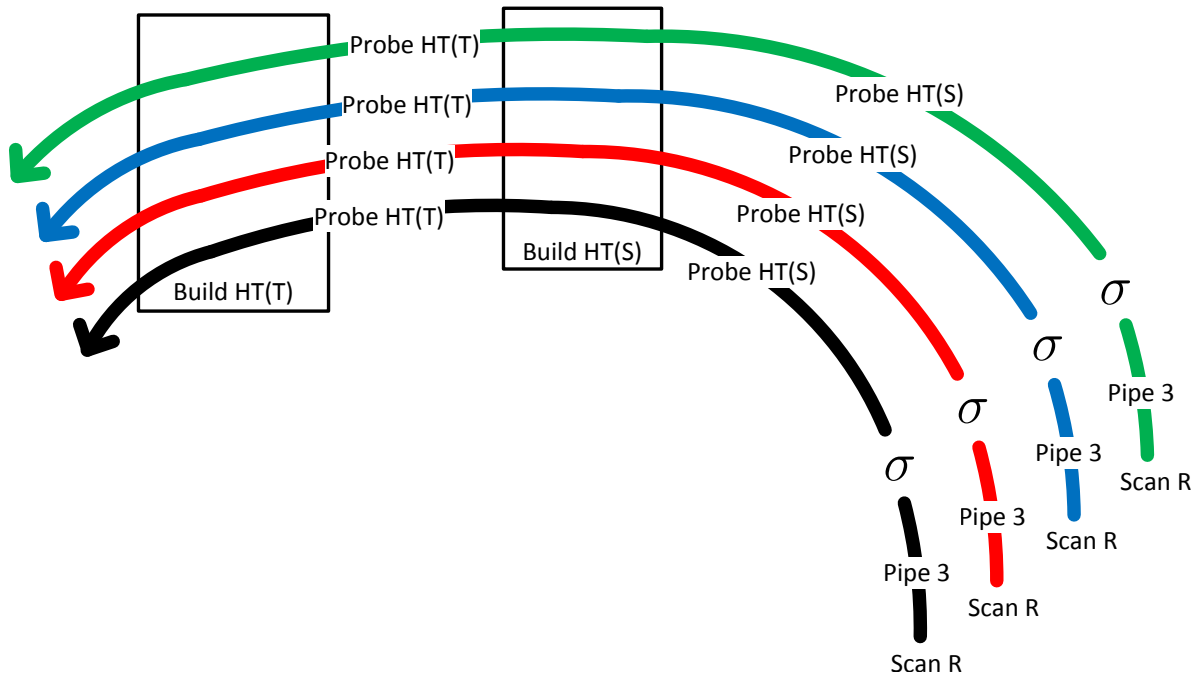




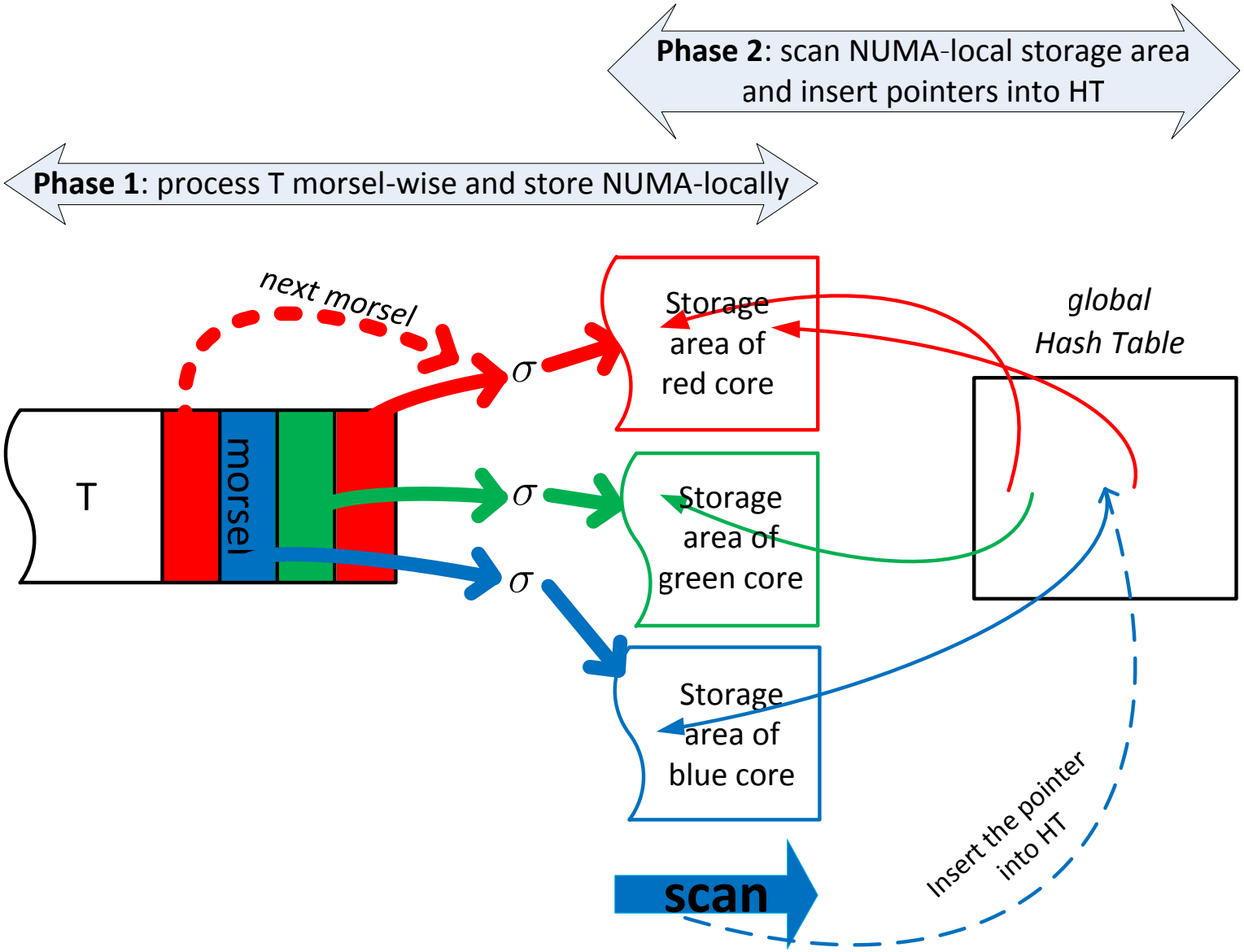
Morsel-Driven Query Execution (2)

- Each pipeline is parallelized individually using all threads



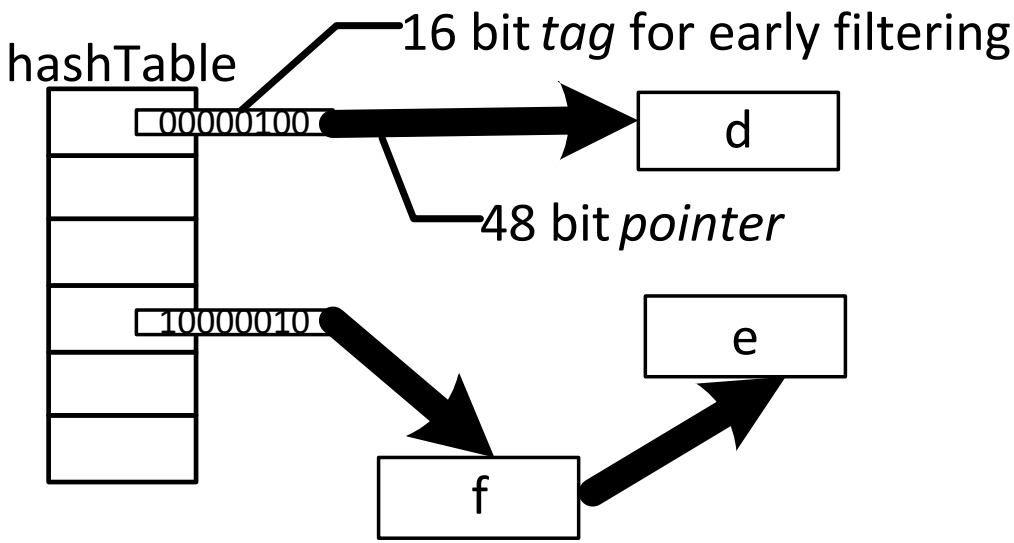


Lock-Free Hash Table



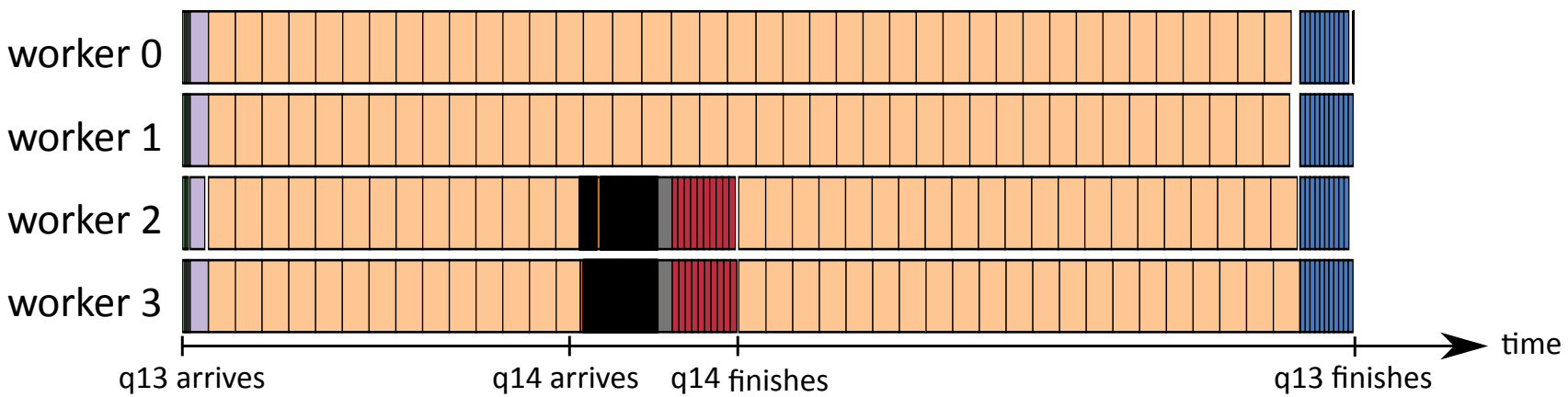
Hash Tagging

- Unused bits in pointers act as a cheap bloom filter



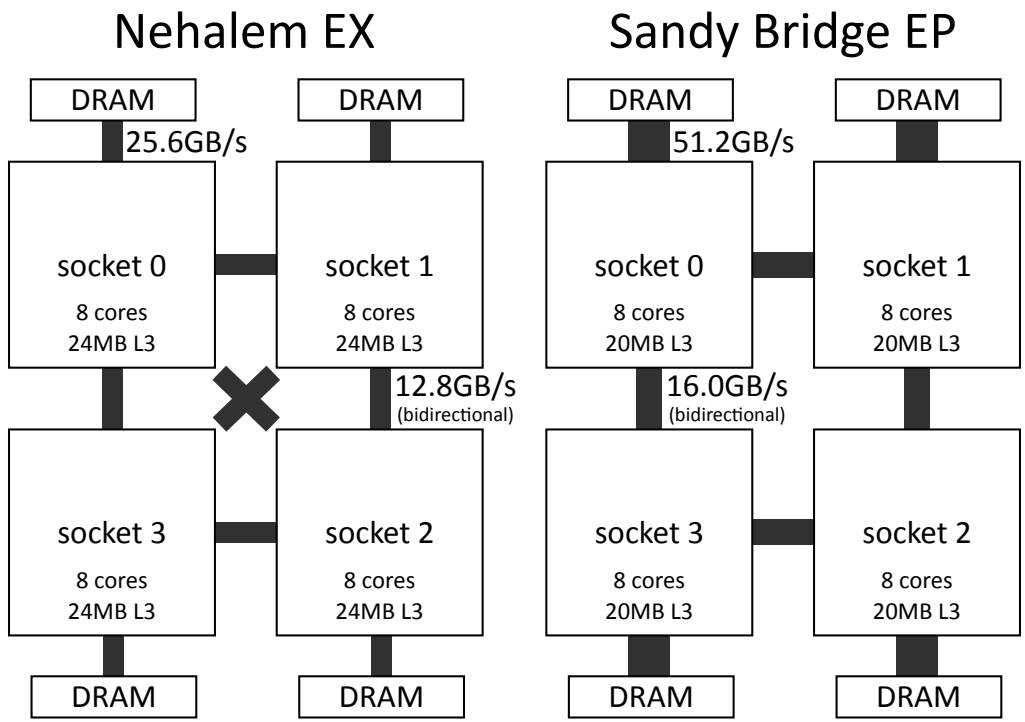
Morsels

- No load imbalances: all workers finish very close in time
- Morsels allow to react to workload changes: priority-based scheduling of dynamic workloads possible



NUMA Awareness

- NUMA awareness at the morsel level
- E.g., Table scan:
  - Relations are partitioned over NUMA nodes
  - Worker threads ask for NUMA-local morsels
  - May steal morsels from other sockets to avoid idle workers



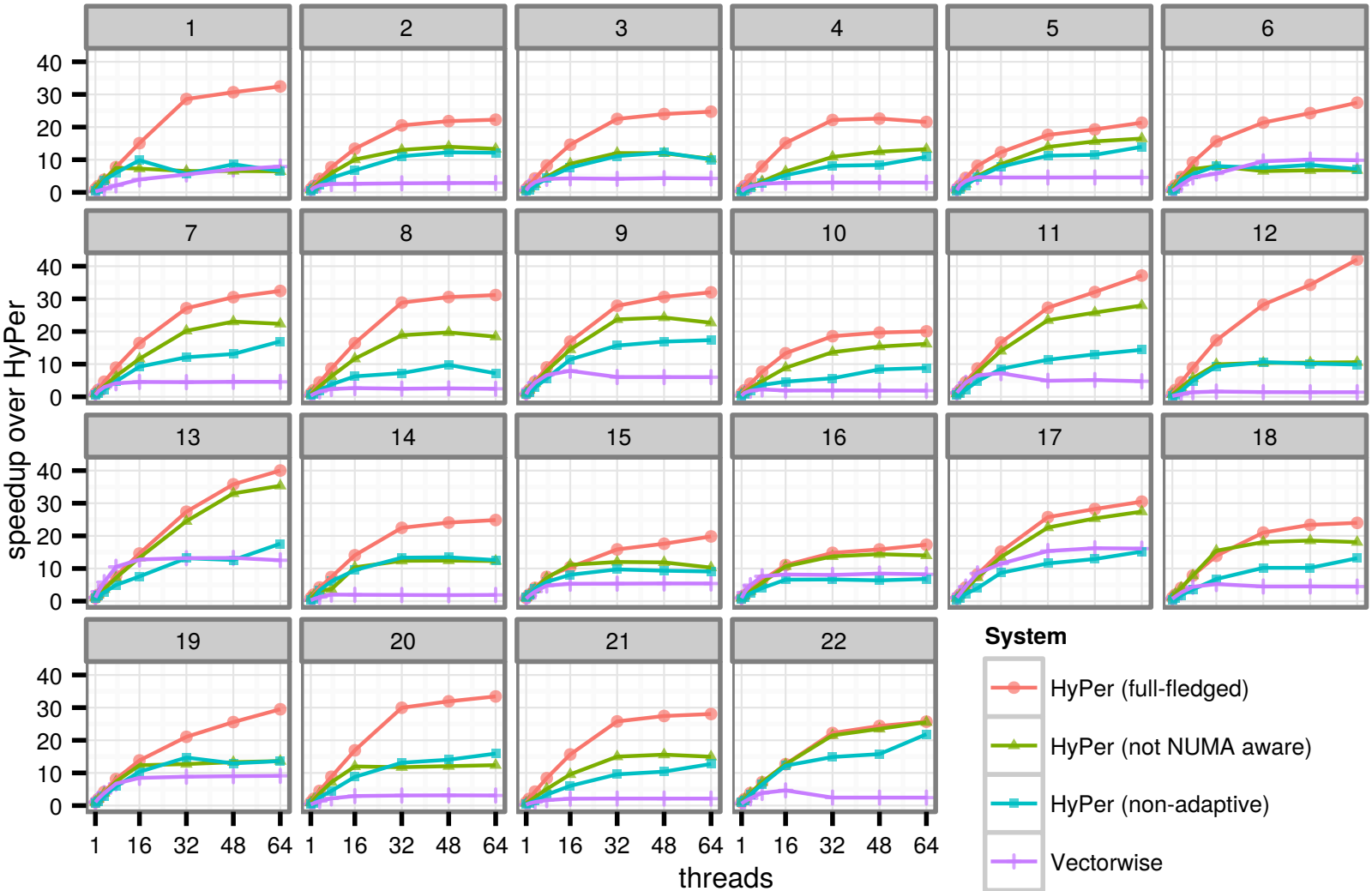
More Details in Paper

- Must parallelize everything (Amdahl's law)
  - Aggregation: partitioning-based with cheap pre-aggregation
  - Merge sort
- Lock-free work-stealing data structure
- Linux kernel scalability issues on heavy writes to 4KB memory pages

Evaluation

- HyPer
  - High-performance main-memory DBMS
  - Supports superset of SQL-92
  - Data-centric query compilation
  - Morsel-driven parallelism
- Vectorwise (Actian Vector)
  - Current official non-clustered TPC-H leader
  - Very high single-threaded performance (similar to HyPer)
  - Volcano-style parallelism (work on better scalability ongoing)
- Nehalem EX: 32 cores

Evaluation: TPC-H



Conclusions

- Getting good scalability and performance on many-core systems is challenging
- It not possible to bolt on parallelism to an existing query engine, one must redesign it with modern hardware in mind

www.hyper-db.com