EST. 1785
UNIVERSITY OF NEW BRUNSWICK

Computer Science

Dataset Skew Resilience in Main Memory Parallel Hash Joins

Puya Memarzia, Virendra Bhavsar, and Suprio Ray

Faculty of Computer Science, University of New Brunswick, Fredericton, New Brunswick, Canada

ABSTRACT

The rapid rise of big data and data analytics are driving the need for greater algorithmic efficiency. Parallel main memory hash joins are prescribed to accelerate database joins. However, the performance of these joins can be hindered by dataset skew. We tested four popular hash join algorithms on an extensive array of datasets. Our results demonstrate that hash joins are acutely affected by dataset skew, and performance is further hampered if the data is unordered. To address these issues, we propose two different hash tables. First, we use a separate chaining hash table that is based on an existing implementation that we have modified. This version outperforms the original implementation on skewed datasets by up to three orders of magnitude. Second, we propose a novel hash table that can further improve performance by up to 17.3x.

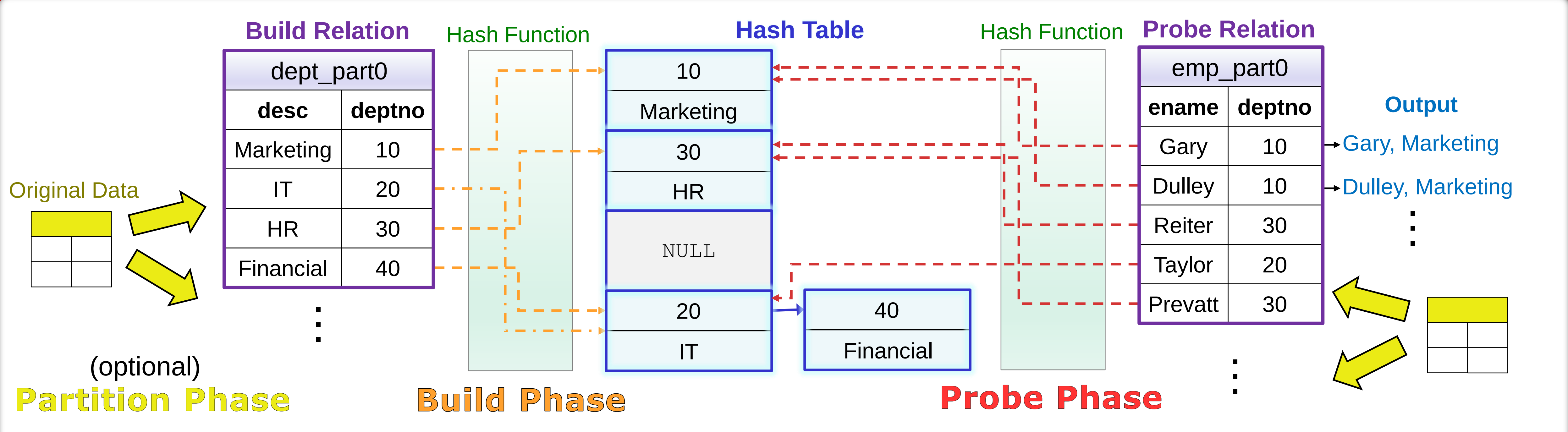


Figure 1. Example of a hash join between two corresponding tables or partitions from each table – using a separate chaining hash table

Motivation

In-memory hash joins on datasets that are skewed and/or shuffled are significantly slower than ordered non-skewed datasets. The performance penalty can be several orders of magnitude.

- Throwing more hardware at the problem is not always an option.
- The design and implementation of the hash table is one of the main bottlenecks
- Join time deteriorates due to increased memory lookups, lock contention, and CPU cache and TLB misses

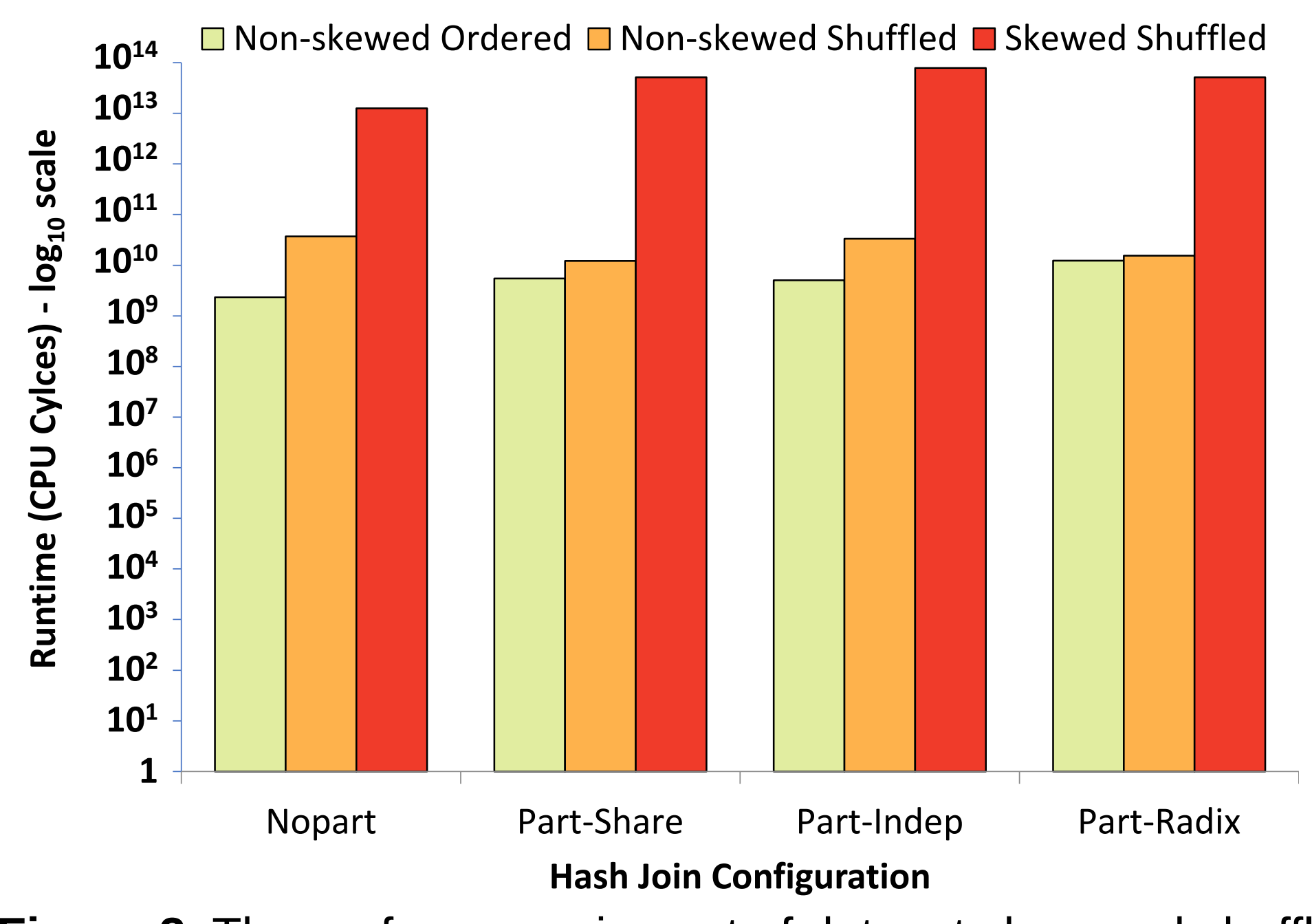
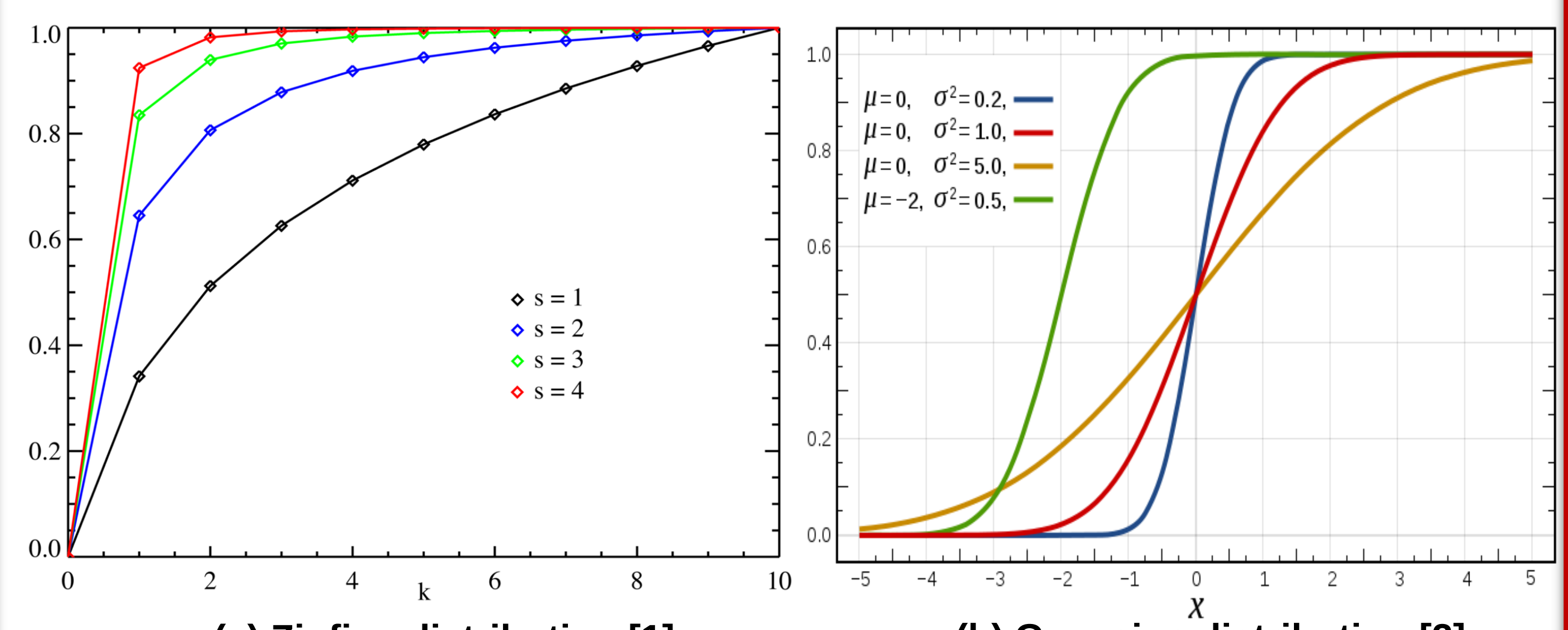


Figure 2. The performance impact of dataset skew and shuffling on four hash join algorithms (note that y axis is log scale)

Examples of Dataset Skew

- Joins on non-primary key columns (duplicate keys)
- Parallel multiway joins
- Compound queries (multiple joins and aggregations)
- Many examples in real-world data
  - The size of cities and the length and frequency of words can be modeled with Zipfian distributions
  - Measurement errors and IQ scores often follow Gaussian distributions
- Dataset skew can occur in attributes as well as in the correlation between relations



(a) Zipfian distribution [1] (b) Gaussian distribution [2]

Figure 3. Cumulative distribution function of data distributions that mimic some skewed datasets

Experiments

- We generated large datasets (hundreds of millions of records) that vary in terms of distribution, correlation, shuffling, and skew on the build and probe tables
- We measured hash join execution time, with three different hash table implementations
  - Separate Chaining hash table (SC) is the original baseline
  - Modified Separate Chaining hash table (MSC)
  - Custom Cuckoo Hashing implementation (CCH)

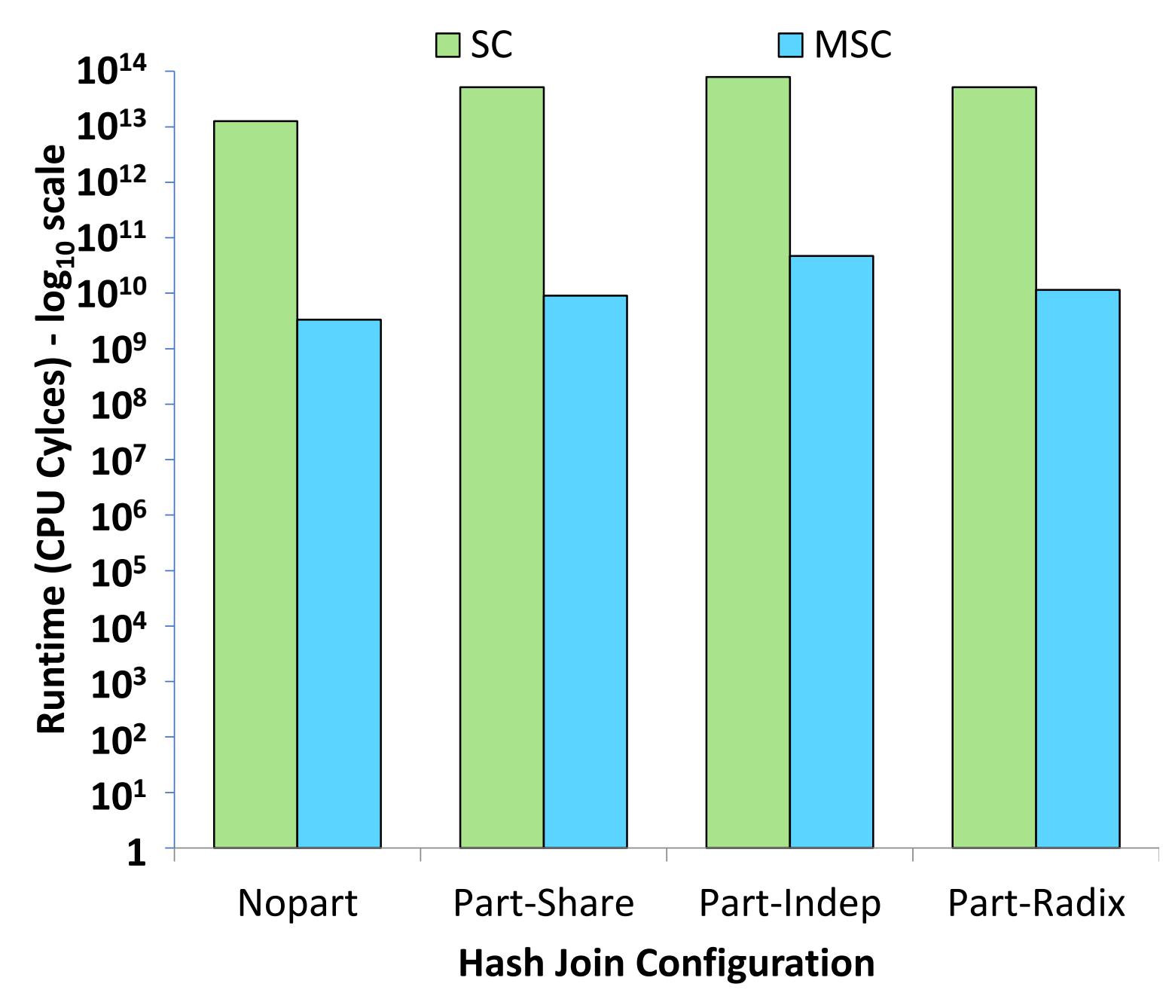


Figure 4. Original (SC) versus modified hash table (MSC) on skewed dataset

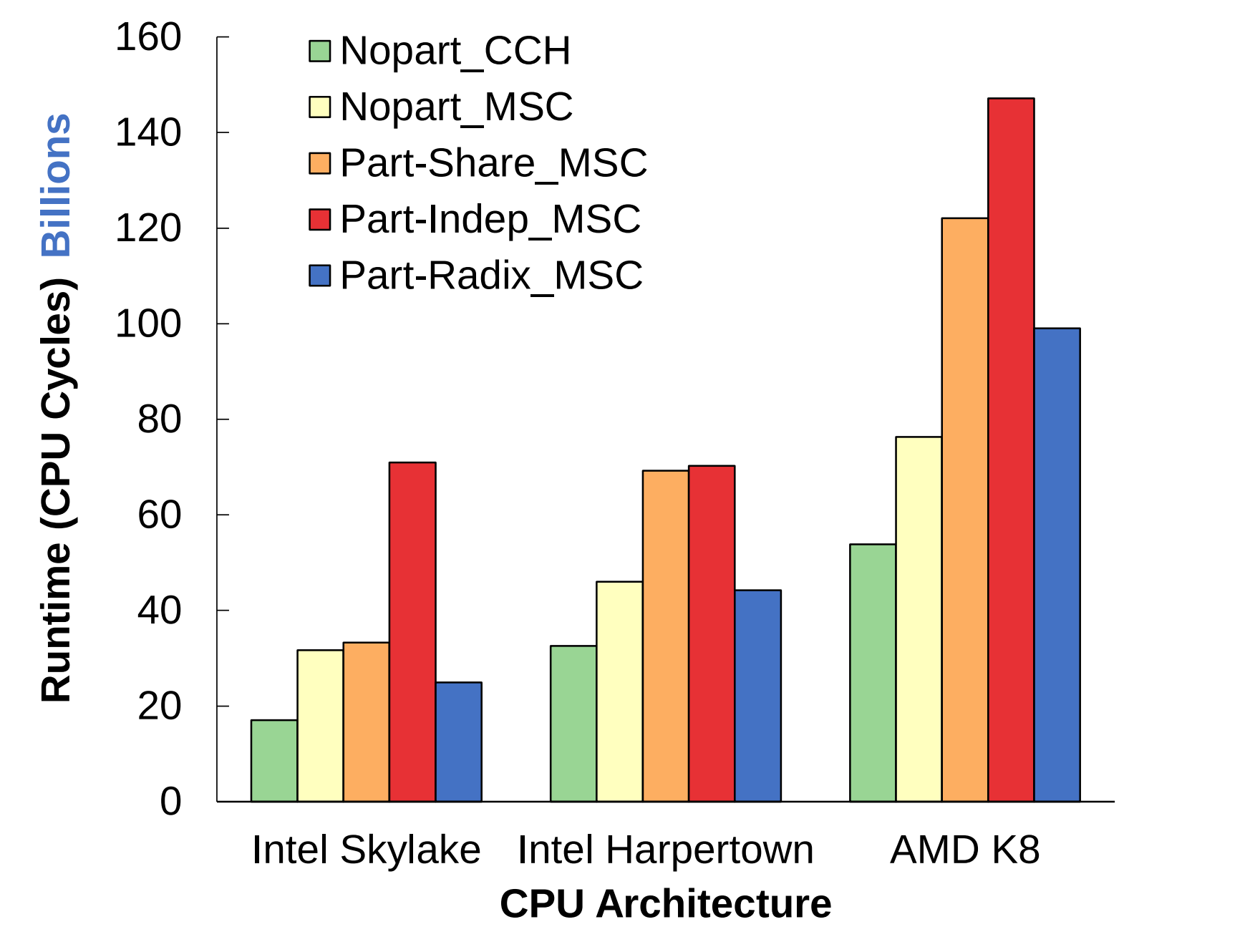


Figure 5. Evaluation of three different CPU architectures on shuffled dataset

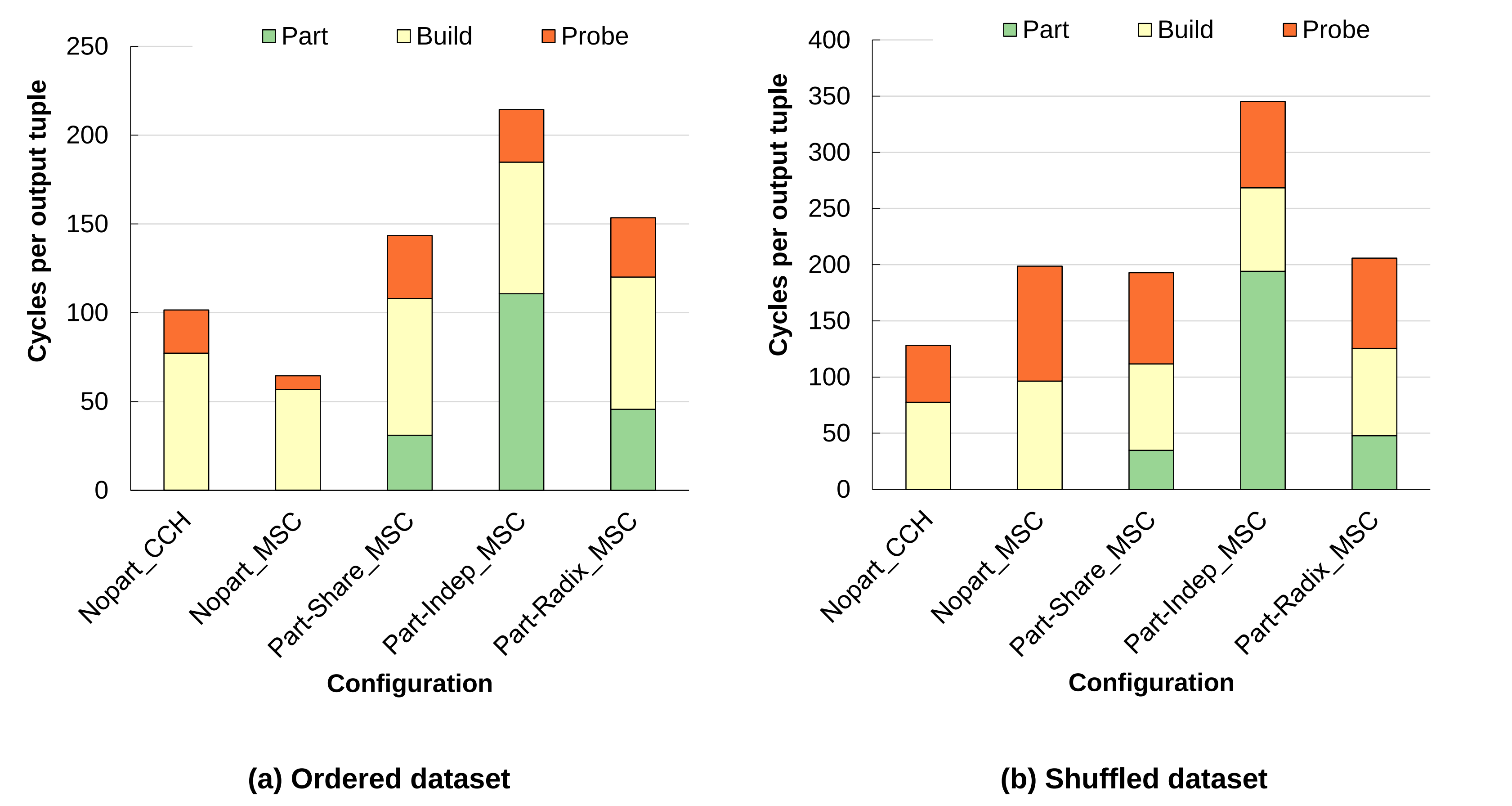


Figure 6. Breakdown of hash join phases on Zipf-skewed datasets

Hash Join Configurations

Hash joins can leverage data parallelism to scale up on multicore processors. Hash join configurations can be categorized based on their partitioning scheme (or lack thereof) and parallel implementation.

1. No Partitioning (**Nopart**): one large lock-protected hash table shared by all threads
2. Shared Partitioning (**Part-Share**): multiple lock-protected partitions shared by all threads
3. Independent Partitioning (**Part-Indep**): private lock-less partitions for each thread
4. Radix Partitioning (**Part-Radix**): a hierarchy of partitions with dynamic load balancing

Conclusion

- Hash table skew resilience is an important feature to consider when designing parallel hash join implementations
- Our approach improves hash join times across a variety of datasets and architectures, allowing for more work with the same resources
- The dataset and CPU can help determine the most efficient method
- Applications such as machine learning, analytics, graph processing, and data warehousing, could benefit from skew resilient algorithms

References: [1] Image from Wikimedia commons - https://commons.wikimedia.org/wiki/File:Zipf_distribution_CMF.png#/media/File:Zipf_distribution_CMF.png - CC-BY-SA 3.0 [2] By Inductiveload (self-made, Mathematica, Inkscape) [Public domain], via Wikimedia Commons https://commons.wikimedia.org/w/index.php?curid=3817960

Sponsored by: NBIF NSERC CRSNG