Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

A Java Implementation of a Linda-like Tuplespace System with Nested Transactions

A thesis presented in partial fulfilment of the

requirements for the degree of

Master of Science

in

Computer Science

at Massey University, Albany, New Zealand

Yinan Yao

2006

Abstract

The *Tuplespace* model is considered a powerful option for the design and implementation of loosely coupled distributed systems. In this report, the features of the Tuplespace model are examined as well as the issues involved in implementing such a Tuplespace system based on Java. The system presented includes the function of *Transactions*: a collection of operations that either all succeed or all fail. The system also permits *Nested Transactions*: an extension of transactions. Nested transactions have a multi-level grouping structure: each nested transaction consists of zero or more operations and possibly some nested transactions. The key advantages offered by nested transactions include that they enable the failure of an operation to be isolated within a certain scope without necessarily aborting the entire transaction, and they allow programmers to sub-divide a complex operation into a number of smaller and simpler concurrent operations. The other features of nested transactions are also examined in this report. Finally, the testing results indicate that it is possible to build an efficient, scalable, and transaction secured distributed application that relies on the Tuplespace model and the system developed for this research.

Acknowledgements

First, thanks to my research supervisor Heath James who has aided me greatly throughout my 2 years of postgraduate study. I know I couldn't finish my thesis without his excellent guidance and support. Thanks to many other lecturers in the Computer Science department who taught me a lot in my first year study at Massey University.

Next, the research facilities provided by Massey University were great. I was given 24-hour access to the computer laboratory. The university library contains a large number of useful materials. And also, I have to thank Massey University for the Masterate Scholarship that was a great financial support.

Last, I have to thank my parents who have offered me constant support and encouragement. Their support was always the key reason that kept me going.

Table of Contents

Abstract .... **II**

Acknowledgements .... **III**

Table of Contents .... **IV**

Table of Figures .... **VII**

Table of Code Samples .... **IX**

1. Introduction **1**

1.1 The Tuplespace Concept 2

1.2 Plan of the Report 3

2. An Overview of Tuplespace Architecture **5**

2.1 Principles of Tuplespace Systems 5

2.2 Advantages of Tuplespaces Implemented in Java 6

2.3 Current Major Tuplespace Implementations 9

2.4 Uses of Tuplespace Architectures 10

3. Design of the Java Implemented Tuplespace System **13**

3.1 Overview of the Tuplespace Model 13

3.2 Connecting to a Tuple-space 14

3.3 Tuples and Templates 16

3.4 Storing Tuples in the Tuple-space 17

3.5 Matching Tuples 19

3.6 Partitioning the Tuple-space 21

3.7 Retrieving Tuples from Tuple-spaces 22

3.8 Retrieving Fields from Tuples 23

4. Tuplespace Distribution Pattern **24**

4.1 Message Passing Communications 24

4.2 Communication Protocols 25

4.3 Asynchronous Message Handling 27

4.4 Synchronizing Operations 29

4.5 Collaborating Servers 30

4.6 Design Patterns 31

4.7 Data Package Switching 33

5. Transactions 34

5.1 Transactions and ACID Properties 34

5.2 Distributed Transaction Model 36

5.3 Remote Transaction Implementation 36

5.4 Transactional Operations 38

5.5 Transaction States 40

5.6 Completing a Transaction 41

6. Nested Transactions 45

6.1 What are Nested Transactions? 45

6.2 Some Terminology 47

6.3 Synchronizing Nested Transactions 48

6.4 Create Nested Transactions 51

6.5 Nested Transaction Operations 54

6.6 Completing a Nested Transaction 56

7. System Testing 62

7.1 Ticket Reservation Application 62

7.2 Nested Transaction Testing 67

7.3 Performance Testing 72

7.3.1 Write tuples 72

7.3.2 Read/Take tuples 76

8. Further Research Areas 80

8.1 Client Access Authorization and Management 80

8.2 Dynamic Class Loading 80

8.3 Higher Efficiency and Reliability 82

8.4 Built-in SQL support 82

9. Conclusions 84

References 85

Appendix A: CD – ROM 88

Table of Figures

Figure1 – Tuple-space based process communication 2

Figure2 – Distributed nodes use spaces and simple operations to coordinate activities 3

Figure3 – Client's request handled by cooperating Tuple-spaces 16

Figure4 – A Tuple-space consists of multiple subspaces 21

Figure5 – Message passing between Tuplespace client and server 25

Figure6 – Message protocol for `out( )`, `rd( )` and `tk( )` operations 26

Figure7 – Multiple requests handling scheme 28

Figure8 – Message protocol for result returned from Tuplespace server 28

Figure9 – Cooperating Tuplespace servers 30

Figure10 – A transaction nesting diagram 32

Figure11 – The internal structure of a typical transaction object 42

Figure12 – A transaction nesting diagram 47

Figure13 – A tree diagram for transaction nesting 47

Figure14 – Transaction nesting 47

Figure15 – Transaction nesting example 48

Figure16 – Lock moving among nested transactions 50

Figure17 – Relationship diagram of nested transactions 52

Figure18 – Process order of retrieval operations under nested transactions 55

Figure19 – Committing sequence of nested transactions 57

Figure20 – Lock passing during nested transaction committing 58

Figure21 – Demonstration of a basic ticket reservation process (1) 63

Figure22 – Demonstration of a basic ticket reservation process (2) 65

Figure23 – Status of the test Tuplespace system 68

Figure24 – Contents of the test tuples 68

Figure25 – Tuples used for testing 73

Figure26 – Performance Test for `out()` operations 73

Figure27 – Performance Test for `out()` operations under (sub)transactions 74

Figure28 – Performance Test for `out()` operations under different space conditions 75

Figure29 – Performance Test for `out()` operations under different server conditions 75

Figure30 – Performance Test for `rd()` operations 76

Figure31 – Performance Test for `rd()` operations under transactions 77

Figure32 – Performance Test for `rd()` when retrieving from alternative servers 77

Figure33 – Performance Test for `rd()` under concurrent access situations 78

Table of Code Samples

Code Sample1 – Establish connection with Tuple-space 15

Code Sample2 – Creating tuples and templates 16

Code Sample3 – Define and write tuples to Tuple-space 18

Code Sample4 – Retrieving tuples from the space 22

Code Sample5 – Retrieving field value and type from a tuple 23

Code Sample6 – Create transaction object, and perform transactional operations 37

Code Sample7 – Transaction state constants defined in the system 40

Code Sample8 – Transaction completion syntax 41

Code Sample9 – Creating nested transactions 52

Code Sample10 – Committing/Aborting nested transactions 56