Distributed System – Lab6 Report

Title: Distributed System Project Report

Course: Distributed System CO3072

Professor: Dr. Thoai Nam

Authors:

Julien Holzer, 2360059

Lê Tiến Phát, 2252600

Group: no12

Institution: Ho Chi Minh City University of Technology

Faculty: Computer Science and Engineering

Date: 05 January 2025

Lab subject: Implementation of a coherent distributed object, applying the concepts of cache coherence to the distributed memory setting

Une image contenant Bleu électrique, bleu, logo, Graphique

Description générée automatiquement

Table of contents

[Introduction 3](#_Toc187243881)

[System Architecture 3](#_Toc187243882)

[Core Components 3](#_Toc187243883)

[Key Features 4](#_Toc187243884)

[Implementation Details 5](#_Toc187243885)

[Token Management 5](#_Toc187243886)

[Network Communication 5](#_Toc187243887)

[Testing Methodology 6](#_Toc187243888)

[Results and Analysis 7](#_Toc187243889)

[Functionality Verification 7](#_Toc187243890)

[Performance Observations 7](#_Toc187243891)

[Test Results 8](#_Toc187243892)

[Conclusions 9](#_Toc187243893)

[Future Improvements 9](#_Toc187243894)

[Demo of the lab 9](#_Toc187243895)

# Introduction

This report details the implementation and testing of a distributed coherent object system that maintains Single Writer Multiple Reader (SWMR) invariance using token coherence. The system was implemented in Java using Remote Method Invocation (RMI) for inter-process communication. Testing demonstrated successful maintenance of data coherence across multiple distributed processes while allowing concurrent reads and properly serialized writes.

# System Architecture

## Core Components

The implementation consists of three main components:

1. DistObjInterface: A Remote interface defining the contract for distributed object operations

2. DistObj: The main implementation class handling distributed state and token management

3. DistObjDemo: A demonstration application showing SWMR functionality

## Key Features

The distributed object system implements several critical features:

1. Token-Based Coherence Management

- Read tokens allow multiple simultaneous readers

- Write token ensures exclusive write access

- Token request and release mechanisms prevent read-write conflicts

2. Network Management

- Dynamic process joining

- Automatic state synchronization for new nodes

- Peer discovery and registration

3. State Management

- Consistent value propagation across nodes

- Thread-safe operations using ReentrantLock

- Remote method invocation for distributed operations

# Implementation Details

## Token Management

The system implements token coherence through two primary mechanisms:

1. Read Token Management

- Tracked using a synchronized Set

- Multiple processes can hold read tokens simultaneously

- Read tokens must be released before write operations

2. Write Token Management

- Single write token tracked using a string identifier

- Write operations require exclusive token ownership

- Token passing implements mutual exclusion for writers

## Network Communication

The implementation utilizes Java RMI with the following characteristics:

1. Registry Management

- Central RMI registry on port 1099

- Automatic registry creation or joining

- Dynamic peer discovery

2. Remote Operations

- Synchronized method calls across network

- Exception handling for network failures

- State propagation to maintain coherence

## Testing Methodology

**Test Environment**

- Platform: macOS

- Development Environment: IntelliJ IDEA

- Testing Configuration: Multiple processes on single machine

**Test Scenarios**

1. Basic Functionality Testing

- Process initialization

- Network joining

- Basic read/write operations

2. Concurrent Operation Testing

- Multiple simultaneous readers

- Write operation serialization

- Reader-writer conflict resolution

3. Network Dynamics Testing

- Process joining during operation

- State synchronization

- Network failure handling

# Results and Analysis

## Functionality Verification

The system successfully demonstrated:

1. SWMR Invariant Maintenance

- Multiple readers operated concurrently without conflicts

- Writers obtained exclusive access

- No read-write conflicts observed

2. State Coherence

- All nodes maintained consistent state

- Updates propagated correctly to all processes

- New nodes received correct initial state

## Performance Observations

1. Read Operations

- Multiple concurrent reads executed efficiently

- Read token acquisition showed minimal latency

- No observable performance degradation with multiple readers

2. Write Operations

- Write serialization maintained correctly

- Token passing occurred smoothly

- Write propagation completed reliably

# Test Results

Example test scenario output:

```

[Initializer] Starting initializer node: node1

[Initializer] node1 current value: 0

[Reader1] Starting reader node: reader1

[Reader1] reader1 read value: 0

[Reader2] Starting reader node: reader2

[Reader2] reader2 read value: 0

[Writer] writer1 attempting to write: 10

[Writer] writer1 successfully wrote: 10

[Reader1] reader1 read value: 10

[Reader2] reader2 read value: 10

```

This output demonstrates:

- Successful process initialization

- Correct value propagation

- Concurrent read operations

- Proper write serialization

# Conclusions

The implemented distributed object system successfully meets its design goals:

1. Maintains SWMR invariant through token coherence

2. Provides reliable distributed state management

3. Handles dynamic process joining and state synchronization

4. Demonstrates proper concurrent operation handling

The system provides a solid foundation for distributed coherent objects and could be extended for various practical applications requiring distributed state management.

# Future Improvements

Several potential enhancements could be considered:

1. Token timeout mechanisms for fault tolerance

2. Performance optimizations for large-scale deployments

3. Enhanced failure recovery mechanisms

4. Support for multiple distributed objects

5. Integration with persistent storage

# Demo of the lab

The demo of the lab is available on YouTube at the following link:

https://youtu.be/t-gpuawlHug