# Lightweight Software Transactions for Games

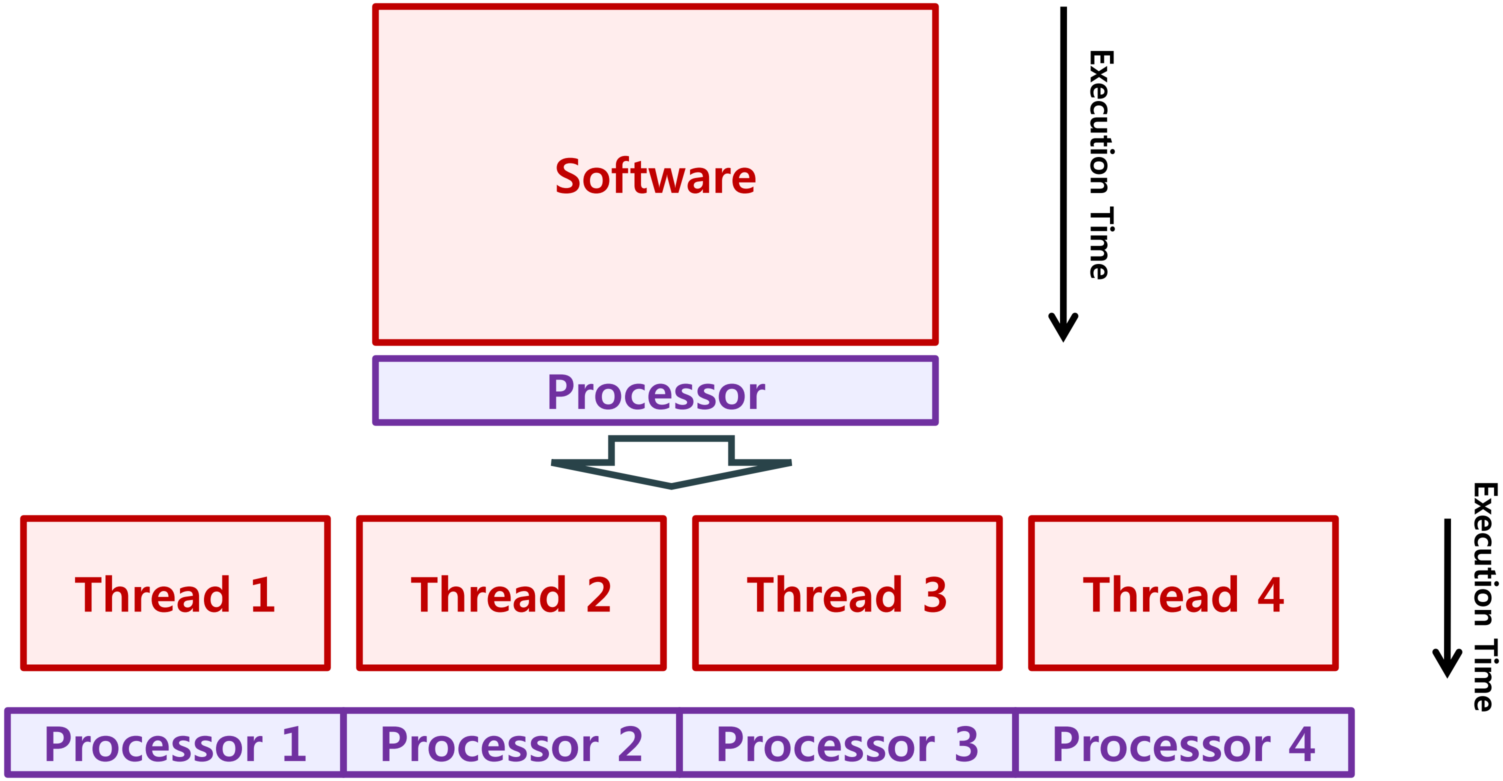
You-Sun Ko
Embedded Systems Languages & Compilers Lab
Yonsei University

# Contents

- Introduction
  - What this paper is about
- The Game
  - Structure of target program or the game
- Challenges
  - Challenges encountered during applying STM
- Solution
  - Solutions for challenges
- Experimental Result
- Future Work

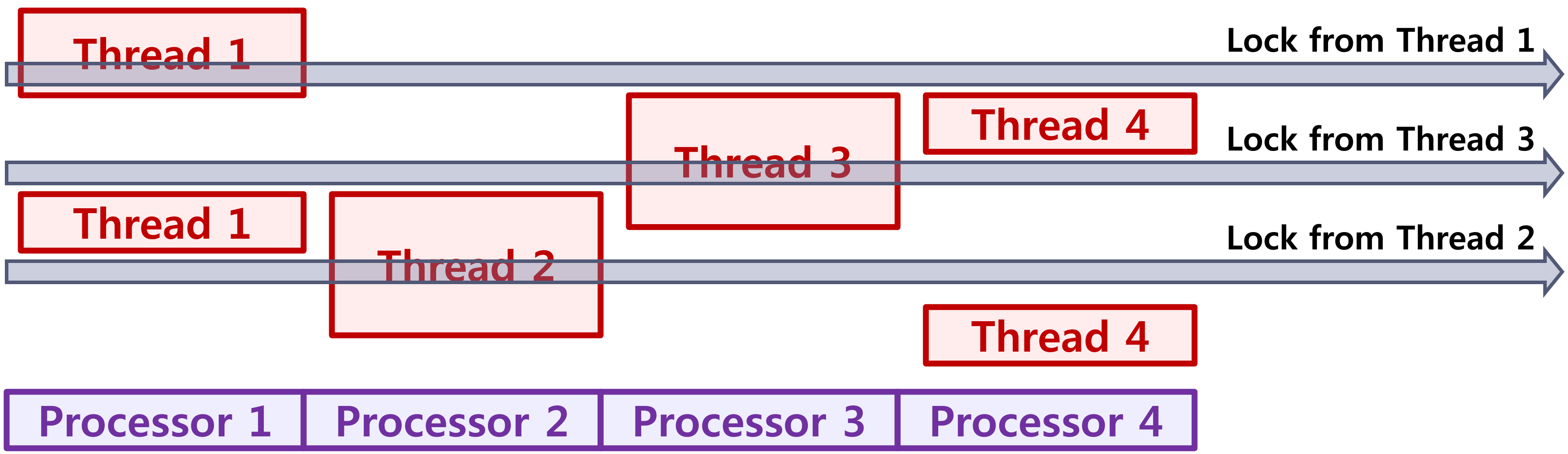
# Introduction

- How to parallelize software?
  - Thread and Lock-based approach on shared memory architecture



# Introduction

- How to parallelize software?
  - Thread and Lock-based approach on shared memory architecture
    - Difficult to use efficiently and correctly



# Introduction

- Why lock-based system is difficult to use?
  - Taking too few locks
  - Taking too many locks
  - Taking the wrong locks
  - Taking  the locks in the wrong order
  - Freeing locks on error

# Introduction

- Transactional Memory
  - Transaction : a block of code that appears to execute atomically and in isolation.
- Concept of Transactional Operation
  - Programmers specify transactions
  - Underlying runtime systems is responsible to detect conflicts and provide a consistent execution of the transactions.

```
lock l;
double a1, a2;

void transfer(double amount){
    lock(l);
    a1 = a1 – amount;
    a2 = a2 + amount;
    unlock(l);
}
```

**Lock-Based Version of Banking**

```
stm_double a1, a2;

void transfer(double amount){
    transaction_start();
    a1 = a1 – amount;
    a2 = a2 – amount;
    transaction_end();
}
```

**Software Transactional Memory Version of Banking**

# Introduction

- But, transactional memory has open issues to solve.
  - How to successfully integrate transactional memory into more mainstream applications?
  - How to deliver a substantially simplified programming experience?
  - How to get competitive performance compared to traditional lock-based designs?

- Here, we are going to see a programming model based on long-running, abort-free transactions with user-specified object-sixe consistency to cover properties of game.

# Game

- SpaceWars3D
  - It is originally made to teach about 3D game programming.
  - It has 3D rendered graphics, sounds, and a network connection.
  - To make computing overhead,  they added moving asteroids.

# Game

- Model-View-Controller Design
  - To express concurrency between the controllers
  - To separate shared data from controller-local data

shared

independent

Renders the game state once each frame

ships, asteroids, projectiles...

Plays sounds each frame

Process incoming packets and send packets

Two tasks each frame
1. collision handling
2. position calculation

handles mouse/keyboard input

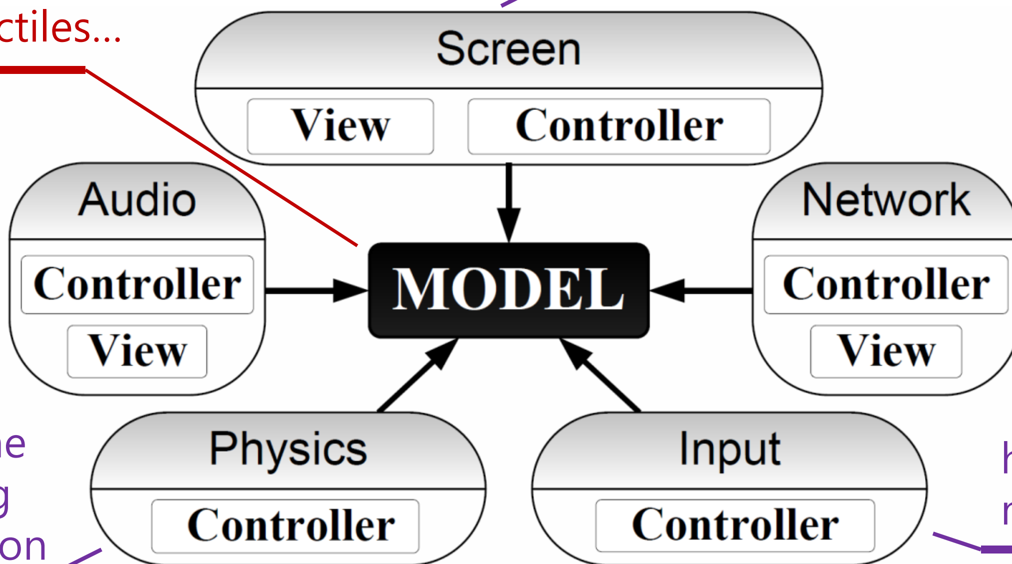



**Figure 1.** Our Model-View-Controller (MVC) design

# Challenges

- Problem 1. Finding Concurrency
  - Original code is alomost completely sequential

- Solution 1.
  - Use natural concurrency among different controllers
  - Some multiple tasks in one module can be concurrent
  - Extreme parallelism can be applied to such as collision handler.

# Challenges

- Problem 2. Which parts are to synchronize(critical section)?
  - Fine-grained locking?
    - higher overhead hard to organize acquires and releases locks without risk of deadlock
    - if task accesses the same data in multiple difference locking phase, data may not be consistent
  - Shortening critical sections?
    - may need to manually copy shared data to local variables and vice verse, hard to maintain
    - changing the length or position of critical sections requires nontrivial code changes

# Challenges

- Problem 3. Optimistic concurrency doesn't work
  - STM uses optimistic concurrency control
  - Optimistic concurrency control : a runtime system monitors the memory accesses performed by a transaction and rolls back if there are any conflicts.
  - But, it did not work because:
    - 1. The game tasks were conflicting every frame(which is not optimistic)
    - 2. Eventhough without conflicts, overhead of transactional execution is discouragingly large
    - 3. There are some features which is not possible to apply transactional memory system, such as I/O.
  - Solution 3.3:
    - Reader Task : do not update the model, but do I/O
    - Updater Task : freely update the model, but not perform I/O

# Solution

- Replica!
  - 1. Each controller tells the runtime system the task is needs to perfom
  - 2. Runtime system then calls these tasks concurrently in each frame, giving each task its own replica of the world to work on
  - 3. At the end of each framework, any updates made to the local replicas are propagated to all replicas

```
public class PhysicsController : Controller
{
   public void Start()
   {
      runtime.NewTask("UpdateCollisions",
                      this.UpdateCollisions);
   }
   public void UpdateCollisions(Context context)
   {
      ...
```

**Figure 2.** Controllers specify periodic tasks, to be called back by the runtime each frame.

# Solution

- Barrier and Merge
  - To deal with task dependencies and conflicting updates, user specify *task barriers* and *merge functions*.
- Optimization to reduce amount of copying
  - readers tasks share the same replica
  - Perform copy on write when a replica is modified for the first time at the end of each frame

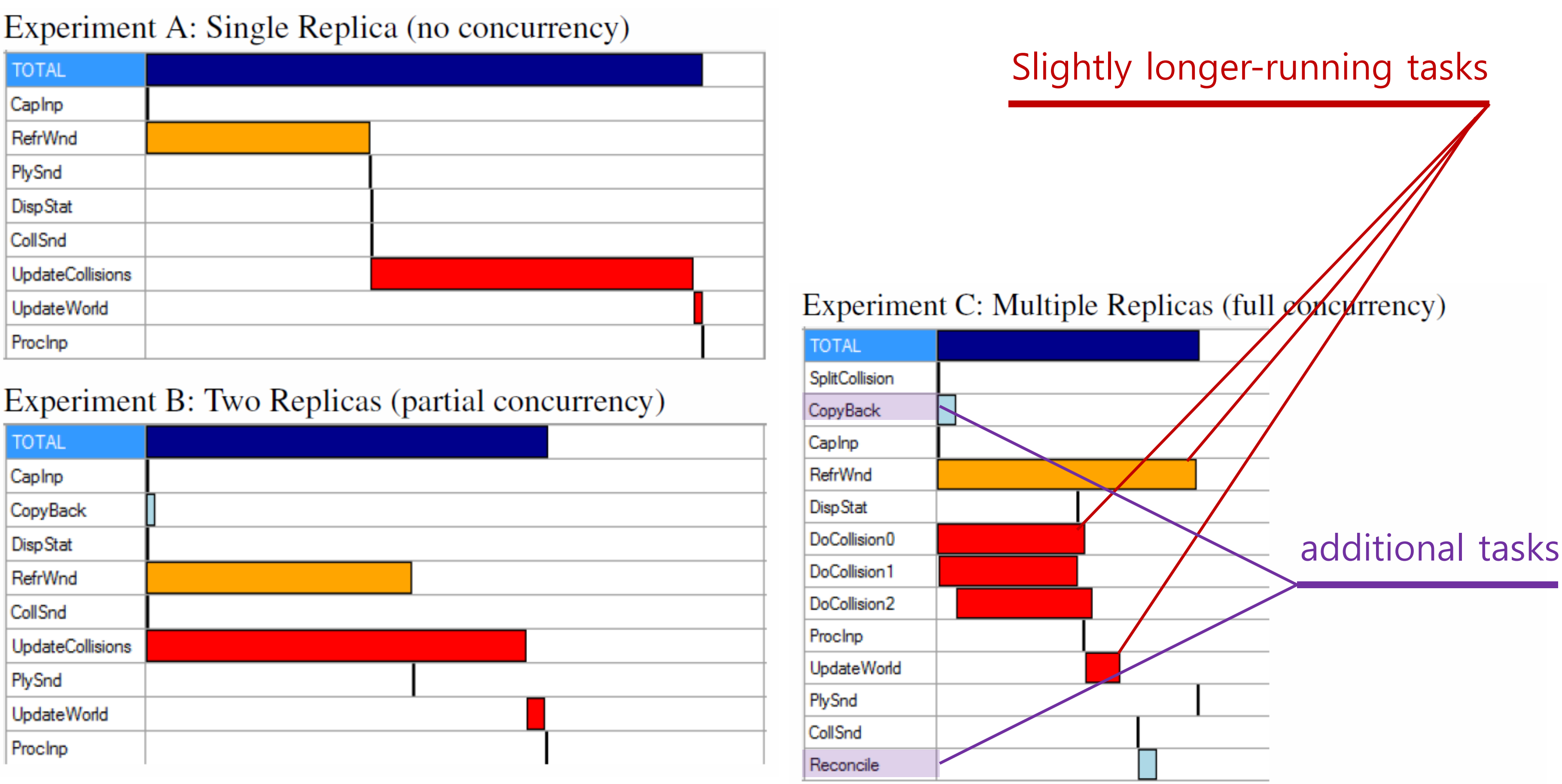
```
MakeBarrier("ProcessInput","UpdateWorld");
MakeBarrier("UpdateWorld", "PlaySounds");
MakeBarrier("UpdateCollisions", "PlaySounds");

ship.score.AddMergeFunction(
  (int old, int new1, ref int new2) =>
                    new2 += (new1 - old));
```

**Figure 4.** The programmer specifies barriers to enforce task dependencies, and merge functions to resolve conflicts.

# Experimental Results

- Experiment A : sequential baseline
  - No replication, no synchronization
- Experiment B : partial concurrency
  - similar to double buffering techniques.
  - one replica is for reader tasks, the other is for updater tasks
- Experiment C : full concurrency
  - uses one replica per task
  - breaks the collison detection task into three pieces

# Common Types of Bugs



# Future Work

- How to further simplify the programmer experience?
  - What about user even do not need to put barrier and merge function?
- Runtime prototype to scale to larger games with many thoughsands of game objects

Thank you for listening.

Any questions?