Fast Scheduling in Distributed Transactional Memory

Costas Busch Maurice Herlihy Miroslav Popovic
Gokarna Sharma

Presented by
Ramesh Adhikari
Graduate Research Assistant
School of Computer and Cyber Sciences, Augusta University

December, 2022

AUGUSTA UNIVERSITY

Table of Contents

AUGUSTA UNIVERSITY

Table of Contents

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.
  - Deadlock

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.
  - Deadlock
  - Priority inversion

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.
  - Deadlock
  - Priority inversion
  - Reliance on programmer conventions

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.
  - Deadlock
  - Priority inversion
  - Reliance on programmer conventions
  - Vulnerability to Failure or Delay

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.
  - Deadlock
  - Priority inversion
  - Reliance on programmer conventions
  - Vulnerability to Failure or Delay
- Solution (Transactional memory)

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.
  - Deadlock
  - Priority inversion
  - Reliance on programmer conventions
  - Vulnerability to Failure or Delay
- Solution (Transactional memory)
  - Using TM, code is split into transactions

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.
  - Deadlock
  - Priority inversion
  - Reliance on programmer conventions
  - Vulnerability to Failure or Delay
- Solution (Transactional memory)
  - Using TM, code is split into transactions
  - Sequence of Reads and Writes on shared variable execute atomically

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.
  - Deadlock
  - Priority inversion
  - Reliance on programmer conventions
  - Vulnerability to Failure or Delay
- Solution (Transactional memory)
  - Using TM, code is split into transactions
  - Sequence of Reads and Writes on shared variable execute atomically
  - **Commit transaction:** In the absence of conflicts or failures

AUGUSTA UNIVERSITY

Introduction

- Concurrent processes (threads) need to **synchronize** to avoid introducing inconsistencies in shared data objects.
- Traditional synchronization mechanisms such as **locks** and barriers have well-known **limitations**.
  - Deadlock
  - Priority inversion
  - Reliance on programmer conventions
  - Vulnerability to Failure or Delay
- Solution (Transactional memory)
  - Using TM, code is split into transactions
  - Sequence of Reads and Writes on shared variable execute atomically
  - **Commit transaction:** In the absence of conflicts or failures
  - **abort transaction:** Incase of synchronization conflicts or failures

AUGUSTA UNIVERSITY

Introduction

- Paper considers a data-flow of transaction execution

AUGUSTA UNIVERSITY

Introduction

- Paper considers a data-flow of transaction execution
- Each transaction executes at a single node, but data objects are mobile.

AUGUSTA UNIVERSITY

Introduction

- Paper considers a data-flow of transaction execution
- Each transaction executes at a single node, but data objects are mobile.
- A transaction initially requests the objects it needs and executes only after it has assembled them

AUGUSTA UNIVERSITY

Introduction

- Paper considers a data-flow of transaction execution
- Each transaction executes at a single node, but data objects are mobile.
- A transaction initially requests the objects it needs and executes only after it has assembled them
- After the transaction commits, it releases its objects, possibly forwarding them to other waiting transactions

AUGUSTA UNIVERSITY

Introduction

- Paper considers a data-flow of transaction execution
- Each transaction executes at a single node, but data objects are mobile.
- A transaction initially requests the objects it needs and executes only after it has assembled them
- After the transaction commits, it releases its objects, possibly forwarding them to other waiting transactions
- In a distributed TM, execution time is dominated by the costs of moving objects from one transaction to another. The goal of a transaction scheduling algorithm (sometimes called contention management) is to minimize delays caused by data conflicts and data movement.

AUGUSTA UNIVERSITY

Table of Contents

AUGUSTA UNIVERSITY

Graph Model

- The network is modeled as a weighted graph G, where transactions reside at nodes

AUGUSTA UNIVERSITY

Graph Model

- The network is modeled as a weighted graph G, where transactions reside at nodes
- Edges are communication links, and edge weights are communication delays.

AUGUSTA UNIVERSITY

Graph Model

- The network is modeled as a weighted graph G, where transactions reside at nodes
- Edges are communication links, and edge weights are communication delays.
- At any time step, a node may perform three actions:

AUGUSTA UNIVERSITY

Graph Model

- The network is modeled as a weighted graph G, where transactions reside at nodes
- Edges are communication links, and edge weights are communication delays.
- At any time step, a node may perform three actions:
  - it may receive objects from adjacent nodes

AUGUSTA UNIVERSITY

Graph Model

- The network is modeled as a weighted graph G, where transactions reside at nodes
- Edges are communication links, and edge weights are communication delays.
- At any time step, a node may perform three actions:
  - it may receive objects from adjacent nodes
  - it executes any transaction that has assembled its required objects

AUGUSTA UNIVERSITY

Graph Model

- The network is modeled as a weighted graph G, where transactions reside at nodes
- Edges are communication links, and edge weights are communication delays.
- At any time step, a node may perform three actions:
  - it may receive objects from adjacent nodes
  - it executes any transaction that has assembled its required objects
  - it may forward objects to adjacent nodes.

AUGUSTA UNIVERSITY

Graph Model

- The network is modeled as a weighted graph G, where transactions reside at nodes
- Edges are communication links, and edge weights are communication delays.
- At any time step, a node may perform three actions:
  - it may receive objects from adjacent nodes
  - it executes any transaction that has assembled its required objects
  - it may forward objects to adjacent nodes.
- A Transaction's execution terminates when it commits.

AUGUSTA UNIVERSITY

Graph Model

- The network is modeled as a weighted graph G, where transactions reside at nodes
- Edges are communication links, and edge weights are communication delays.
- At any time step, a node may perform three actions:
  - it may receive objects from adjacent nodes
  - it executes any transaction that has assembled its required objects
  - it may forward objects to adjacent nodes.
- A Transaction's execution terminates when it commits.
- Paper provided offline algorithms to compute conflict-free schedules.

AUGUSTA UNIVERSITY

Table of Contents

AUGUSTA UNIVERSITY

Greedy Schedule

- Use dependency graph
- Pick the transaction and assign valid color i.e. execution time

AUGUSTA UNIVERSITY

Greedy Schedule

- Use dependency graph
- Pick the transaction and assign valid color i.e. execution time
- Repeat until all transactions(nodes) are colored

AUGUSTA UNIVERSITY

Greedy Schedule

- Use dependency graph
- Pick the transaction and assign valid color i.e. execution time
- Repeat until all transactions(nodes) are colored
- Each color is a separate time step

AUGUSTA UNIVERSITY

Greedy Schedule

- Use dependency graph
- Pick the transaction and assign valid color i.e. execution time
- Repeat until all transactions(nodes) are colored
- Each color is a separate time step
- Apply this greedy schedule to

AUGUSTA UNIVERSITY

Greedy Schedule

- Use dependency graph
- Pick the transaction and assign valid color i.e. execution time
- Repeat until all transactions(nodes) are colored
- Each color is a separate time step
- Apply this greedy schedule to
  - Line Graph

AUGUSTA UNIVERSITY

Greedy Schedule

- Use dependency graph
- Pick the transaction and assign valid color i.e. execution time
- Repeat until all transactions(nodes) are colored
- Each color is a separate time step
- Apply this greedy schedule to
  - Line Graph
  - Grid Graph

AUGUSTA UNIVERSITY

Greedy Schedule

- Use dependency graph
- Pick the transaction and assign valid color i.e. execution time
- Repeat until all transactions(nodes) are colored
- Each color is a separate time step
- Apply this greedy schedule to
  - Line Graph
  - Grid Graph
  - Cluster Graph

AUGUSTA UNIVERSITY

Greedy Schedule

- Use dependency graph
- Pick the transaction and assign valid color i.e. execution time
- Repeat until all transactions(nodes) are colored
- Each color is a separate time step
- Apply this greedy schedule to
  - Line Graph
  - Grid Graph
  - Cluster Graph
  - Star Graph

AUGUSTA UNIVERSITY

Line Graph

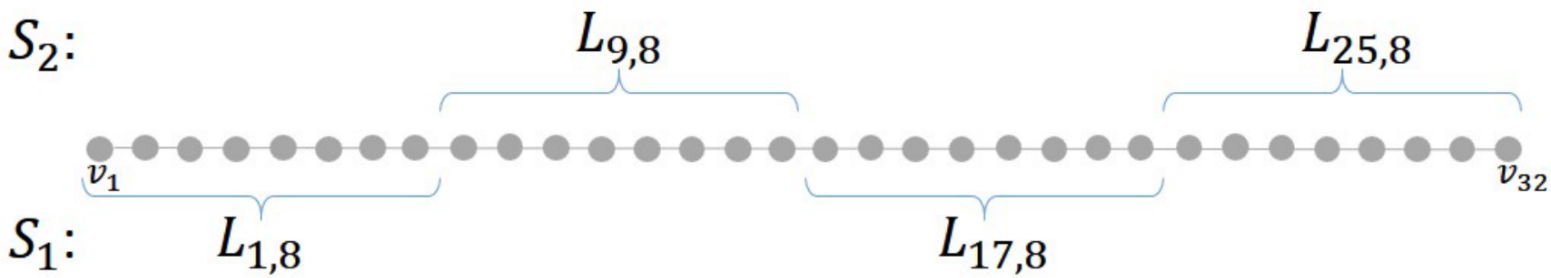

Fig. 1: A line graph with $n = 32$ nodes and $\ell = 8$.

- Each node has a transaction of any node of objects
- L be the longest shortest walk of any object
- Authors schedule the transactions in two phases
- **first phase** execute the transactions in $S1$ And in the **second phase** execute the transactions in $S2$.
- Total time: $\mathcal{O}(L)$

AUGUSTA UNIVERSITY

Grid Graph

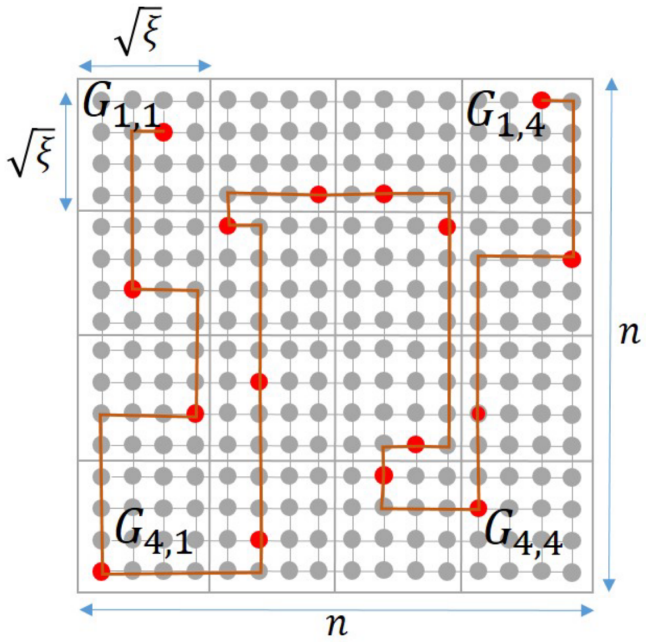

Fig. 2: A grid of size 16×16 with subgrids of size 4×4. It depicts the path of an object.

- Each transaction requests a uniformly random set of k objects (out of w)
- nxn nodes and w objects and decompose graph into sub-grids
- Execute each sub-grid one after another
- In every sub-grid apply **greedy schedule**

AUGUSTA UNIVERSITY

Cluster Graph

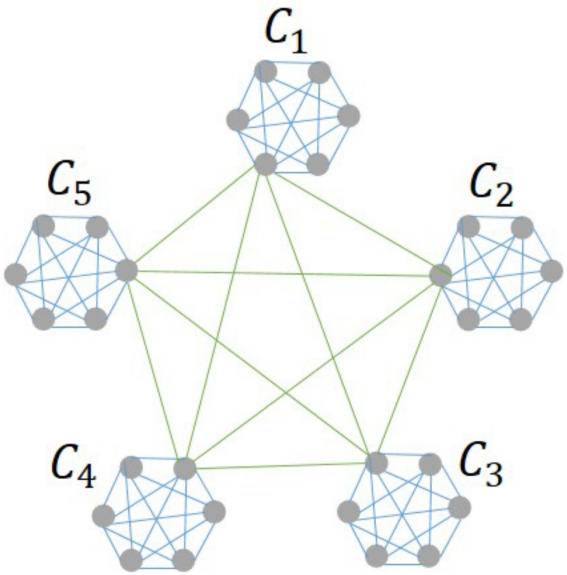

Fig. 3: A graph with 5 clusters where each cluster C_i is a complete graph with 6 nodes; links within clusters have weight 1, while links between clusters have weight γ.

- Each transaction requests k arbitrary objects (out of w)

AUGUSTA UNIVERSITY

Star Graph

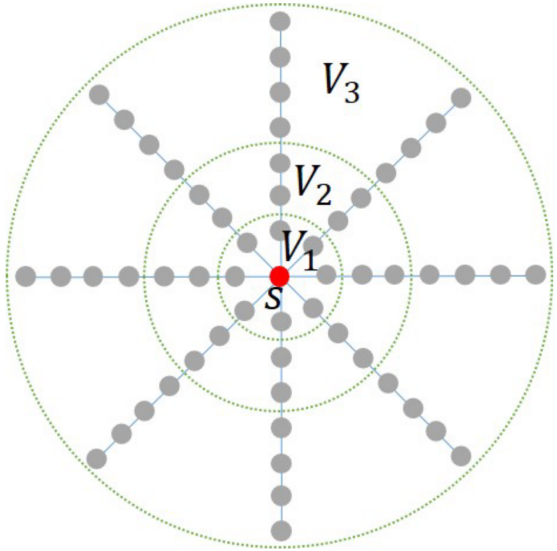

Fig. 4: A star graph with 8 rays, each ray consisting of 7 nodes; the rings depict the set of segments V_1, V_2, V_3.

- Each transaction requests k arbitrary objects (out of w)
- Divide star into rings
- Treat each ring as a cluster graph
- Apply the line algorithm for each ray cluster

AUGUSTA UNIVERSITY

Table of Contents

AUGUSTA UNIVERSITY

Conclusion

- Paper discussed the data-flow model in Transactional Memory
- Paper presented efficient schedules for many network typologies

AUGUSTA UNIVERSITY

Thank You!

AUGUSTA UNIVERSITY