Polynesia:
Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design

P&S Processing-in-Memory
Spring 2022
16 June 2022

Amirali Boroumand
Geraldo F. Oliveira
Saugata Ghose
Onur Mutlu

SAFARI Google University of Illinois Urbana-Champaign ETH Zürich

Executive Summary

- Context: Many applications need to perform real-time data analysis using an Hybrid Transactional/Analytical Processing (HTAP) system
  - An ideal HTAP system should have **three properties**: (1) **data freshness** and **consistency**, (2) **workload-specific optimization**, (3) **performance isolation**

- Problem: Prior works **cannot achieve all properties** of an ideal HTAP system

- Key Idea: Divide the system into transactional and analytical **processing islands**
  - Enables **workload-specific optimizations** and **performance isolation**

- Key Mechanism: Polynesia, a novel hardware/software cooperative design for in-memory HTAP databases
  - Implements **custom algorithms and hardware** to reduce the costs of **data freshness** and **consistency**
  - Exploits **PIM** for analytical processing to alleviate **data movement**

- Key Results: Polynesia outperforms three state-of-the-art HTAP systems
  - Average transactional/analytical throughput improvements of **1.7x/3.7x**
  - **48%** reduction on energy consumption

Outline

Outline

Real-Time Analysis

An explosive interest in many applications domains to perform data analytics on the most recent version of data (real-time analysis)

Use transactions to record each periodic sample of data from all sensors

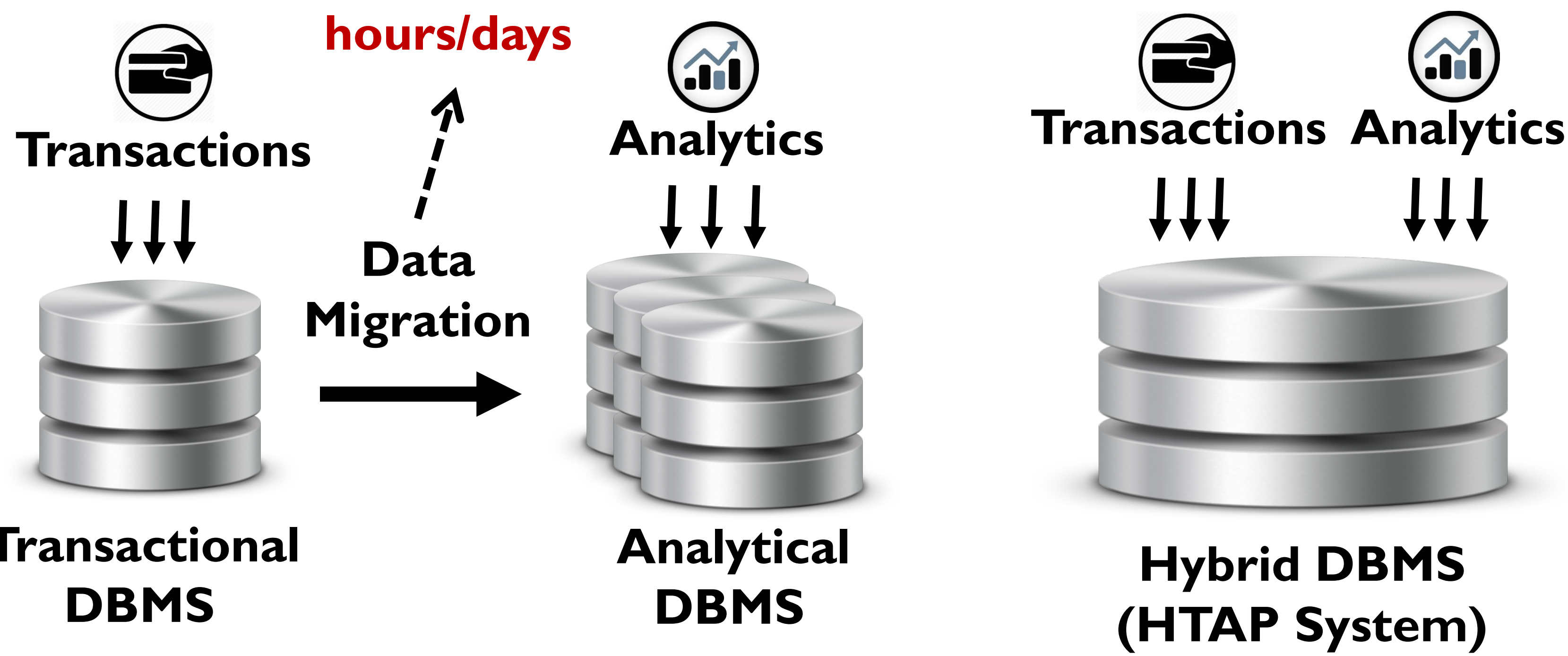
Run analytics across sensor data to make real-time steering decisions

Self-Driving Cars

For these applications, it is critical to analyze the transactions in real-time as the data's value diminishes over time

HTAP: Supporting Real-Time Analysis

Traditionally, new transactions (updates) are propagated to the analytical database using a periodic and costly process



To support real-time analysis: a single hybrid DBMS is used to execute both transactional and analytical workloads

Ideal HTAP System Properties

An ideal HTAP system should have three properties:

1 **Workload-Specific Optimizations**

- Transactional and analytical workloads must benefit from their own specific optimizations

2 **Data Freshness and Consistency Guarantees**

- Guarantee access to the most recent version of data for analytics while ensuring that transactional and analytical workloads have a consistent view of data

3 **Performance Isolation**

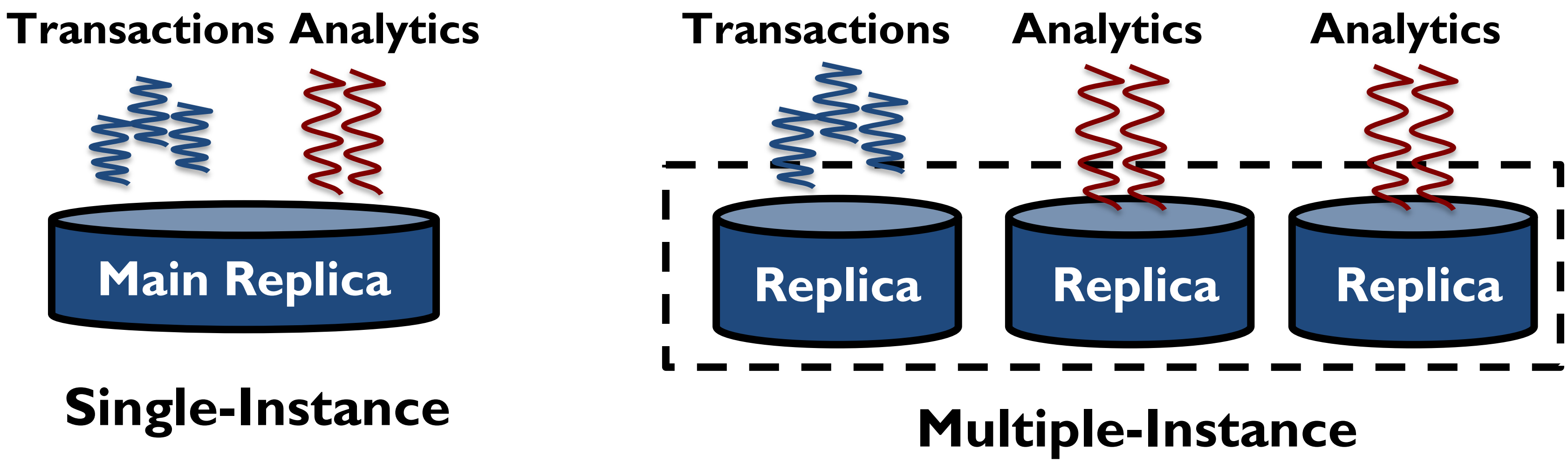
- Latency and throughput of transactional and analytical workloads are the same as if they were run in isolation

Achieving all three properties at the same time is very challenging

Outline

State-of-the-Art HTAP Systems

We study two major types of HTAP systems:



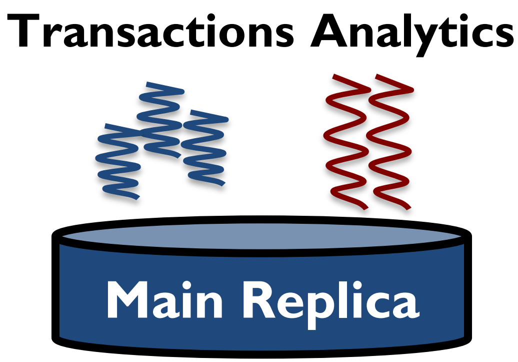
We observe two key problems:

1 Data freshness and consistency mechanisms are costly and cause a drastic reduction in throughput

2 These systems fail to provide performance isolation because of high main memory contention

State-of-the-Art HTAP Systems

We study two major types of HTAP systems:



Single-Instance

Multiple-Instance

We observe two key problems:

1 Data freshness and consistency mechanisms are costly and cause a drastic reduction in throughput

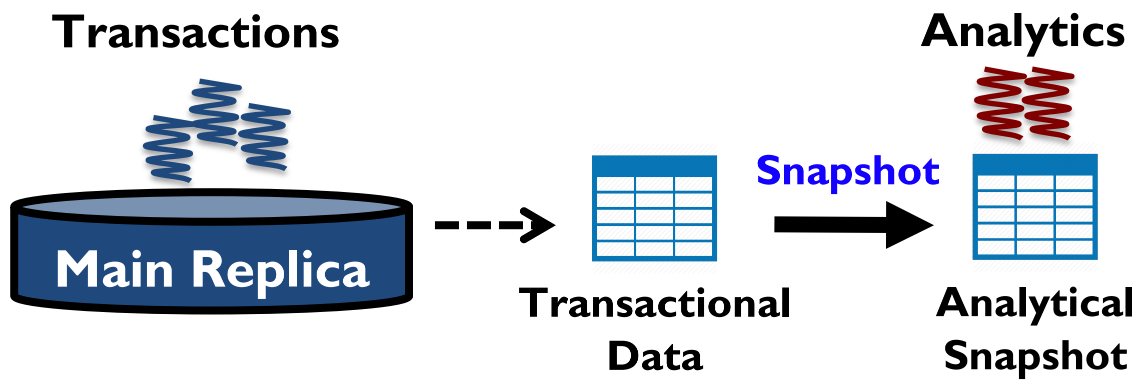
2 These systems fail to provide performance isolation because of high main memory contention

Single-Instance: Data Consistency

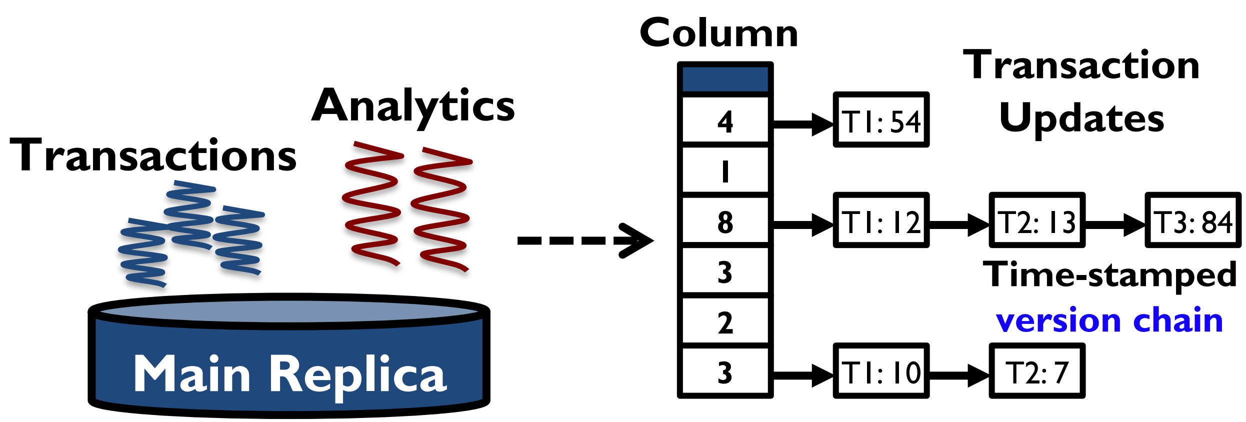
Since both analytics and transactions work on the same data concurrently, we need to ensure that the data is consistent

There are two major mechanisms to ensure consistency:

1 Snapshotting

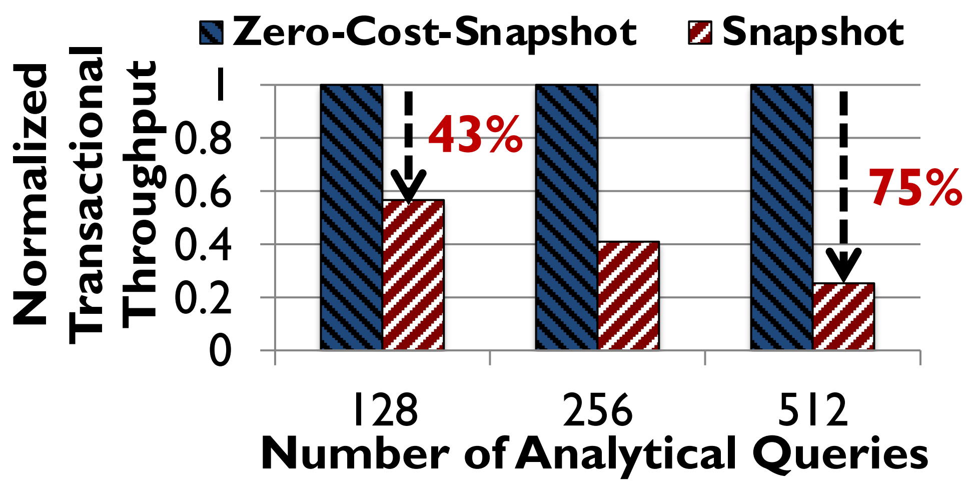


2 Multi-Version Concurrency Control (MVCC)



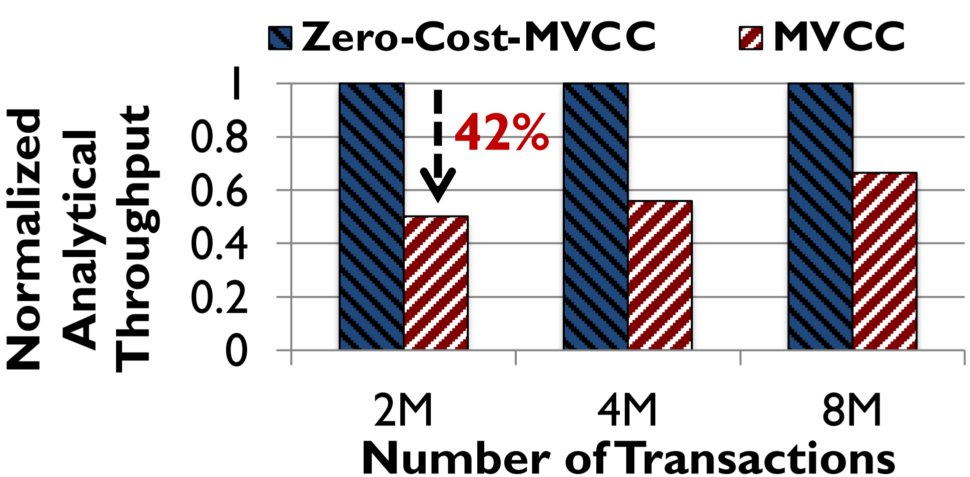
Drawbacks of Snapshotting and MVCC

We evaluate the throughput loss caused by Snapshotting and MVCC:



Throughput loss comes from memcpy operation:

generates a large amount of data movement

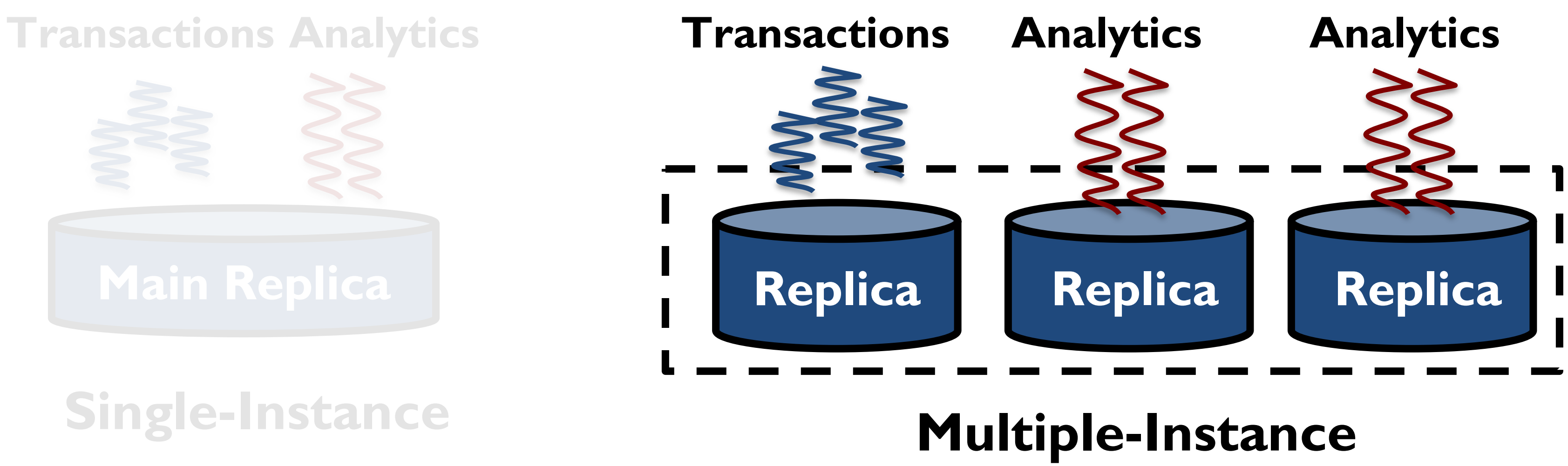


Throughput loss comes from long version chains:

expensive time-stamp comparison and a large number of random memory accesses

State-of-the-Art HTAP Systems

We study two major types of HTAP systems:



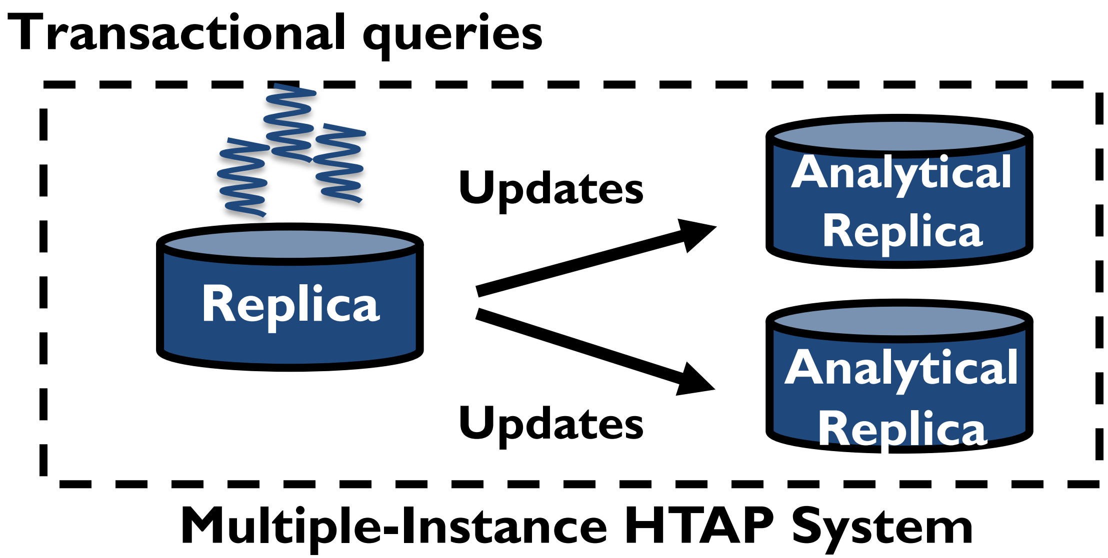
We observe two key problems:

1 Data freshness and consistency mechanisms are costly and cause a drastic reduction in throughput

2 These systems fail to provide performance isolation because of high main memory contention

Maintaining Data Freshness

One of the major challenges in multiple-instance systems is to keep analytical replicas up-to-date



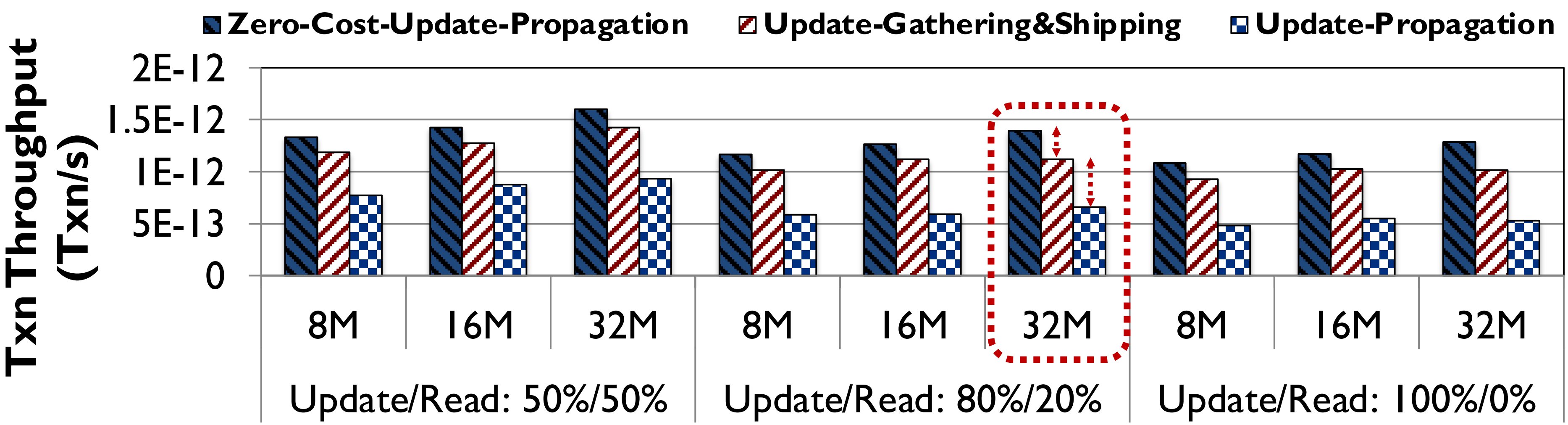
To maintain data freshness (via Update Propagation):

1 **Update Gathering and Shipping: gather updates from transactional threads and ship them to analytical the replica**

2 **Update Application: perform the necessary format conversation and apply those updates to analytical replicas**

Cost of Update Propagation

We evaluate the throughput loss caused by Update Propagation:



Transactional throughput reduces by up to 21.2% during the update gathering & shipping process

Transactional throughput reduces by up to 64.2% during the update application process

Problem and Goal

Problems:

1 State-of-the-art HTAP systems do not achieve all of the desired HTAP properties

2 Data freshness and consistency mechanisms are data-intensive and cause a drastic reduction in throughput

3 These systems fail to provide performance isolation because of high main memory contention

Goal:

Take advantage of custom algorithm and processing-in-memory (PIM) to address these challenges

Outline

Polynesia

Key idea: **partition computing resources into two types of isolated and specialized processing islands**

Isolating **transactional islands** from **analytical islands** allows us to:

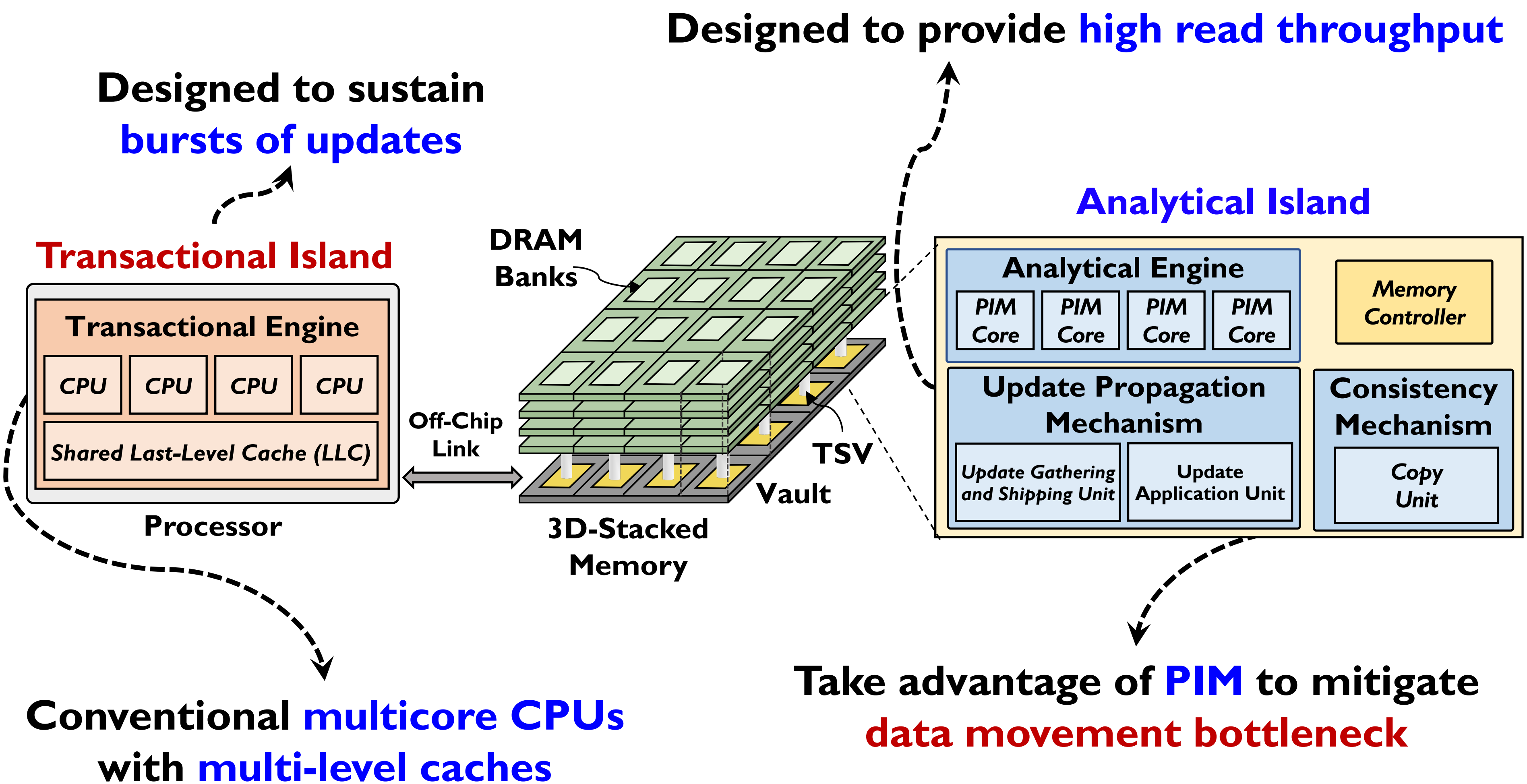
1. **Apply workload-specific optimizations to each island**

2. **Avoid high main memory contention**

3. **Design efficient data freshness and consistency mechanisms without incurring high data movement costs**
   - **Leverage processing-in-memory (PIM) to reduce data movement**
   - **PIM mitigates data movement overheads by placing computation units nearby or inside memory**

Polynesia: High-Level Overview

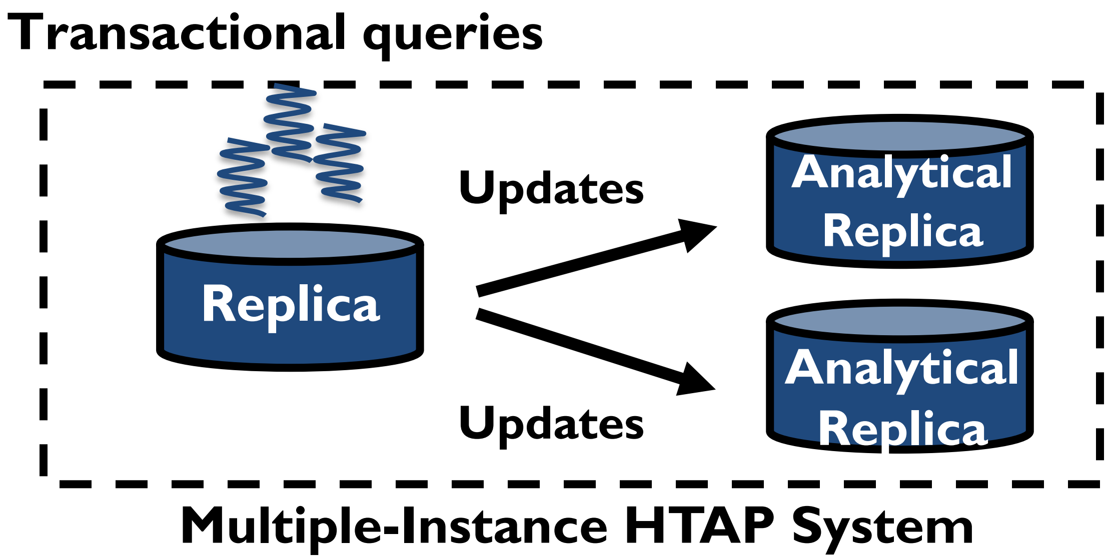
Each island includes (1) a replica of data, (2) an optimized execution engine, and (3) a set of hardware resources



Outline

Maintaining Data Freshness

One of the major challenges in multiple-instance systems is to keep analytical replicas up-to-date



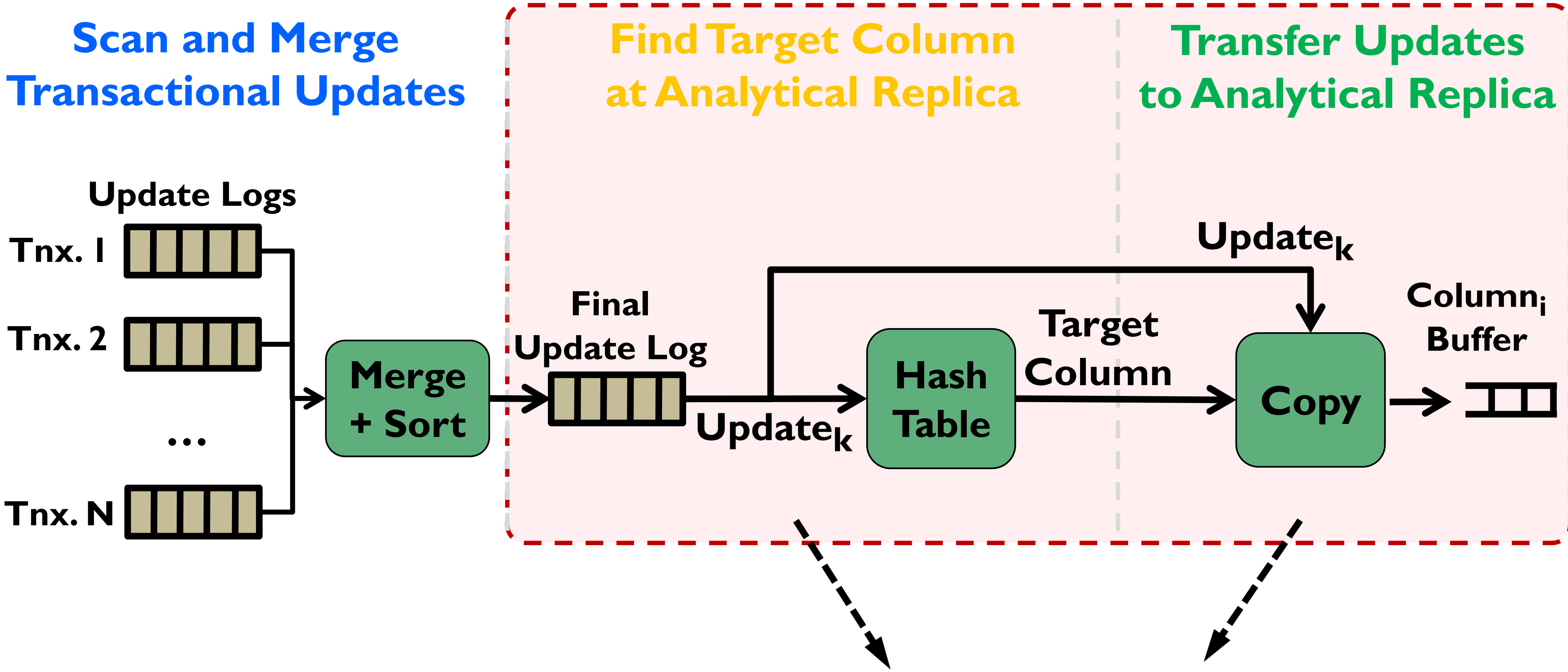
To maintain data freshness (via Update Propagation):

1 **Update Gathering and Shipping: gather updates from transactional threads and ship them to analytical the replica**

2 **Update Application: perform the necessary format conversation and apply those updates to analytical replicas**

Update Gathering & Shipping: Algorithm

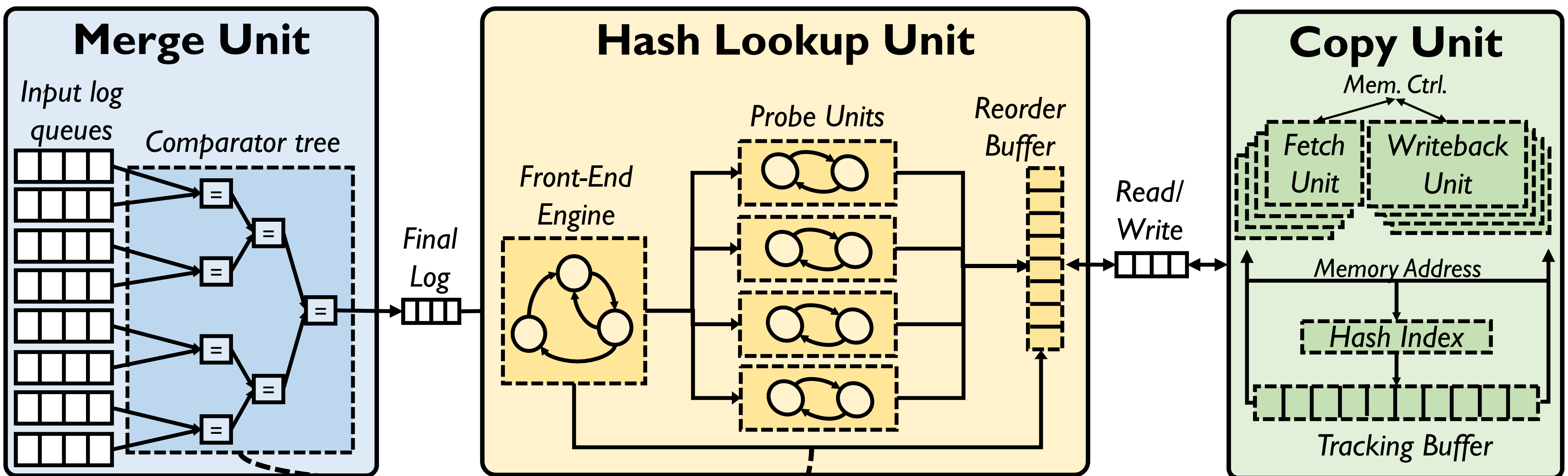
Update gathering & shipping algorithm has three major stages:



2nd and 3rd stages generate a large amount of data movement and account for 87.2% of our algorithm's execution time

Update Gathering & Shipping: Hardware

To avoid these bottlenecks, we design a new hardware accelerator, called update gathering & shipping unit



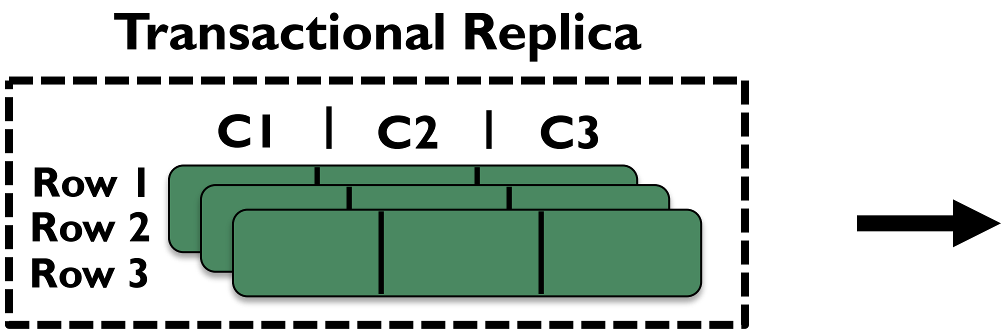
A 3-level comparator tree to merge updates

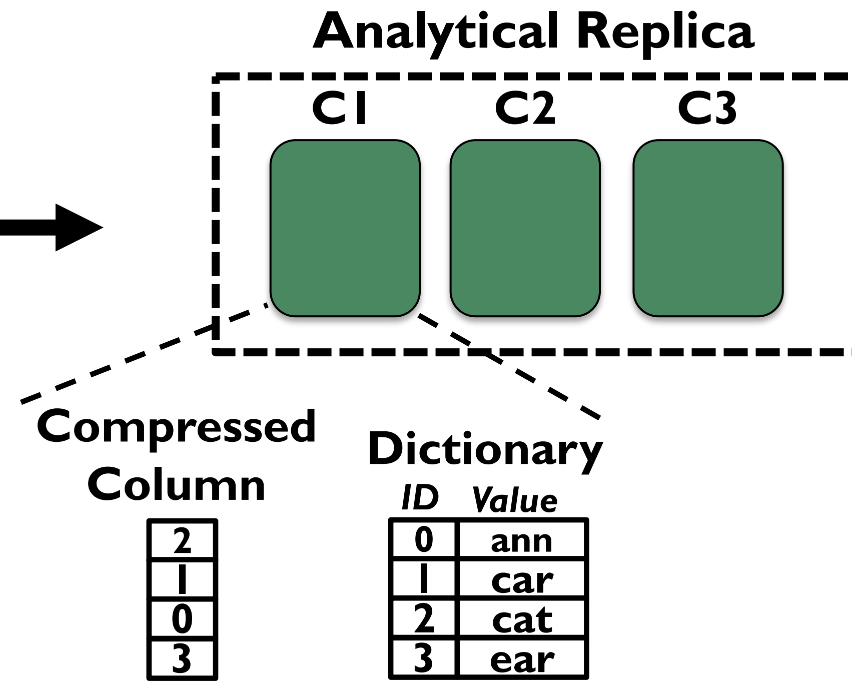
Decoupled hash computation from the hash bucket traversal to allow for concurrent hash lookups

Multiple fetch and write-back units to issue multiple memory accesses concurrently

Update Propagation: Update Application

***Goal*: perform the necessary format conversation and apply transactional updates to analytical replicas**





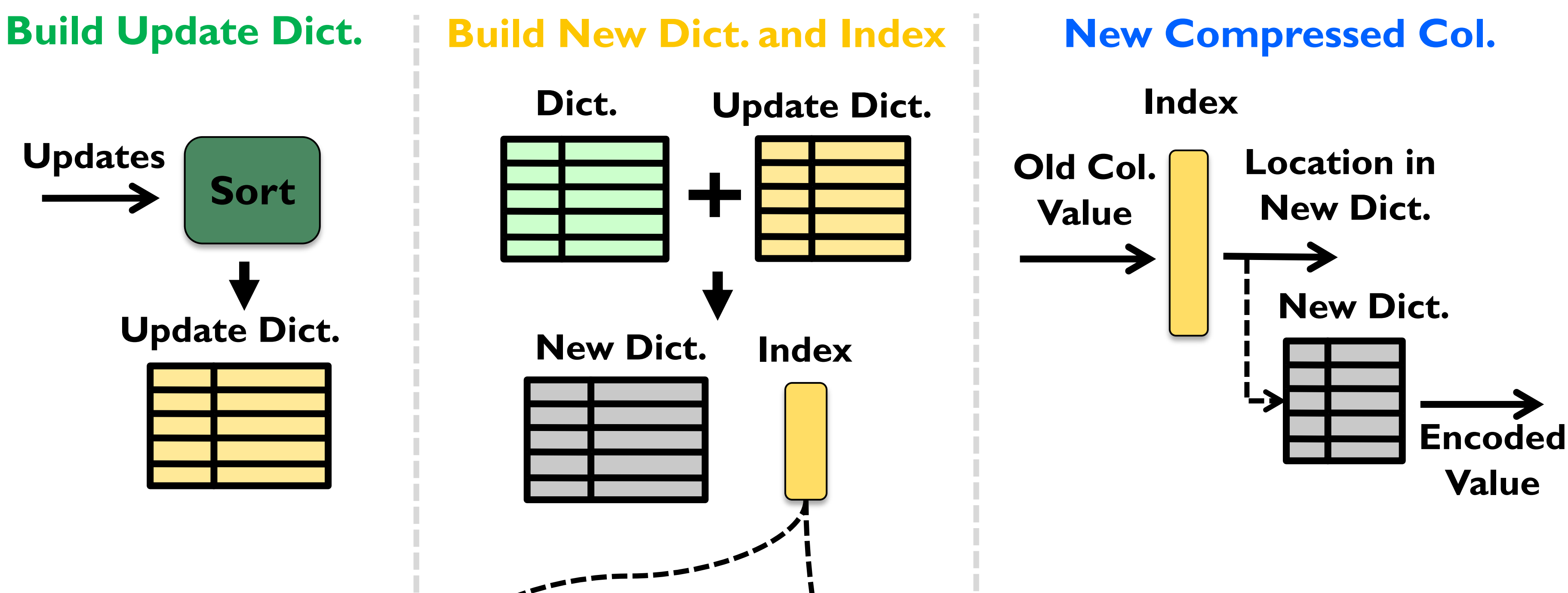
Update: Row 2, Column 1 and 3

1. **A simple tuple update in row-wise layout leads to multiple random accesses in column-wise layout**

2. **Updates change encoded value in the dictionary → (1) Need to reconstruct the dictionary, and (2) recompress the column**

Update Application: Algorithm

We design our update application algorithm to be aware of PIM logic characteristics and constraints

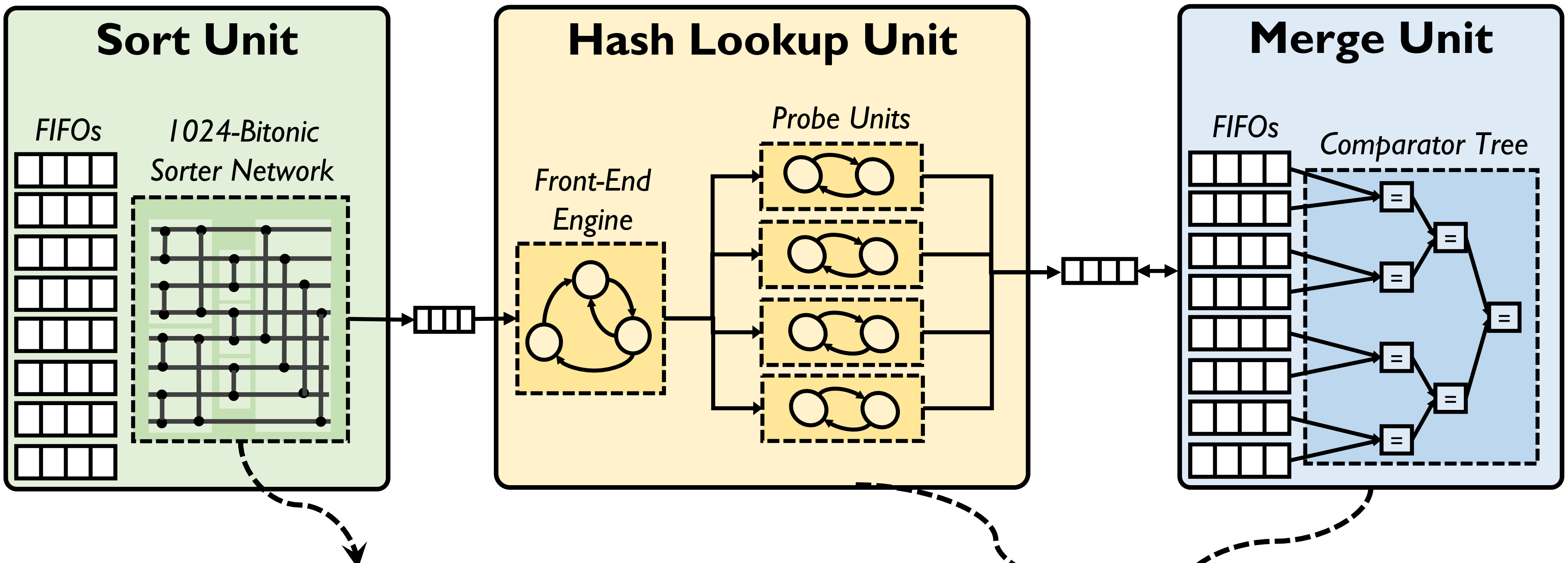


We maintain a hash index that links the old encoded value in a column to the new encoded value

Avoids the need to decompress the column and add updates, eliminating data movement and random accesses to 3D DRAM

Update Application: Hardware

We design a hardware implementation of our algorithm, and add it to each in-memory analytical island



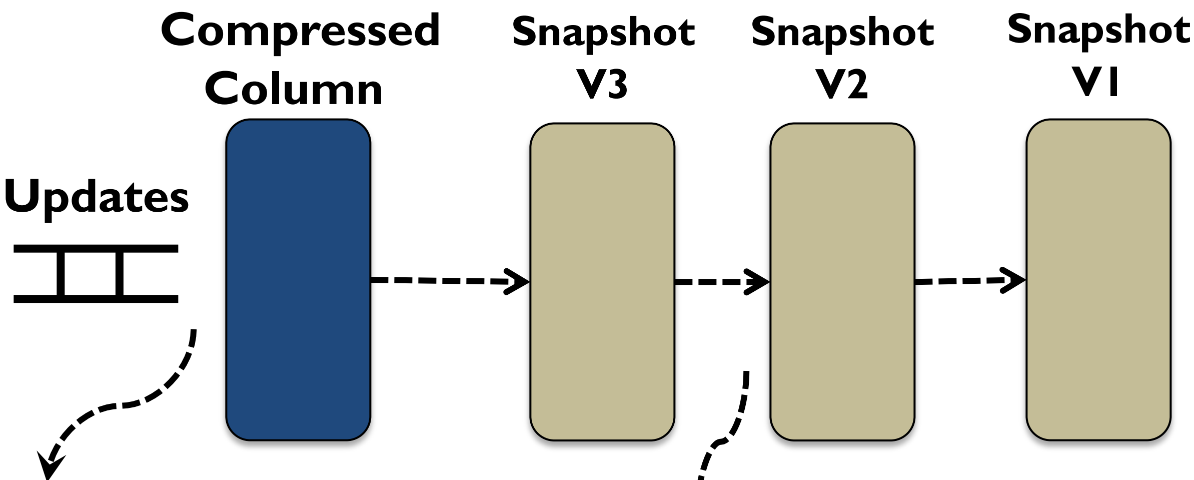
A 1024-value bitonic sorter, whose basic building block is a network of comparators

Similar design as our update gathering & shipping unit

Outline

Consistency Mechanism: Algorithm

For each column, there is a chain of snapshots where each chain entry corresponds to a version of the column



Polynesia **does not** create a snapshot every time a column is updated. Instead, Polynesia marks the column as **dirty**

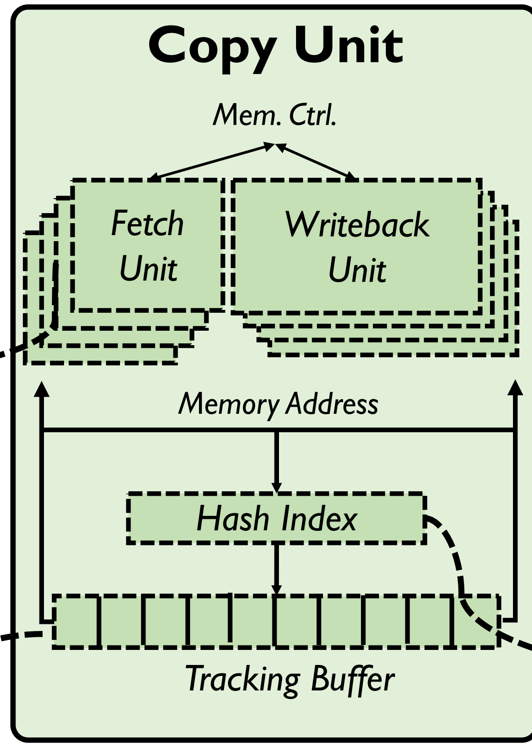
Unlike chains in MVCC, each version is associated **with a column, not a row**

Polynesia creates a new snapshot only if (1) any of the columns are dirty, and (2) no current snapshot exists for the same column

Consistency Mechanism: Hardware

Our algorithm success at satisfying performance isolation relies on how fast we can do memcpy to minimize snapshotting latency

Multiple fetch and writeback units to issue multiple memory accesses concurrently



Look-ups at the tracking buffer limit performance → use a hash index to alleviate performance bottlenecks

Track outstanding reads, as they may come back from memory out of order. Allows to immediately initiate a write after a read is complete

Outline

Analytical Engine: Query Execution

Efficient analytical query execution strongly depends on:

1 Data layout and data placement

2 Task scheduling policy

3 How each physical operator is executed

The execution of physical operators of analytical queries significantly benefit from PIM

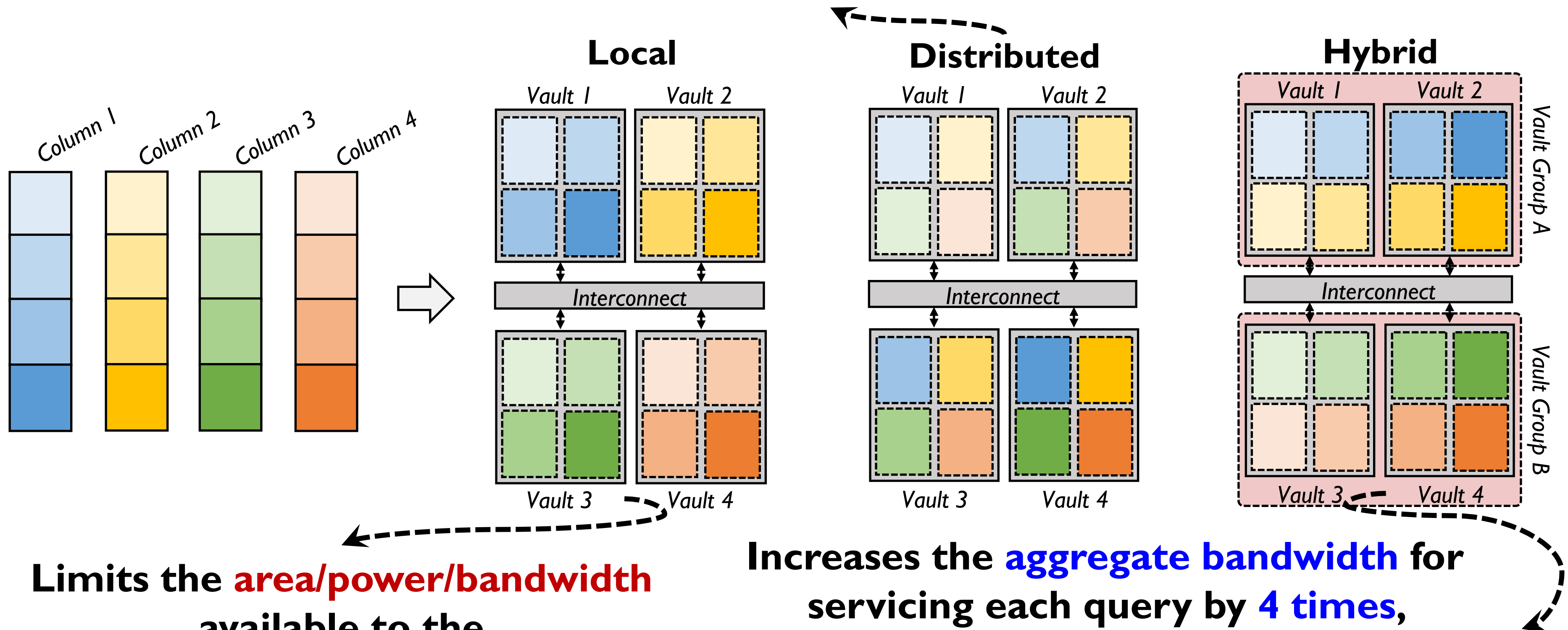
↓

Without PIM-aware data placement/task scheduler, PIM logic for operators alone cannot provide throughput

Analytical Engine: Data Placement

***Problem*: how to partition analytical data across vaults of the 3D-stacked memory**

Creates inter-vault communication overheads



Limits the area/power/bandwidth available to the analytical engine inside a vault

Increases the aggregate bandwidth for servicing each query by 4 times, and provides up to 4 times the power/area for PIM logic compared to Local

Analytical Engine: Query Execution

Other details in the paper:

Task scheduling policy

We design **a pull-based** task assignment strategy, where **PIM threads cooperatively** pull tasks from the task queue **at runtime**

How each physical operator is executed

We employ the **top-down Volcano (Iterator)** execution model to execute physical operations (e.g., scan, filter, join) while respecting operator's dependencies

Analytical Engine: Query Execution

Other details in the paper:

Task scheduling policy

Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design

Amirali Boroumand[†] Saugata Ghose[◇] Geraldo F. Oliveira[‡] Onur Mutlu[‡]

[†]*Google* [◇]*Univ. of Illinois Urbana-Champaign* [‡]*ETH Zürich*

We employ the top-down Volcano (Iterator) execution mo
execute physical operations (e.g., scan, filter, join) while resp
operator's dependencies

SAFARI

Full Draft

Outline

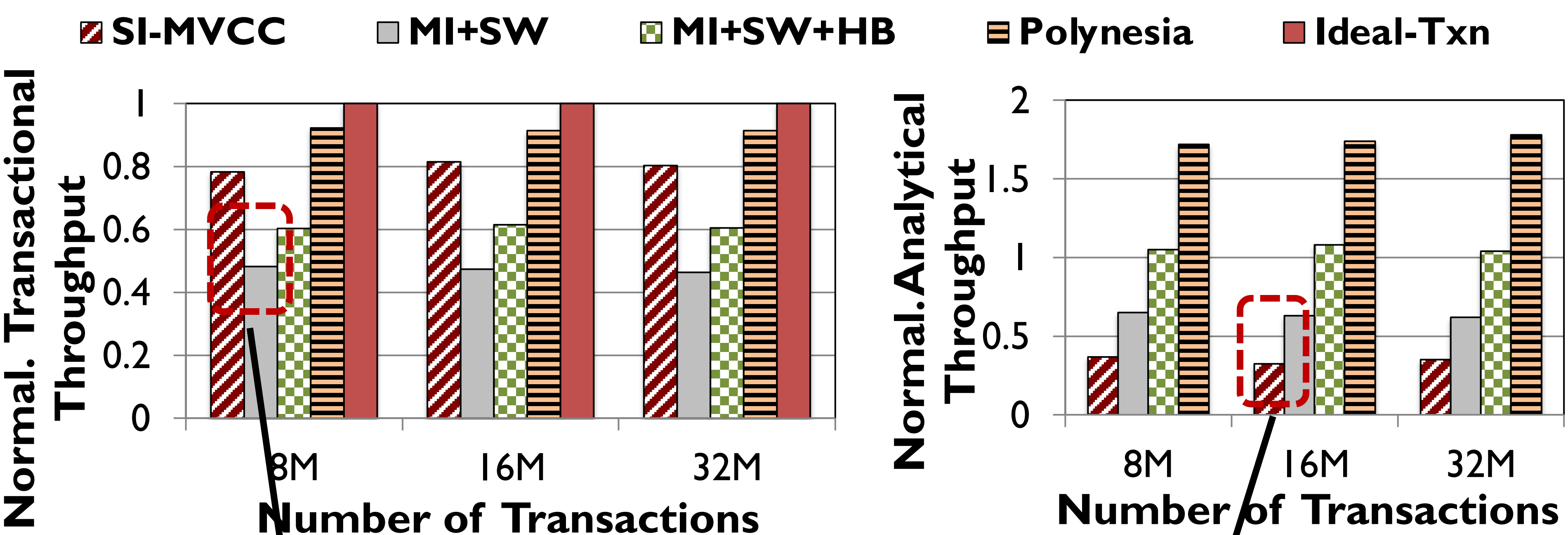
Methodology

- We adapt previous transactional/analytical engines with our new algorithms
  - DBx1000 for transactional engine
  - C-store for analytical engine

- We use gem5 to simulate Polynesia
  - Available at: https://github.com/CMU-SAFARI/Polynesia

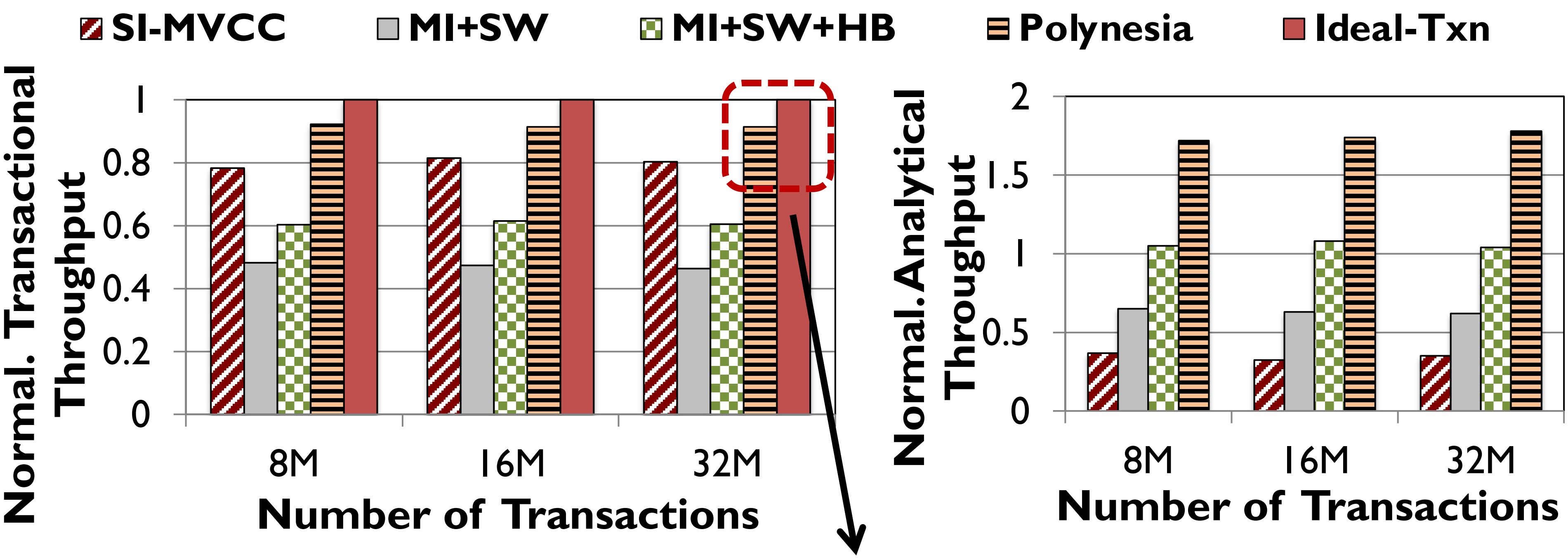
- We compare Polynesia against:
  - Single-Instance-Snapshotting (SI-SI)
  - Single-Instance-MVCC (SI-MVCC)
  - Multiple-Instance + Polynesia's new algorithms (MI+SW)
  - MI+SW+HB: MI+SW with a 256 GB/s main memory device
  - Ideal-Txn: the peak transactional throughput if transactional workloads run in isolation

End-to-End System Analysis (1/5)



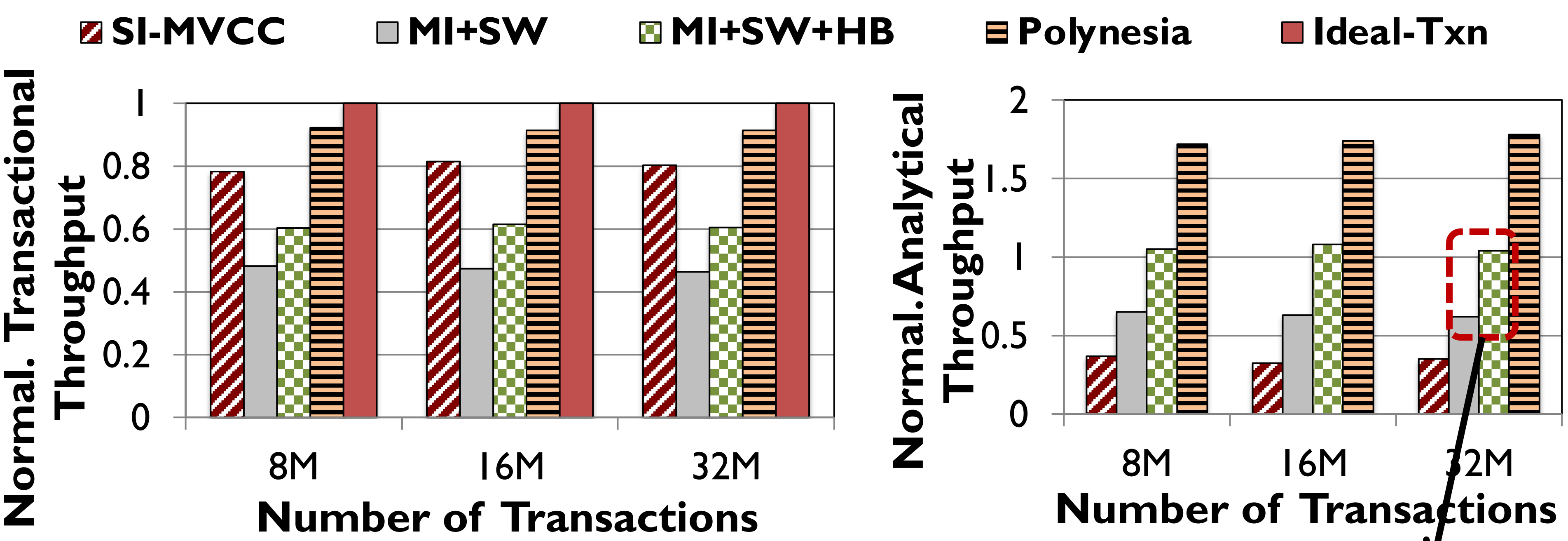
While SI-MVCC is the best baseline for transactional throughput, it degrades analytical throughput by 63.2%, due to its lack of workload-specific optimizations and consistency mechanism

End-to-End System Analysis (2/5)



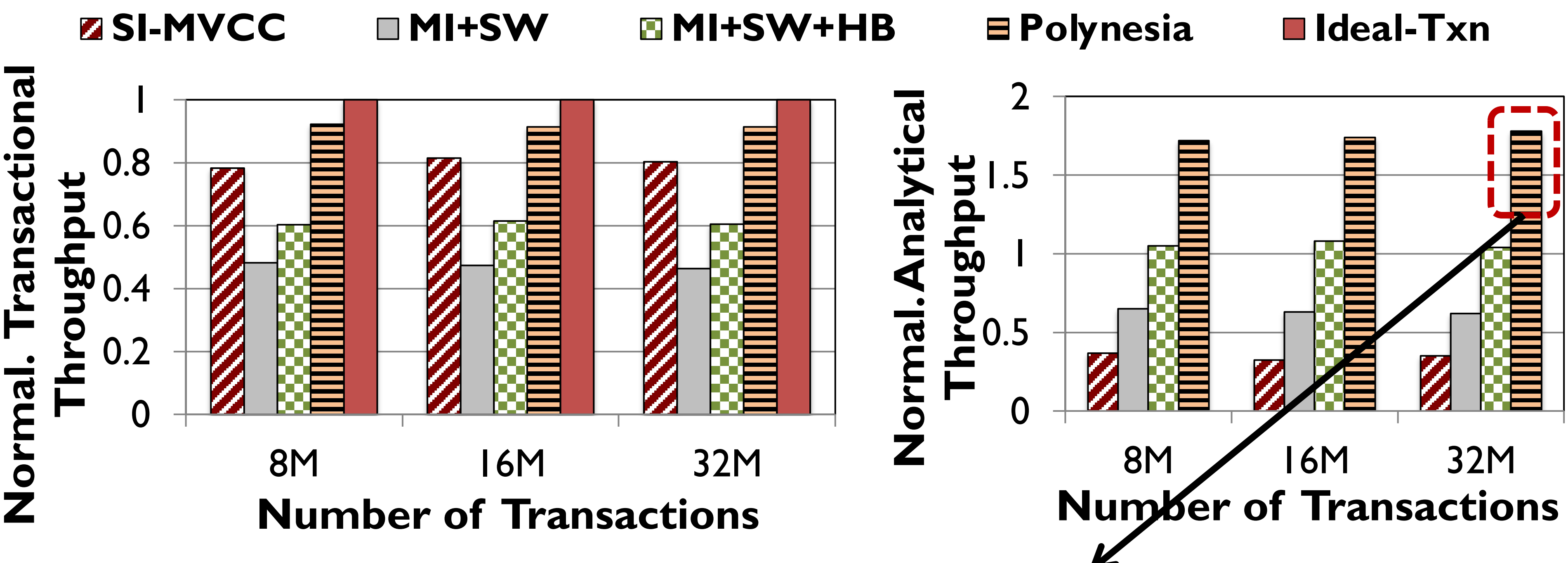
Polynesia comes within 8.4% of ideal Txn because it uses custom PIM logic for data freshness/consistency mechanisms, significantly reducing main memory contention and data movement

End-to-End System Analysis (3/5)



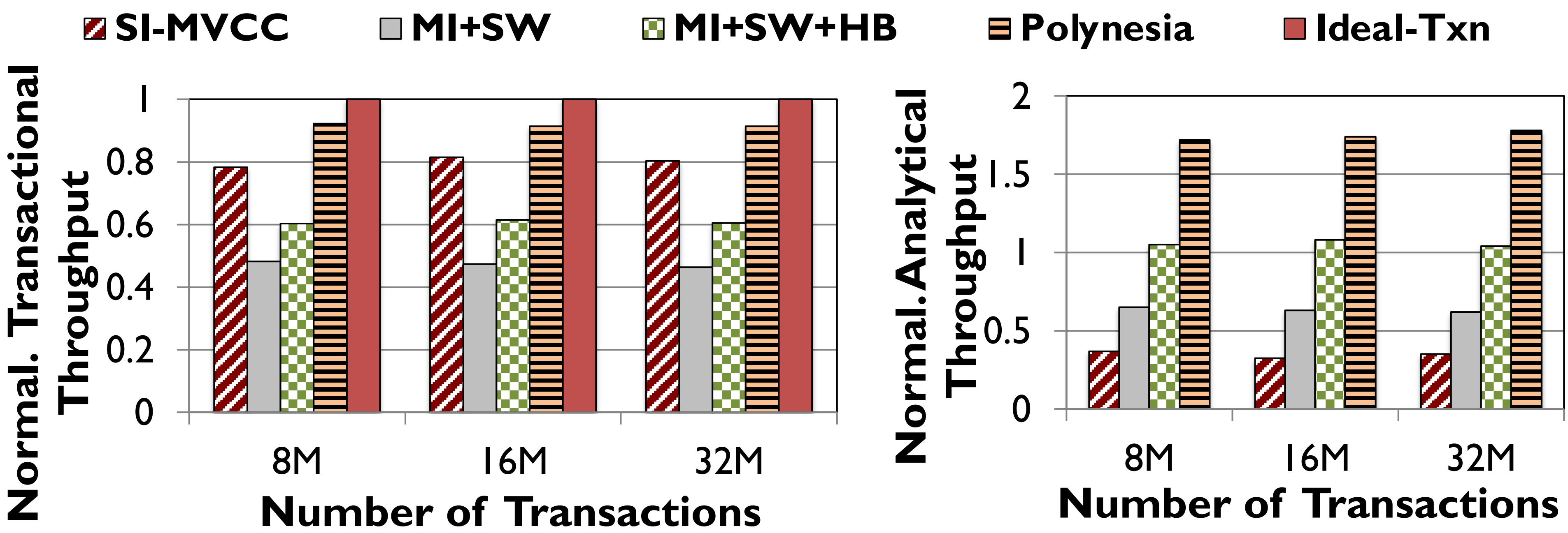
MI+SW+HB is the best software-only HTAP for analytical workloads, because it provides workload-specific optimizations, but it still loses 35.3% of the analytical throughput due to high main memory contention

End-to-End System Analysis (4/5)



Polynesia improves over MI+SW+HB by 63.8%, by eliminating data movement, and using custom logic for update propagation and consistency

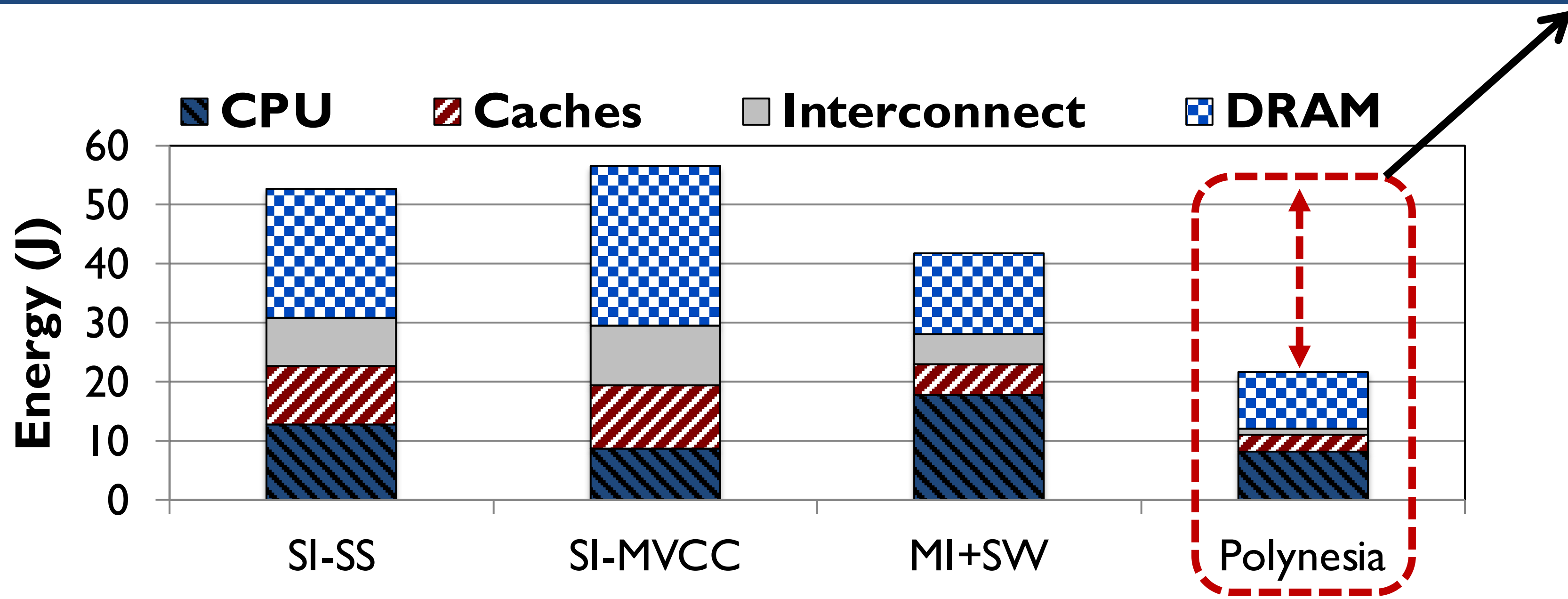
End-to-End System Analysis (5/5)



Overall, Polynesia achieves all three properties of HTAP system and has a higher transactional/analytical throughput (1.7x/3.74x) over prior HTAP systems

Energy Analysis

Polynesia consumes 0.4x/0.38x/0.5x the energy of SI-SS/SI-MVCC/MI+SW since Polynesia eliminates a large fraction (30%) of off-chip DRAM accesses



Polynesia is an energy-efficient HTAP system, reducing energy consumption by 48%, on average across prior works

More in the Paper

- Real workload analysis
- Effect of the update propagation technique
- Effect of the consistency mechanism
- Effect of the analytical engine
- Effect of the dataset size
- Area Analysis

More in the Paper

- Real workload analysis
- Effect of the update propagation technique

Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design

Amirali Boroumand[†] Saugata Ghose[◇] Geraldo F. Oliveira[‡] Onur Mutlu[‡]
[†]*Google* [◇]*Univ. of Illinois Urbana-Champaign* [‡]*ETH Zürich*

- Effect of the dataset size
- Area Analysis

Full Draft

Outline

Conclusion

- Context: Many applications need to perform real-time data analysis using an Hybrid Transactional/Analytical Processing (HTAP) system
  - An ideal HTAP system should have three properties: (1) data freshness and consistency, (2) workload-specific optimization, (3) performance isolation

- Problem: Prior works cannot achieve all properties of an ideal HTAP system

- Key Idea: Divide the system into transactional and analytical processing islands
  - Enables workload-specific optimizations and performance isolation

- Key Mechanism: Polynesia, a novel hardware/software cooperative design for in-memory HTAP databases
  - Implements custom algorithms and hardware to reduce the costs of data freshness and consistency
  - Exploits PIM for analytical processing to alleviate data movement

- Key Results: Polynesia outperforms three state-of-the-art HTAP systems
  - Average transactional/analytical throughput improvements of 1.7x/3.7x
  - 48% reduction on energy consumption

Polynesia:
Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design

P&S Processing-in-Memory
Spring 2022
16 June 2022

Full Draft

Amirali Boroumand
Geraldo F. Oliveira
Saugata Ghose
Onur Mutlu

SAFARI Google UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN ETH Zürich