Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages

Maciej Besta, Torsten Hoefler
LARGE-SCALE IRREGULAR GRAPH PROCESSING
LARGE-SCALE IRREGULAR GRAPH PROCESSING

- Becoming more important [1]

**LARGE-SCALE IRREGULAR GRAPH PROCESSING**

- Becoming more important [1]
  - Machine learning

LARGE-SCALE IRREGULAR GRAPH PROCESSING

- Becoming more important [1]
  - Machine learning
  - Computational science

LARGE-SCALE IRREGULAR GRAPH PROCESSING

- Becoming more important [1]
  - Machine learning
  - Computational science
  - Social network analysis

SYNCHRONIZATION MECHANISMS
COARSE LOCKS
SYNCHRONIZATION MECHANISMS
COARSE LOCKS
SYNCHRONIZATION MECHANISMS
COARSE LOCKS

An example graph
SYNCHRONIZATION MECHANISMS
COARSE LOCKS

An example graph

Proc p

lock

Proc q
SYNCHRONIZATION MECHANISMS
COARSE LOCKS

An example graph

Proc p

lock

accesses

...

Proc q
SYNCHRONIZATION MECHANISMS

COARSE LOCKS

An example graph

Proc p

\begin{align*}
\text{lock} \\
\text{accesses} \\
\ldots \\
\text{unlock}
\end{align*}

Proc q
SYNCHRONIZATION MECHANISMS
COARSE LOCKS

An example graph

Proc p

lock
accesses
unlock

Proc q

lock
accesses
SYNCHRONIZATION MECHANISMS
COARSE LOCKS

Simple protocols

An example graph
SYNCHRONIZATION MECHANISMS
COARSE LOCKS

Simple protocols

Serialization

An example graph

Proc p

lock

accesses

unlock

Proc q

lock

accesses

...
SYNCHRONIZATION MECHANISMS
COARSE LOCKS

Simple protocols

Serialization

Detrimental performance

An example graph

Proc p

lock

accesses

unlock

Proc q

lock

accesses
SYNCHRONIZATION MECHANISMS
FINE LOCKS
SYNCHRONIZATION MECHANISMS
FINE LOCKS

Complex access patterns 😊
SYNCHRONIZATION MECHANISMS
FINE LOCKS

Complex access patterns 😊
SYNCHRONIZATION MECHANISMS
FINE LOCKS

Higher performance possible

Complex access patterns 😊
SYNCHRONIZATION MECHANISMS
FINE LOCKS

Higher performance possible

Complex protocols

Complex access patterns 😊
**Synchronization Mechanisms**

**Fine locks**

- Higher performance possible
- Complex protocols
- Risk of deadlocks

![Diagram showing complex access patterns](attachment:image.png)
SYNCHRONIZATION MECHANISMS
ATOMIC OPERATIONS

Complex access patterns 😊
SYNCHRONIZATION MECHANISMS
ATOMIC OPERATIONS

High performance (may be challenging to get)

Complex access patterns 😊
SYNCHRONIZATION MECHANISMS
ATOMIC OPERATIONS

High performance (may be challenging to get)

Complex protocols

Complex access patterns 😊
SYNCHRONIZATION MECHANISMS

ATOMIC OPERATIONS

High performance (may be challenging to get)

Complex protocols

Subtle issues (ABA, ...)

Complex access patterns 😊
SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

**SYNCHRONIZATION MECHANISMS**

**SOFTWARE TRANSACTIONAL MEMORY (STM) [1]**

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

Conflicts solved with rollbacks and/or serialization.

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

Conflicts solved with rollbacks and/or serialization.

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

Conflicts solved with rollbacks and/or serialization.

SYNCHRONIZATION MECHANISMS
SOFTWARE TRANSACTIONAL MEMORY (STM) [1]

Conflicts solved with rollbacks and/or serialization.

**Synchronization Mechanisms**

**Software Transactional Memory (STM)** [1]

Conflicts solved with rollbacks and/or serialization.

---

SYNCHRONIZATION MECHANISMS

HARDWARE TRANSACTIONAL MEMORY (HTM)

Conflicts solved with rollbacks and/or HW serialization.
SYNCHRONIZATION MECHANISMS
HARDWARE TRANSACTIONAL MEMORY (HTM)

Conflicts solved with rollbacks and/or HW serialization.

Simple protocols
**Synchronization Mechanisms**

**Hardware Transactional Memory** (HTM)

Conflicts solved with rollbacks and/or HW serialization.

High performance? For graphs?

Simple protocols
HARDWARE TRANSACTIONAL MEMORY
Hardware Transactional Memory

AZUL Systems®

Vega
HARDWARE TRANSACTIONAL MEMORY

Rock

AZUL SYSTEMS

Vega

Sun Microsystems
HARDWARE TRANSACTIONAL MEMORY

AZUL SYSTEMS®

Vega

Rock

BlueGene/Q

POWER8
HARDWARE TRANSACTIONAL MEMORY

Azul Systems

Vega

IBM

BlueGene/Q

Rock

Power8

Haswell

Intel

4th Gen Intel Core i7
HARDWARE TRANSACTIONAL MEMORY

- AZUL SYSTEMS
- Vega

- IBM
- BlueGene/Q

- Rock

- Sun Microsystems

- Haswell

- intel
  - 4th Gen Intel® Core™ i7

- POWER8
They offer programmability, how about performance?
**SHARED- & DISTRIBUTED-MEMORY MACHINES**

Diagram showing connections between Proc p and Proc q with "start transaction" labels.
SHARED- & DISTRIBUTED-MEMORY MACHINES

- HTM works fine for single shared-memory domains
**SHARED- & DISTRIBUTED-MEMORY MACHINES**

- HTM works fine for single shared-memory domains
  - Most graphs fit in such machines [1]

---

**SHARED- & DISTRIBUTED-MEMORY MACHINES**

- HTM works fine for single shared-memory domains
  - Most graphs fit in such machines [1]

SHARED- & DISTRIBUTED-MEMORY MACHINES

- HTM works fine for single shared-memory domains
  - Most graphs fit in such machines [1]
- However, some do not:

**Shared- & Distributed-memory Machines**

- HTM works fine for single shared-memory domains
  - Most graphs fit in such machines [1]
- However, some do not:
  - Very large instances

---

**SHARED- & DISTRIBUTED-MEMORY MACHINES**

- HTM works fine for single shared-memory domains
  - Most graphs fit in such machines [1]
- However, some do not:
  - Very large instances
  - Rich vertex/edge data

---

SHARED- & DISTRIBUTED-MEMORY MACHINES

- HTM works fine for single shared-memory domains
  - Most graphs fit in such machines [1]
- However, some do not:
  - Very large instances
  - Rich vertex/edge data
- Fat nodes with lots of RAM still expensive ($35K for a machine with 1TB of RAM [1])

**SHARED- & DISTRIBUTED-MEMORY MACHINES**

- HTM works fine for single shared-memory domains
  - Most graphs fit in such machines [1]
- However, some do not:
  - Very large instances
  - Rich vertex/edge data
- Fat nodes with lots of RAM still expensive ($35K for a machine with 1TB of RAM [1])

How to apply HTM in such a setting?

OVERVIEW OF OUR RESEARCH
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages
= Atomic Active Messages
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages = Atomic Active Messages

Coarsening & coalescing
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

Coarsening & coalescing

HTM + Active Messages = Atomic Active Messages

Performance Modeling & Analysis
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages = Atomic Active Messages

Coarsening & coalescing

Performance Modeling & Analysis

Haswell & BG/Q Analysis
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages = Atomic Active Messages

Coarsening & coalescing

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Performance model
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages = Atomic Active Messages

Coarsening & coalescing

Evaluation

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Performance model
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages = Atomic Active Messages

Coarsening & coalescing

Evaluation

Considered engines and graphs

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Performance model
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

Coarsening & coalescing

HTM + Active Messages = Atomic Active Messages

Evaluation

Considered engines and graphs

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Performance model

Accelerating state-of-the-art
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

Coarsening & coalescing

HTM + Active Messages = Atomic Active Messages

Evaluation

Considered engines and graphs

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Performance model

Accelerating state-of-the-art

Scalability
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

Coarsening & coalescing

HTM + Active Messages = Atomic Active Messages

Evaluation

Haswell & BG/Q Analysis

Performance Modeling & Analysis

Accelerating state-of-the-art

Scalability
ACTIVE MESSAGES (AM)
ACTIVE MESSAGES (AM)
ACTIVE MESSAGES (AM)
ACTIVE MESSAGES (AM)
ACTIVE MESSAGES (AM)
ACTIVE MESSAGES (AM)

Process p

Active message

Process q

Memory

A’s addr: Handler A

Z’s addr: Handler Z
ACTIVE MESSAGES (AM)

Process p

Active message

Z’s addr

Process q

Memory

A’s addr:  Handler A

...  

Z’s addr:  Handler Z
ACTIVE MESSAGES (AM)

Process p

Active message
Z’s addr
Payload

Process q

Memory

A’s addr: Handler A

Z’s addr: Handler Z
ACTIVE MESSAGES (AM)

Process p

Active message
Z’s addr | Payload

Process q

Memory
A’s addr: Handler A
... 
Z’s addr: Handler Z
ACTIVE MESSAGES (AM)

Process p

Active message
Z’s addr Payload

Process q

Memory

A’s addr: Handler A

Z’s addr: Handler Z

IBM
ACTIVE MESSAGES (AM)


AM + HTM = ...
AM + HTM = ...
AM + HTM = ...

AM handlers run as HTM transactions

Node A

Proc p

Node B

Proc q

start transaction

start transaction
AM + HTM = ATOMIC ACTIVE MESSAGES

Node A
Proc p

Node B
Proc q

AM handlers run as HTM transactions

start transaction

start transaction
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS
ACCESSING MULTIPLE VERTICES ATOMICALLY
Example: BFS

Size (the number of vertices) must be appropriate to minimize overheads from both commits and rollbacks
TRANSFERRING TRANSACTIONS ACROSS NODES
TRANSFERRING TRANSACTIONS ACROSS NODES
TRANSFERRING TRANSACTIONS ACROSS NODES
TRANSFERRING TRANSACTIONS ACROSS NODES
Transactions must be appropriately coalesced to minimize communication overheads.
Transactions must be appropriately coalesced to minimize communication overheads.
EXECUTING TRANSACTIONS ON MULTIPLE NODES
EXECUTING TRANSACTIONS ON MULTIPLE NODES

Node A

Node B
EXECUTING TRANSACTIONS ON MULTIPLE NODES
EXECUTING TRANSACTIONS ON MULTIPLE NODES
EXECUTING TRANSACTIONS ON MULTIPLE NODES

Vertices must be appropriately relocated to enable execution of a hardware transaction.
EXECUTING TRANSACTIONS ON MULTIPLE NODES

Vertices must be appropriately relocated to enable execution of a hardware transaction.
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

Coarsening & coalescing

HTM + Active Messages = Atomic Active Messages

Performance Modeling & Analysis

Evaluation

Considered engines and graphs

Accelerating state-of-the-art

Scalability
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages = Atomic Active Messages

Evaluation

Considered engines and graphs

Acceleration state-of-the-art

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Coarsening & coalescing

Performance model

Scalability

HTM for graphs in SM & DM environments

Coarsening & coalescing

HTM + Active Messages = Atomic Active Messages

Evaluation

Considered engines and graphs

Acceleration state-of-the-art

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Coarsening & coalescing

Performance model

Scalability
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages = Atomic Active Messages

Evaluation

Coarsening & coalescing

Considered engines and graphs

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Accelerating state-of-the-art

Scalability
PERFORMANCE ANALYSIS
RESEARCH QUESTIONS
How can we implement AAM handlers to run most efficiently?
PERFORMANCE ANALYSIS RESEARCH QUESTIONS

What are performance tradeoffs related to HTM?

How can we implement AAM handlers to run most efficiently?
PERFORMANCE ANALYSIS RESEARCH QUESTIONS

- How can we implement AAM handlers to run most efficiently?
- What are performance tradeoffs related to HTM?
- What are advantages of HTM over atomics for AAM?
**Performance Analysis Research Questions**

- **What are advantages of HTM over atomics for AAM?**
- **How can we implement AAM handlers to run most efficiently?**
- **What are performance tradeoffs related to HTM?**
- **What are the optimal transaction sizes? Can we amortize transaction overheads?**
PERFORMANCE ANALYSIS
TYPES OF MACHINES
PERFORMANCE ANALYSIS

TYPES OF MACHINES

- Evaluation on 3 machines
PERFORMANCE ANALYSIS

TYPES OF MACHINES

- Evaluation on 3 machines
  - Intel Haswell server
PERFORMANCE ANALYSIS

TYPES OF MACHINES

- Evaluation on 3 machines
  - Intel Haswell server
  - InfiniBand cluster

Commodity machines

HPC clusters
Performance Analysis
Types of Machines

- Evaluation on 3 machines
  - Intel Haswell server
  - InfiniBand cluster
  - IBM BlueGene/Q

Commodity machines

Supercomputing machines

HPC clusters
PERFORMANCE ANALYSIS

CONSIDERED MECHANISMS
PERFORMANCE ANALYSIS

CONSIDERED MECHANISMS

Haswell HTM
Haswell HTM

Deployed in L1
PERFORMANCE ANALYSIS
CONSIDERED MECHANISMS

Haswell HTM

- 32KB per core
- Deployed in L1

L1 L1
PERFORMANCE ANALYSIS

CONSIDERED MECHANISMS

Haswell HTM

- 32KB per core
- Deployed in L1
- 8-way associative
PERFORMANCE ANALYSIS CONSIDERED MECHANISMS

Haswell HTM

- 32KB per core
- Deployed in L1
- 8-way associative

RTM (Restricted Transactional Memory)

HLE (Hardware Lock Elision)
PERFORMANCE ANALYSIS
CONSIDERED MECHANISMS

Haswell HTM

- Intel
- 32KB per core
- Deployed in L1
- 8-way associative

RTM (Restricted Transactional Memory)

BlueGene/Q HTM

IBM

HLE (Hardware Lock Elision)
PERFORMANCE ANALYSIS
CONSIDERED MECHANISMS

Haswell HTM

- 32KB per core
- Deployed in L1
- 8-way associative
- RTM (Restricted Transactional Memory)
- HLE (Hardware Lock Elision)

BlueGene/Q HTM

- Deployed in L2
**PERFORMANCE ANALYSIS**

**CONSIDERED MECHANISMS**

**Haswell HTM**
- Deployment: L1
- Size: 32KB per core
- Way: 8-way associative
- Features:
  - RTM (Restricted Transactional Memory)
  - HLE (Hardware Lock Elision)

**BlueGene/Q HTM**
- Deployment: L2
- Size: 2MB per core
- Features:
  - L1
  - L1

---

ETH Zurich

@spcl_eth
PERFORMANCE ANALYSIS
CONSIDERED MECHANISMS

Haswell HTM

- 32KB per core
- Deployed in L1
- 8-way associative

RTM (Restricted Transactional Memory)
HLE (Hardware Lock Elision)

BlueGene/Q HTM

- 2MB per core
- Deployed in L2
- 16-way associative
PERFORMANCE ANALYSIS
CONSIDERED MECHANISMS

Haswell HTM
- Intel
- 32KB per core
- Deployed in L1
- 8-way associative
- RTM (Restricted Transactional Memory)
- HLE (Hardware Lock Elision)

BlueGene/Q HTM
- IBM
- 2MB per core
- Deployed in L2
- 16-way associative
- The Long Running Mode
PERFORMANCE ANALYSIS
CONSIDERED MECHANISMS

Haswell HTM
- 32KB per core
- Deployed in L1
- 8-way associative

RTM (Restricted Transactional Memory)
HLE (Hardware Lock Elision)

BlueGene/Q HTM
- 2MB per core
- Deployed in L2
- 16-way associative

The Long Running Mode
The Short Running Mode
**Performance Analysis Considered Mechanisms**

**Haswell HTM**
- **L1 L1**
- 32KB per core
- Deployed in L1
- 8-way associative
- RTM (Restricted Transactional Memory)
- HLE (Hardware Lock Elision)

**BlueGene/Q HTM**
- **L1 L1**
- 2MB per core
- Deployed in L2
- 16-way associative
- The Long Running Mode
- The Short Running Mode
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Used in BFS, SSSP, ...

// start handler
if(!v.visited) {
    v.visited = 1;
}
// finish handler
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Lower contention
(10 racing accesses/vertex)

// start handler
if(!v.visited) {
    v.visited = 1;
}
// finish handler

Used in BFS, SSSP, ...

![Graph showing total time vs. threads per node (T)]
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Lower contention
(10 racing accesses/vertex)

// start handler
if(!v.visited) {
  v.visited = 1;
}
// finish handler

Atoms (CAS) slightly faster than HTM

Used in BFS, SSSP, ...
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Lower contention
(10 racing accesses/vertex)

// start handler
if(!v.visited) {
  v.visited = 1;
}
// finish handler

Atomics (CAS) slightly faster than HTM

Commit overheads dominate

Used in BFS, SSSP, ...
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Lower contention
(10 racing accesses/vertex)

// start handler
if (!v.visited) {
    v.visited = 1;
}
// finish handler

Atomics (CAS) slightly faster than HTM

Commit overheads dominate
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Lower contention
(10 racing accesses/vertex)

Very few aborts

Atomics (CAS) slightly faster than HTM

Commit overheads dominate

Used in BFS, SSSP, ...

// start handler
if(!v.visited) {
    v.visited = 1;
}
// finish handler
**SINGLE-VERTEX TRANSACTIONS**

**MARKING A VERTEX AS VISITED**

- Very few aborts
- Lower contention
  - (10 racing accesses/vertex)

```
// start handler
if(!v.visited) {
  v.visited = 1;
}
// finish handler
```

- Atomics (CAS) slightly faster than HTM
- Commit overheads dominate

**Graph:**
- BG/Q HTM (long mode)
- BG/Q HTM (short mode)
- BG/Q atomics
- Intel RTM / HLE
- Intel atomics

**Numbers are total aborts per data point**
- 9
- 7
- 0
- 1
- 1.1k
- 0.8k

**Graph axes:**
- Total time [ms]
- Threads per node (T)
  - 1, 2, 4, 8, 16, 32

**Used in BFS, SSSP, ...**
**Single-Vertex Transactions**

**Marking a Vertex as Visited**

Higher contention

(100 racing accesses/vertex)

```c
// start handler
if(!v.visited) {
    v.visited = 1;
}
// finish handler
```

- **Intel atomics**
- **Intel RTM**
- **BG/Q HTM** (short mode)
- **BG/Q HTM** (long mode)
- **BG/Q atomics**
- **Intel HLE**

![Graph](image-url)
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Higher contention
(100 racing accesses/vertex)

// start handler
if(!v.visited) {
  v.visited = 1;
}
// finish handler

RTM better than atomics
**Single-Vertex Transactions**

**Marking a Vertex as Visited**

Higher contention (100 racing accesses/vertex)

```
// start handler
if(!v.visited) {
    v.visited = 1;
}
// finish handler
```

- **BG/Q HTM** better than atomics
- **BG/Q** HTM still worse (L1 vs L2 matters!)
- **RTM** better than atomics

- Intel atomics
- Intel RTM
- Intel HLE
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Used in BFS, SSSP, ...

Still very few aborts

BG/Q HTM still worse (L1 vs L2 matters!)

RTM better than atomics

Higher contention
(100 racing accesses/vertex)

// start handler
if(!v.visited) {
    v.visited = 1;
}
// finish handler
**SINGLE-VERTEX TRANSACTIONS**

MARKING A VERTEX AS VISITED

Still very few aborts

Used in BFS, SSSP, ...

BG/Q HTM still worse (L1 vs L2 matters!)

RTM better than atomics

// start handler
if(!v.visited) {
  v.visited = 1;
}
// finish handler

Higher contention
(100 racing accesses/vertex)

![Graph showing performance of different transaction types](chart.png)

- **BG/Q HTM**: (short mode), (long mode)
- **BG/Q atomics**
- **Intel HLE**
- **Intel RTM**

Numbers are total aborts per data point.
SINGLE-VERTEX TRANSACTIONS
INCREMENTING VERTEX RANK

// start handler
v.rank++;
// finish handler

Used in PageRank
SINGLE-VERTEX TRANSACTIONS
INCREMENTING VERTEX RANK

// start handler
v.rank++;
// finish handler

Atomics always outperform HTM

Used in PageRank
**SINGLE-VERTEX TRANSACTIONS**
**INCREMENTING VERTEX RANK**

- Atomics always outperform HTM

The reason: each transaction always modifies some memory cell, increasing the number of conflicts

// start handler
v.rank++;
// finish handler
PERFORMANCE MODEL
ATOMICS VS TRANSACTIONS
Performance Model
Atomics vs Transactions

Time to modify $N$ vertices with atomics:

$$T_{AT}(N) = A_{AT} N + B_{AT}$$
**Performance Model**

**Atomics vs Transactions**

Time to modify $N$ vertices with atomics:

$$T_{AT}(N) = A_{AT}N + B_{AT}$$

Startup overheads
**Performance Model**

**Atomics vs Transactions**

Time to modify $N$ vertices with atomics:

$$T_{AT}(N) = A_{AT}N + B_{AT}$$

- **Overhead per vertex**
- **Startup overheads**
**Performance Model**

**Atomics vs Transactions**

Time to modify $N$ vertices with atomics:

$$T_{AT}(N) = A_{AT}N + B_{AT}$$

Overhead per vertex

Startup overheads

Time to modify $N$ vertices with a transaction

$$T_{HTM}(N) = A_{HTM}N + B_{HTM}$$
PERFORMANCE MODEL
ATOMICS VS TRANSACTIONS

Time to modify $N$ vertices with atomics:

$$T_{AT}(N) = A_{AT}N + B_{AT}$$

Time to modify $N$ vertices with a transaction

$$T_{HTM}(N) = A_{HTM}N + B_{HTM}$$

Overhead per vertex

Startup overheads

Overhead per vertex

Startup overheads
PERFORMANCE MODEL
ATOMICS VS TRANSACTIONS

Time to modify $N$ vertices with atomics:

$$T_{AT}(N) = A_{AT}N + B_{AT}$$

Time to modify $N$ vertices with a transaction

$$T_{HTM}(N) = A_{HTM}N + B_{HTM}$$

We predict that:

$$B_{AT} < B_{HTM}$$
$$A_{AT} > A_{HTM}$$

Overhead per vertex

Startup overheads

Overhead per vertex

Startup overheads
**Performance Model**

**Atoms vs Transactions**

Time to modify $N$ vertices with atomics:

$$T_{AT}(N) = A_{AT}N + B_{AT}$$

Overhead per vertex

Startup overheads

Time to modify $N$ vertices with a transaction

$$T_{HTM}(N) = A_{HTM}N + B_{HTM}$$

Overhead per vertex

Startup overheads

We predict that:

$$B_{AT} < B_{HTM}$$

$$A_{AT} > A_{HTM}$$

Transaction startup overheads dominate
**Performance Model**

**Atomics vs Transactions**

Time to modify $N$ vertices with atomics:

$$T_{AT}(N) = A_{AT}N + B_{AT}$$

Time to modify $N$ vertices with a transaction

$$T_{HTM}(N) = A_{HTM}N + B_{HTM}$$

Overhead per vertex

Startup overheads

Overhead per vertex

Startup overheads

We predict that:

$$B_{AT} < B_{HTM}$$

$$A_{AT} > A_{HTM}$$

Transactions’ cost grows slower

Transaction startup overheads dominate
PERFORMANCE MODEL
ATOMICS VS TRANSACTIONS
Performance Model

Atomics vs Transactions

![Graph showing performance comparison between Atomics and Transactions](image)

- **Mechanism:**
  - RTM-CAS
  - CAS

- **Total time [us]** vs **Accessed vertices**

- **Model**
  - The long mode results in higher latency than the short mode
Indeed:

\[ B_{AT} < B_{HTM} \]

\[ A_{AT} > A_{HTM} \]
**Performance Model**

**Atomics vs Transactions**

- Can we amortize HTM startup/commit overheads with larger transaction sizes?

Indeed:

\[
B_{AT} < B_{HTM}
\]

\[
A_{AT} > A_{HTM}
\]
**Performance Model**

**Atomics vs Transactions**

- Can we amortize HTM startup/commit overheads with larger transaction sizes?

Indeed:

\[ B_{AT} < B_{HTM} \]

\[ A_{AT} > A_{HTM} \]

Yes, we can!
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

![Graph showing BGQ mechanism with HTM-Long-Mode and HTM-Short-Mode compared to Atomic CAS for different transaction sizes. The graph plots total time against transaction size (M) in vertices. The BGQ mechanism shows a curve that increases with transaction size, with HTM-Long-Mode starting higher and HTM-Short-Mode starting lower. The Atomic CAS line is a horizontal line below the BGQ mechanism, indicating a lower total time.]
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Startup and commit overheads
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Startup and commit overheads

Abort and rollback overheads
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

- Startup and commit overheads
- Abort and rollback overheads
- The sweetspot! (144 vertices)
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

![Graph showing the relationship between transaction size and total time for different mechanisms. The graph compares HTM-HLE and HTM-RTM with Has-C mechanism. The x-axis represents transaction size (M [vertices]), and the y-axis represents total time [s]. The graph includes markers indicating the percentage of time for different transaction sizes: 77%, 79%, 84%, 96%, 98%, 2.2%, 5.8%, and 2.3%. The y-axis ranges from 0.08 to 0.14.]
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Startup and commit overheads
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Startup and commit overheads

Abort and rollback overheads
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Startup and commit overheads

Abort and rollback overheads

The sweetspot! (2 vertices)
MULTI-VERTEX TRANSACTIONS MARKING VERTICES AS VISITED

Numbers: % of aborts due to HTM capacity overflows

Abort and rollback overheads

Startup and commit overheads

The sweetspot! (2 vertices)
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Abort and rollback overheads

Numbers: % of aborts due to HTM capacity overflows

Majority of aborts are due to HTM capacity overflows (small cache size & associativity)

The sweetspot! (2 vertices)

Startup and commit overheads

Has-C mechanism
- HTM-HLE
- HTM-RTM

84% 96%
79%
77%

2.2% 5.8%
2.3%

0.08
0.10
0.12
0.14
Total time [s]

Transaction size (M) [vertices]

Atomic CAS
PERFORMANCE ANALYSIS
QUESTIONS ANSWERED

How can we implement AAM handlers most effectively?

What are the advantages of HTM over atomics for AAM?

What are the optimal transaction sizes? Can we amortize transaction overheads?

What are performance tradeoffs related to HTM?
What are advantages of HTM over atomics for AAM?

“It really depends” 😊. But... There are some regularities.

What are performance tradeoffs related to HTM?

What are the optimal transaction sizes? Can we amortize transaction overheads?
PERFORMANCE ANALYSIS

QUESTIONS ANSWERED

What are the optimal transaction sizes?

Can we amortize transaction overheads?

What are performance tradeoffs related to HTM?

„It really depends” 😊. But... There are some regularities

For some algorithms (BFS) HTM is better

“It really depends” 😊.
PERFORMANCE ANALYSIS
QUESTIONS ANSWERED

What are the optimal transaction sizes?

What are performance tradeoffs related to HTM?

What are the optimal transaction sizes? Can we amortize transaction overheads?

„It really depends” 😊. But... There are some regularities

For some algorithms (BFS) HTM is better

For others (PageRank) atomics give more performance
PERFORMANCE ANALYSIS

QUESTIONS ANSWERED

“May fail”
For some algorithms (BFS) HTM is better

“It really depends” 😊. But... There are some regularities

What are the optimal transaction sizes?
Can we amortize transaction overheads?

What are performance tradeoffs related to HTM?

For others (PageRank) atomics give more performance

“It really depends” 😊. But... There are some regularities
**Performance Analysis Questions Answered**

- What are the optimal transaction sizes?
  - "It really depends" 😊. But... There are some regularities

- Can we amortize transaction overheads?
  - "May fail"

- What are performance tradeoffs related to HTM?
  - "Always succeed"

- What are the optimal transaction sizes? Can we amortize transaction overheads?
  - For some algorithms (BFS) HTM is better
  - For others (PageRank) atomics give more performance
**PERFORMANCE ANALYSIS QUESTIONS ANSWERED**

What are the optimal transaction sizes?

What are performance tradeoffs related to HTM?

What are the optimal transaction sizes? Can we amortize transaction overheads?

„It really depends“ 😊. But... There are some regularities

„May fail“

For some algorithms (BFS) HTM is better

„Always succeed“

For others (PageRank) atomics give more performance

AAM establishes a whole hierarchy of algorithms; check the paper 😊
Performance Analysis Questions Answered

"It really depends" 😊. But... There are some regularities.

What are performance tradeoffs related to HTM?

"May fail"
For some algorithms (BFS) HTM is better.

"Always succeed"
For others (PageRank) atomics give more performance.

AAM establishes a whole hierarchy of algorithms; check the paper 😊.

Size for BG/Q ~100
Size for Haswell ~10

Yes, we can.
**Performance Analysis Questions Answered**

- **What are performance tradeoffs related to HTM?**
  - "It really depends" 😊. But... There are some regularities

- **For some algorithms (BFS) HTM is better**
  - "May fail"

- **For others (PageRank) atomics give more performance**
  - "Always succeed"

- **AAM establishes a whole hierarchy of algorithms; check the paper 😊**

- **Size for BG/Q ~100 > Size for Haswell ~10**
  - Same for other graphs

- **Yes, we can**
**Performance Analysis Questions Answered**

- „It really depends” 😊. But... There are some regularities

- For some algorithms (BFS) HTM is better
- „May fail"

- For others (PageRank) atomics give more performance
- „Always succeed”

- AAM establishes a whole hierarchy of algorithms; check the paper 😊

- Larger cache & associativity → fewer aborts & more coarsening

- Size for BG/Q ~100
- Size for Haswell ~10

- Same for other graphs

- Yes, we can
**Performance Analysis Questions Answered**

- „It really depends“ 😊. But... There are some regularities.
- Larger cache & associativity → fewer aborts & more coarsening.
- Size for BG/Q ~100
  > Size for Haswell ~10
- Larger (L2) cache → higher latency.
- „May fail“
- For some algorithms (BFS) HTM is better.
- „Always succeed”
- For others (PageRank) atomics give more performance.
- AAM establishes a whole hierarchy of algorithms; check the paper 😊.
- Yes, we can.
- Same for other graphs.
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages = Atomic Active Messages

Coarsening & coalescing

Evaluation

Considered engines and graphs

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Performance model

Accelerating state-of-the-art

Scalability
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

HTM + Active Messages = Atomic Active Messages

Coarsening & coalescing

Evaluation

Haswell & BG/Q Analysis

Accelerating state-of-the-art

Scalability

Performance Modeling & Analysis

Considered engines and graphs
OVERVIEW OF OUR RESEARCH

HTM for graphs in SM & DM environments

Coarsening & coalescing

HTM + Active Messages = Atomic Active Messages

Performance Modeling & Analysis

Haswell & BG/Q Analysis

Evaluation

Considered engines and graphs

Accelerating state-of-the-art

Scalability
EVALUATION

CONSIDERED ENGINES
EVALUATION

CONSIDERED ENGINES


EVALUATION

CONSIDERED ENGINES


AAM + Improving Graph500 design

EVALUATION

CONSIDERED ENGINES


[2] Runtimes that exploit amorphous data-parallelism

AAM +

Improving Graph500 design

EVALUATION
CONSIDERED ENGINES


AAM + Improving Graph500 design

[2] Runtimes that exploit amorphous data-parallelism

[3] Hadoop-based BSP engines

EVALUATION
CONSIDERED ENGINES

PBGL [4]
Distributed HPC libraries


[2] Runtimes that exploit amorphous data-parallelism

AAM +
Improving Graph500 design

HAMA [3]
Hadoop-based BSP engines

EVALUATION

CONSIDERED TYPES OF GRAPHS
EVALUATION

CONSIDERED TYPES OF GRAPHS

Synthetic graphs
EVALUATION

CONSIDERED TYPES OF GRAPHS

Synthetic graphs

Kronecker [1]

EVALUATION

CONSIDERED TYPES OF GRAPHS

Synthetic graphs

Kronecker [1]

Erdös-Rényi [2]

EVALUATION
CONSIDERED TYPES OF GRAPHS

Real-world SNAP graphs [3]

Synthetic graphs

Kronecker [1]

Erdős-Rényi [2]

EVALUATION
CONSIDERED TYPES OF GRAPHS

Synthetic graphs

Kronecker [1]

Erdös-Rényi [2]

Real-world SNAP graphs [3]

Social networks

Road networks

Comm. graphs

Citation graphs

Web graphs

Purchase networks

ACCELERATING STATE-OF-THE-ART
GRAPH500 + AAM (BLUEGENE/Q)  IBM
ACCELERATING STATE-OF-THE-ART GRAPH500 + AAM (BLUEGENE/Q) IBM
ACCELERATING STATE-OF-THE-ART
GRAPH500 + AAM (BlueGene/Q)

Numbers are speedups of AAM over
Graph500 for a given data point

Implementation
- Graph500–BGQ
- AAM–BGQ
ACCELERATING STATE-OF-THE-ART
GRAPH500 + AAM (BlueGene/Q) IBM

Total time [s]

Numbers are speedups of AAM over Graph500 for a given data point

Implementation
- Graph500-BGQ
- AAM-BGQ
ACCELERATING STATE-OF-THE-ART
GRAPH500 + AAM (BLUEGENE/Q) IBM

Fill the whole memory

Numbers are speedups of AAM over Graph500 for a given data point

Implementation
- Graph500–BGQ
- AAM–BGQ

Total time [s]

Edges per vertex ($\bar{d}$)
ACCELERATING STATE-OF-THE-ART GRAPH500 + AAM (BlueGene/Q) IBM

Fill the whole memory

Numbers are speedups of AAM over Graph500 for a given data point

Total time [s]

Edges per vertex ($\bar{d}$)

Implementation
- Graph500-BGQ
- AAM-BGQ
ACCELERATING STATE-OF-THE-ART
GRAPH500 + AAM (HASWELL)

Numbers are speedups of AAM over Graph500 for a given data point

Implementation
- Graph500–Haswell
- AAM–Haswell
ACCELERATING STATE-OF-THE-ART
GRAPH500 + AAM (HASWELL)

Numbers are speedups of AAM over
Graph500 for a given data point

Implementation
- Graph500–Haswell
- AAM–Haswell

Edges per vertex ($\bar{d}$)
ACCELERATING STATE-OF-THE-ART GRAPH500 + AAM (HASWELL)

Fill the whole memory

Numbers are speedups of AAM over Graph500 for a given data point

Implementation
- Graph500-Haswell
- AAM-Haswell

Edges per vertex ($\bar{d}$)

Graphs with 8M vertices
- 1.09
- 1.12
- 1.15
- 1.15

Graphs with 2M vertices
- 1.03
- 1.05
- 1.09
- 1.14

Graph with 64M vertices
- 1.23

Total time [s]
ACCELERATING STATE-OF-THE-ART
GRAPH500 + AAM (HASWELL)

Numbers are speedups of AAM over Graph500 for a given data point

Implementation
- Graph500–Haswell
- AAM–Haswell

Fill the whole memory

Total time [s]

Edges per vertex (\(\bar{d}\))
OUTPERFORMING STATE-OF-THE-ART
# Outperforming State-of-the-Art

| Type                          | ID   | Name                | $|V|$ | $|E|$ | $S$ over g500 ($M = 24$) | $S$ over g500 ($M = 2$) | $S$ over Galois ($M = 2$) | $M$ | $S$ over g500 | $S$ over Galois | $S$ over HAMA |
|------------------------------|------|---------------------|-----|-----|--------------------------|--------------------------|--------------------------|-----|----------------|----------------|---------------|
| Comm. networks (CNs)         | cWT  | wiki-Talk           | 2.4M| 5M  | 2.82                     | 3.35                     | 0.91                     | 24 | 0.96          | 1.28           | 344           |
|                             | cEU  | email-EuAll         | 265k| 420k| 3.67                     | 4.36                     | 0.76                     | 32 | 0.97          | 1.12           | 1448          |
| Social networks (SNs)        | sLV  | soc-LiveJ.          | 4.8M| 69M | 1.44                     | 1.56                     | 1.05                     | 12 | 1.07          | 1.12           | $> 10^4$       |
|                             | sQR  | com-orkut           | 3M  | 117M| 1.22                     | 1.27                     | 1.06                     | 20 | 1.13          | 0.74           | $> 10^4$       |
|                             | sLJ  | com-lj              | 4M  | 34M | 1.44                     | 1.54                     | 1.03                     | 12 | 1.04          | 1.04           | 603           |
|                             | sYT  | com-youtube         | 1.1M| 2.9M| 1.67                     | 1.84                     | 0.96                     | 8  | 0.98          | 1.11           | 670           |
|                             | sDB  | com-dblp            | 317k| 1M  | 1.33                     | 1.80                     | $\approx 1$              | 8  | 2.5           | $\approx 1$    | 2160          |
|                             | sAM  | com-amazon          | 334k| 925k| 1.14                     | 1.62                     | 1.04                     | 8  | 2.5           | $\approx 1$    | 1426          |
| Purchase network (PNs)       | pAM  | amazon0601          | 403k| 3.3M| 1.45                     | 1.91                     | $\approx 1$              | 8  | 1.03          | 1.30           | 618           |
| Road networks (RNs)          | rCA  | roadNet-CAC         | 1.9M| 5.5M| $\approx 1$              | 1.59                     | 1.33                     | 2  | 1.38          | 1.80           | $> 10^4$       |
|                             | rTX  | roadNet-TX          | 1.3M| 3.8M| $\approx 1$              | 1.53                     | 1.29                     | 2  | 1.42          | 2.08           | $> 10^4$       |
|                             | rPA  | roadNet-PA          | 1M  | 3M  | $\approx 1$              | 1.52                     | $\approx 1$              | 2  | 1.07          | 2.16           | $> 10^4$       |
| Citation graphs (CGs)        | ciP  | cit-Patents         | 3.7M| 16.5M| 1.16                     | 1.57                     | 1.01                     | 8  | 1.01          | 1.26           | 1875          |
| Web graphs (WGs)             | wGL  | web-Google          | 875k| 5.1M| 1.78                     | 2.08                     | 0.98                     | 12 | 1.06          | 1.35           | 365           |
|                             | wBS  | web-BerkStan        | 685k| 7.6M| 1.91                     | 1.91                     | 0.93                     | 24 | 1.07          | 1.40           | 755           |
|                             | wSF  | web-Stanford        | 281k| 2.3M| 1.89                     | 1.89                     | 0.98                     | 24 | 1.07          | 1.58           | 1077          |
### Outperforming State-of-the-Art

![SNAP Logo]

😊 No, you don’t have to read it. All details are in the paper. Here: just a summary.

<table>
<thead>
<tr>
<th>Input graph properties</th>
<th>BG/Q analysis</th>
<th>Haswell analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S$ over $g500$ ($M = 24$)</td>
<td>$S$ over $g500$ ($M = 2$)</td>
</tr>
<tr>
<td>Type</td>
<td>$V$</td>
<td>$E$</td>
</tr>
<tr>
<td>Country networks (Os)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Republic</td>
<td>cWF</td>
<td>wiki-Talk</td>
</tr>
<tr>
<td></td>
<td>eEU</td>
<td>email-EuAll</td>
</tr>
<tr>
<td></td>
<td>543k</td>
<td>1.5M</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Road networks (mes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Republic</td>
<td>rCA</td>
<td>roadNet-CA</td>
</tr>
<tr>
<td></td>
<td>rTX</td>
<td>roadNet-TX</td>
</tr>
<tr>
<td></td>
<td>rPA</td>
<td>roadNet-PA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Citation graphs (Ggs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Republic</td>
<td>eDP</td>
<td>cit-Patents</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web graphs (Ws)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Republic</td>
<td>wGL</td>
<td>web-Google</td>
</tr>
<tr>
<td></td>
<td>wHS</td>
<td>web-BerkStan</td>
</tr>
<tr>
<td></td>
<td>wSF</td>
<td>web-Stanford</td>
</tr>
</tbody>
</table>

| Clique searches         |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|------------------------|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Republic | cDP | cli-Patents | 16.2M | 1.75 | 8 | 3.63 | 0.01 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 |
|          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Web graphs (Ws)        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Republic | wGL | web-Google | 9.4M | 1.75 | 8 | 3.63 | 0.01 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 |
|          | wHS | web-BerkStan | 7.6M | 1.75 | 8 | 3.63 | 0.01 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 |
|          | wSF | web-Stanford | 2.3M | 1.75 | 8 | 3.63 | 0.01 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 | 1.28 |

SMALL
OUTPERFORMING STATE-OF-THE-ART HASWELL
OUTPERFORMING STATE-OF-THE-ART

HASWELL

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000
OUTPERFORMING STATE-OF-THE-ART
HASWELL

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000

1.85x on average, up to 4.3x
OUTPERFORMING STATE-OF-THE-ART
SCALABILITY ANALYSIS: DISTRIBUTED-MEMORY

PBGL does not support threading, thus we run more than 1 process/node.

PBGL 1 process/node
PBGL, 4 processes/node
AAM, 1 thread/node
AAM, 4 threads/node

The whole node memory filled
PBGL, 128 nodes
PBGL, 16 nodes

AAM, 128 nodes
AAM, 16 nodes
OTHER ANALYSES
OTHER ANALYSES

- BPG mechanism
- Post-Non-NL
- HTM implementations of atomic CAS

- Has-P mechanism
- HTM-HTM

- No aborts due to buffer overflows

- Percentage of the aborts

- Memory conflicts

- FTM buffer overflows

- Total time (s)

- Transaction size (M) (vertices)

- Has-RTM
- BGQ-HTM-L
- BGQ-HTM-S

- Abort due to:
  Memory conflicts
  Buffer overflows
  Other reasons

- Has-RTM
- BGQ-HTM-L
- BGQ-HTM-S

- Memory conflicts
- Buffer overflows
- Other reasons

- Nodal metrics
- Commodity hardware
- Commodity graphics
- Web graphs

- Numbers are local + remote marked vertices
CONCLUSIONS
CONCLUSIONS

HTM for graphs in SM & DM environments
CONCLUSIONS

HTM for graphs in SM & DM environments

AAM: Combine the advantages of HTM and Active Messages
CONCLUSIONS

HTM for graphs in SM & DM environments

AAM: Combine the advantages of HTM and Active Messages

Illustrate HTM’s advantages in performance, next to programmability
CONCLUSIONS

HTM for graphs in SM & DM environments

AAM: Combine the advantages of HTM and Active Messages

Illustrate HTM’s advantages in performance, next to programmability

Deliver the hierarchy of atomic messages that covers various graph algorithms
CONCLUSIONS

HTM for graphs in SM & DM environments

AAM: Combine the advantages of HTM and Active Messages

Illustrate HTM’s advantages in performance, next to programmability

Deliver the of hierarchy of atomic messages that covers various graph algorithms

Detailed performance analysis
CONCLUSIONS

HTM for graphs in SM & DM environments

AAM: Combine the advantages of HTM and Active Messages

Illustrate HTM's advantages in performance, next to programmability

Deliver the hierarchy of atomic messages that covers various graph algorithms

Detailed performance analysis

Model & analyze performance tradeoffs
CONCLUSIONS

HTM for graphs in SM & DM environments

AAM: Combine the advantages of HTM and Active Messages

Illustrate HTM’s advantages in performance, next to programmability

Deliver the of hierarchy of atomic messages that covers various graph algorithms

Detailed performance analysis

Derive close-to-optimal transaction sizes for Haswell & BG/Q

Model & analyze performance tradeoffs
CONCLUSIONS

HTM for graphs in SM & DM environments

AAM: Combine the advantages of HTM and Active Messages

Illustrate HTM's advantages in performance, next to programmability

Deliver the hierarchy of atomic messages that covers various graph algorithms

Detailed performance analysis

Derive close-to-optimal transaction sizes for Haswell & BG/Q

Model & analyze performance tradeoffs

Accelerating state-of-the-art

Average speedup 1.85x Up to 4x
Thank you for your attention
DISTRIBUTED HTM TRANSACTIONS

![Graph showing the relationship between total time and number of nodes.]
TRANSFERRING TRANSACTIONS
INCREMENTING RANKS OF VERTICES
Can we amortize HTM transactions’ transfer overheads with coalescing?
Can we amortize HTM transactions’ transfer overheads with coalescing?

Yes, we can!
SINGLE-VERTEX TRANSACTIONS
INCREMENTING VERTEX RANK

Lower contention
(10 accesses/vertex)

Higher contention
(100 accesses/vertex)

Numbers are total aborts per data point

Threads per node (T)

Total time [ms]

Numbers are total aborts per data point

Threads per node (T)

Total time [ms]
**Single-Vertex Transactions**

**Incrementing Vertex Rank**

- Lower contention
  - (10 accesses/vertex)
  - Atomics always outperform HTM

- Higher contention
  - (100 accesses/vertex)

---

Used in PageRank

---

**Graphs**

- **Operation**
  - BGQ-HTM-L
  - BGQ-HTM-S
  - Has-RTM
  - Has-ACC

- **Total time [ms]**
  - 624 (BGQ)
  - 1.5k
  - 76
  - 43 (RTM)
  - 10 (HLE)

- **Total time [s]**
  - 1.000
  - 0.100
  - 0.010
  - 0.001

- **Numbers are total aborts per data point**
  - 433
  - 452
  - 18k
  - 5k
  - 2.1k
  - 568

- **Threads per node (T)**
  - 1 2 4 8 16 32 64

---

**Mentions**

- IBM
- Intel
**SINGLE-VERTEX TRANSACTIONS**

**INCREMENTING VERTEX RANK**

Lower contention
(10 accesses/vertex)

Higher contention
(100 accesses/vertex)

*Atomics always outperform HTM*

*The reason: each transaction always modifies some memory cell, increasing the number of conflicts*
**Single-Vertex Transactions**

**Incrementing Vertex Rank**

- More aborts
- Lower contention (10 accesses/vertex)
- Higher contention (100 accesses/vertex)
- Atomics always outperform HTM

The reason: each transaction always modifies some memory cell, increasing the number of conflicts

Used in PageRank
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

![Graph showing the comparison of BGQ mechanism with HTM-Long-Mode and HTM-Short-Mode. The x-axis represents the transaction size (M) in vertices, and the y-axis represents the total time in seconds. The graph includes points for 0%, 0.03%, 5.7%, 5.9%, 6.7%, and 5.5%.]
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Numbers: % of aborts due to the lack of HTM resources + memory conflicts
**Multi-Vertex Transactions**

Marking vertices as visited

Numbers: % of aborts due to the lack of HTM resources + memory conflicts

Startup and commit overheads

---

![Diagram](image_url)

**BGQ mechanism**

- HTM-Long-Mode
- HTM-Short-Mode

**Total time [s]**

- 0% (HTM-Long-Mode)
- 0.03% (HTM-Short-Mode)
- 5.7%
- 5.5%
- 6.7%
- 5.5%
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Numbers: % of aborts due to the lack of HTM resources + memory conflicts

Abort and rollback overheads

Startup and commit overheads
MULTI-VERTEX TRANSACTIONS
MARKING VERTICES AS VISITED

Startup and commit overheads

Abort and rollback overheads

The sweetspot! (144 vertices)

Numbers: % of aborts due to the lack of HTM resources + memory conflicts
**MULTI-VERTEX TRANSACTIONS**

**MARKING VERTEXES AS VISITED**

- **Startup and commit overheads**
- **Abort and rollback overheads**
- **The sweetspot!** (144 vertices)
- **Not too many aborts due to the lack of HW resources (large cache size & associativity)**

**Numbers:** % of aborts due to the lack of HTM resources + memory conflicts
OUTPERFORMING STATE-OF-THE-ART 
BLUEGENE/Q IBM
OUTPERFORMING STATE-OF-THE-ART BLUEGENE/Q IBM

Average overall speedup over Graph500 (geometric mean): 1.51 (1.85)
OUTPERFORMING STATE-OF-THE-ART BLUEGENE/Q IBM

Average overall speedup over Graph500 (geometric mean): 1.51 (1.85)
OUTPERFORMING STATE-OF-THE-ART
BLUEGENE/Q IBM

Average overall speedup over Graph500 (geometric mean): 1.51 (1.85)

The same transaction size for all graphs
OUTPERFORMING STATE-OF-THE-ART
BLUEGENE/Q IBM

Average overall speedup over Graph500 (geometric mean): 1.51 (1.85)

The same transaction size for all graphs
The same transaction sizes for each graph separately
OUTPERFORMING STATE-OF-THE-ART
BLUEGENE/Q IBM

Average overall speedup over Graph500 (geometric mean): 1.51 (1.85)

The same transaction size for all graphs
The same transaction sizes for each graph separately
OUTPERFORMING STATE-OF-THE-ART
BLUEGENE/Q
IBM

Average speedup: 1

Average overall speedup over Graph500 (geometric mean): 1.51 (1.85)

Average speedup: 1.85

The same transaction size for all graphs
The same transaction sizes for each graph separately
OUTPERFORMING STATE-OF-THE-ART
BLUEGENE/Q IBM

Average speedup: 1

Average speedup: 3.20

Average overall speedup over Graph500 (geometric mean): 1.51 (1.85)

The same transaction size for all graphs

The same transaction sizes for each graph separately
OUTPERFORMING STATE-OF-THE-ART
BLUEGENE/Q IBM

Average speedup: 1

Average overall speedup over Graph500 (geometric mean): 1.51 (1.85)

Average speedup: 3.20

Average speedup: 1.85

The same transaction size for all graphs

The same transaction sizes for each graph separately

Best transaction size: ~24-100 vertices accessed
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Used in BFS, SSSP, ...
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Lower contention
(10 accesses/vertex)

Higher contention
(100 accesses/vertex)

Used in BFS, SSSP, ...
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Lower contention
(10 accesses/vertex)

Higher contention
(100 accesses/vertex)

Used in BFS, SSSP, ...

Atomics (CAS) slightly faster than HTM
Single-vertex transactions

Marking a vertex as visited

- Lower contention (10 accesses/vertex)
- Higher contention (100 accesses/vertex)

- Atomics (CAS) slightly faster than HTM
- Commit overheads dominate

Used in BFS, SSSP, ...
SINGLE-VERTEX TRANSACTIONS
MARKING A VERTEX AS VISITED

Lower contention
(10 accesses/vertex)

Higher contention
(100 accesses/vertex)

Atoms (CAS) slightly faster than HTM
Commit overheads dominate
RTM outperforms other (overcontended) targets

Used in BFS, SSSP, ...
**Single-Vertex Transactions**

**Marking a vertex as visited**

- **Lower contention** (10 accesses/vertex)
  - BG/Q HTM still worse (L1 vs L2 matters!)

- **Higher contention** (100 accesses/vertex)
  - RTM outperforms other (overcontended) targets

- **Commit overheads dominate**
  - Has-RTM vs Has-CAS

- **Atomsics (CAS) slightly faster than HTM**
  - Has-RTM vs Has-CAS

- **Used in BFS, SSSP, ...**

---

**Graphs**:

   - **Total aborts per data point**

   - **Total time [ms]**

- **Threads per node (T)**: 1, 2, 4, 8, 16, 32, 64

- **Numbers are total aborts per data point**

- **Total time [ms]**: 1e-01, 1e-02, 1e-03, 1e-04

- **Total time [ms]**: 10, 0.1, 0.01, 0.001

---

**Key Points**:

- Lower contention (10 accesses/vertex)
  - BG/Q HTM still worse (L1 vs L2 matters!)

- Higher contention (100 accesses/vertex)
  - RTM outperforms other (overcontended) targets

- Atomics (CAS) slightly faster than HTM

- Commit overheads dominate

---

**Legend**:

- BGQ-HTM-L
- Has-HLE
- BGQ-HTM-S
- Has-RTM
- BGQ-CAS
- Has-CAS
**SINGLE-VERTEX TRANSACTIONS**

**MARKING A VERTEX AS VISITED**

- Very few aborts
- Lower contention (10 accesses/vertex)
- BG/Q HTM still worse (L1 vs L2 matters!)
- Higher contention (100 accesses/vertex)
- Atomics (CAS) slightly faster than HTM
- Commit overheads dominate
- RTM outperforms other (overcontended) targets

**Graphs**

- Numbers are total aborts per data point
- Total time [ms]
- Threads per node (T)
- BG/Q HTM still worse (L1 vs L2 matters!)

**Used in BFS, SSSP, ...**
OUTPERFORMING STATE-OF-THE-ART HASWELL
OUTPERFORMING STATE-OF-THE-ART
HASWELL

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000
OUTPERFORMING STATE-OF-THE-ART HASWELL

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000
OUTPERFORMING STATE-OF-THE-ART
HASWELL

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000
OUTPERFORMING STATE-OF-THE-ART
HASWELL

Average speedup: 1

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000

Average speedup: 1.85
OUTPERFORMING STATE-OF-THE-ART
HASWELL

Average speedup: 1

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000

Average speedup: 3.20

Average speedup: 1.85
OUTPERFORMING STATE-OF-THE-ART HASWELL

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000

Average speedup: 1

Average speedup: 3.20

Average speedup: 1.85

1.85x on average, up to 4.3x

Best transaction size: ~2-9 vertices accessed

Average speedup: 1
OUTPERFORMING STATE-OF-THE-ART

HASWELL

Average speedup: 1

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000

Average speedup: 3.20

Average speedup: 1.85

1.85x on average, up to 4.3x

Best transaction size: ~2-9 vertices accessed
OUTPERFORMING STATE-OF-THE-ART

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000

Average speedup: 1

Average speedup: 3.20

Average speedup: 1.85

1.85x on average, up to 4.3x

Best transaction size: ~2-9 vertices accessed
OUTPERFORMING STATE-OF-THE-ART HASWELL

Best transaction size:
~4 vertices
~14 vertices
OUTPERFORMING STATE-OF-THE-ART

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000

Best transaction size:
~4 vertices
~14 vertices
Outperforming State-of-the-Art
Haswell

Average overall speedup (geometric mean) over Graph500: 1.07, Galois: 1.40, HAMA ~1000

1.85x on average, up to 4.3x

Best transaction size:
~4 vertices
~14 vertices
OUTPERFORMING STATE-OF-THE-ART
SCALABILITY ANALYSIS: SHARED-MEMORY

Contestion from atomics dominates the runtime of Graph500
## Outperforming State-of-the-Art BlueGene/Q

| Type                  | ID   | Name              | $|V|   | $|E|$ | $S_{over \ g500}$ ($M = 24$) | $M$ | $S_{over \ g500}$ |
|-----------------------|------|-------------------|-----|-----|-----------------|-----|----------------|
| Comm. networks (CNs)  | cWT  | wiki-Talk         | 2.4 M | 5M  | 2.82            | 48  | 3.35           |
|                       | cEU  | email-EuAll       | 265k  | 420k | 3.67            | 32  | 4.36           |
| Social networks (SNs) | sLV  | soc-LiveJ.        | 4.8 M | 69M  | 1.44            | 12  | 1.56           |
|                       | sOR  | com-orkut         | 3M    | 117M | 1.22            | 20  | 1.27           |
|                       | sLJ  | com-lj            | 4M    | 34M  | 1.44            | 12  | 1.54           |
|                       | sYT  | com-youtube       | 1.1M  | 2.9M | 1.67            | 8   | 1.84           |
|                       | sDB  | com-dblp          | 317k  | 1M   | 1.33            | 8   | 1.80           |
|                       | sAM  | com-amazon        | 334k  | 925k | 1.14            | 8   | 1.62           |
| Purchase network (PNs)| pAM  | amazon0601        | 403k  | 3.3M | 1.45            | 8   | 1.91           |
| Road networks (RNs)   | rCA  | roadNet-CA        | 1.9 M | 5.5M | $\approx 1$     | 2   | 1.59           |
|                       | rTX  | roadNet-TX        | 1.3 M | 3.8M | $\approx 1$     | 2   | 1.53           |
|                       | rPA  | roadNet-PA        | 1M    | 3M   | $\approx 1$     | 2   | 1.52           |
| Citation graphs (CGs) | cIP  | cit-Patents       | 3.7 M | 16.5M | 1.16           | 8   | 1.57           |
| Web graphs (WGs)      | wCL  | web-Google        | 875k  | 5.1M | 1.78            | 12  | 2.08           |
|                       | wBS  | web-BerkStan      | 685k  | 7.6M | 1.91            | 24  | 1.91           |
|                       | wSF  | web-Stanford      | 281k  | 2.3M | 1.89            | 24  | 1.89           |
# Outperforming State-of-the-Art Haswell

<table>
<thead>
<tr>
<th>Input graph properties</th>
<th>Haswell analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td></td>
</tr>
<tr>
<td>Comm. networks (CNs)</td>
<td></td>
</tr>
<tr>
<td>cWT</td>
<td>wiki-Talk</td>
</tr>
<tr>
<td>cEU</td>
<td>email-EuAll</td>
</tr>
<tr>
<td>Social networks (SNs)</td>
<td></td>
</tr>
<tr>
<td>sLV</td>
<td>soc-LiveJ.</td>
</tr>
<tr>
<td>sOR</td>
<td>com-orkut</td>
</tr>
<tr>
<td>sLJ</td>
<td>com-lj</td>
</tr>
<tr>
<td>sYT</td>
<td>com-youtube</td>
</tr>
<tr>
<td>sDB</td>
<td>com-dblp</td>
</tr>
<tr>
<td>sAM</td>
<td>com-amazon</td>
</tr>
<tr>
<td>Purchase network (PNs)</td>
<td></td>
</tr>
<tr>
<td>pAM</td>
<td>amazon0601</td>
</tr>
<tr>
<td>Road networks (RNs)</td>
<td></td>
</tr>
<tr>
<td>rCA</td>
<td>roadNet-CA</td>
</tr>
<tr>
<td>rTX</td>
<td>roadNet-TX</td>
</tr>
<tr>
<td>rPA</td>
<td>roadNet-PA</td>
</tr>
<tr>
<td>Citation graphs (CGs)</td>
<td></td>
</tr>
<tr>
<td>ciP</td>
<td>cit-Patents</td>
</tr>
<tr>
<td>Web graphs (WGs)</td>
<td></td>
</tr>
<tr>
<td>wGL</td>
<td>web-Google</td>
</tr>
<tr>
<td>wBS</td>
<td>web-BerkStan</td>
</tr>
<tr>
<td>wSF</td>
<td>web-Stanford</td>
</tr>
</tbody>
</table>