

# **PERSISTENT DATA LIBRARY**

## **MSC IN COMPUTER SCIENCE**



University of Moratuwa, Sri Lanka.  
Electronic Theses & Dissertations  
[www.lib.mrt.ac.lk](http://www.lib.mrt.ac.lk)


**AMILA JAYASEKARA**

**UNIVERSITY OF MORATUWA  
SRI LANKA**

**JANUARY/2008**

# **Persistent Data Library**

**Amila Jayasekara**


This dissertation was submitted to the  
 **Department of Computer Science and Engineering**  
Of the  
**University of Moratuwa**  
In partial fulfillment of the requirements of the  
**Degree of MSc in Computer Science**

Department of Computer Science and Engineering  
University of Moratuwa  
Sri Lanka

January/2008

# **Persistent Data Library**

**Amila Jayasekara**

This dissertation was submitted to the  
 **Department of Computer Science and Engineering**  
Of the  
**University of Moratuwa**  
In partial fulfillment of the requirements of the  
**Degree of MSc in Computer Science**

Department of Computer Science and Engineering  
University of Moratuwa  
Sri Lanka

January/2008

# Abstract

Persistent Data Library (PDL) manages object persistence. PDL provides a set of data structures which transparently handles persistence. Data structures in PDL are quite similar to data structures in Standard Template Library (STL) in C++.

Such a framework is useful for a fault tolerant application. State based applications need to check point data periodically. In case of a failure, such applications need fast recovery of data. In developing such applications, each time a new state is introduced to the system, the programmer needs to write code to serialize and de-serialize data. PDL framework helps the programmer to write less code on serialization and de-serialization. Due to the direct memory dumping technology PDL uses, the time taken to write data to the disk and recover data from the storage are minimized.

The data structures are stored in files in a format that is structurally equivalent to the format of the objects on a volatile medium, such as the memory of a computer.

Objects are stored according to data relationship with other objects. Object references are converted to offset references before they are stored. The objects are stored as a stream of objects and offset references. Offset references are converted to actual memory addresses when objects are retrieved into the volatile medium.

A performance evaluation was carried out between PDL and STL in order to compare performance. Research results shows that there is a significant performance improvement in persisting data inside PDL data structures and loading data into PDL data structures.

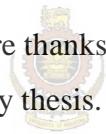
# Acknowledgements

I would like to take this opportunity to show appreciation to every person who gave valuable support in completing this research project.

First I am very thankful to my supervisor Dr. Sanath Jayasena for his excellent supervision and guidance. Dr. Jayasena provided valuable advice and guided me on to the correct direction when required and aided me with lot of guidance and had patients to be with me through out till the successful completion. Also as the MSC project coordinator Dr Jayasena always made sure that I had sufficient resources to carry out my research.

I am grateful to Mr. Tharindu Dissanyake for providing guidelines for finding a good research project. Mr. Dissanayake provided valuable advice and information when ever essential.

My sincere thanks go to Ms. Eroma Abeysinghe for her support in correcting language used in my thesis.



University of Moratuwa, Sri Lanka.  
Electronic Theses & Dissertations  
[www.lib.mrt.ac.lk](http://www.lib.mrt.ac.lk)

# Table of Contents

<b>1 Introduction</b>	<b>1</b>
<b>1.1 Background</b>	<b>1</b>
<b>1.2 The Problem</b>	<b>2</b>
<b>1.3 Research Objectives</b>	<b>4</b>
<b>1.4 Expected Outcomes</b>	<b>5</b>
<b>2 Literature Review</b>	<b>6</b>
<b>2.1 Hibernate [2]</b>	<b>6</b>
<b>2.2 General Persistence [4]</b>	<b>9</b>
<b>2.3A Method for Providing Persistence to Objects in C++ [1]</b>	<b>10</b>
<b>2.4 PTL [15]</b>	<b>13</b>
<b>2.5 POST++ [5]</b>	<b>15</b>
<b>3 Materials and Methods</b>	<b>18</b>
<b>3.1 Big Picture</b>	<b>18</b>
<b>3.2 Object Orientation</b>	<b>18</b>
<b>3.3 Managing Memory</b>	<b>20</b>
<b>3.4 Memory Layout Inside One Block</b>	<b>23</b>
3.4.1 Memory Management with Bitmaps	23
3.4.2 Memory Management with Linked Lists	24
<b>3.5 Memory Management across Memory Blocks</b>	<b>26</b>
<b>3.6 Memory Fragmentation Issue</b>	<b>27</b>
<b>3.7 Memory Arrangement Inside one Block</b>	<b>29</b>
<b>3.8 Metadata</b>	<b>32</b>
<b>3.9 Composing and Decomposing Memory</b>	<b>32</b>
<b>3.10 Making Data Persistent</b>	<b>33</b>
<b>3.11 Loading Persistent Data</b>	<b>36</b>
<b>3.12 The API</b>	<b>38</b>
3.12.1 The API for Creating Objects Using PDL	38
3.12.2 API for Using PDL Data Containers	40
3.12.3 Reference Primitive Data Type Support	42
3.12.4 STL (Standard Template Library) vs. PDL	43
<b>4 Evaluation and Results</b>	<b>45</b>
<b>4.1 Unit Testing</b>	<b>45</b>
<b>4.2 Performance Testing</b>	<b>45</b>
4.2.1 Experimental Setup	45
4.2.2 Setup for Goal 1	46
4.2.3 Setup for Goal 2	50
<b>4.3 Other Results</b>	<b>57</b>
<b>5 Conclusion and Future Improvements</b>	<b>59</b>



University of Moratuwa, Sri Lanka.  
Electronic Theses & Dissertations  
[www.lib.mrt.ac.lk](http://www.lib.mrt.ac.lk)

# List of Figures

Figure 1-1 Operations in a typical fault tolerant environment.....	2
Figure 1-2 Serialized data stream for an object of class 'X' .....	4
Figure 2-1 Hibernate Architecture .....	7
Figure 2-2 Process of creating a persistent object .....	12
Figure 3-1 High level PDL architecture.....	18
Figure 3-2 PDL managed memory layout .....	22
Figure 3-3 : Memory division of 32MB Heap (Division are in 32 bytes).....	23
Figure 3-4 : 1MB Free bitmap structure .....	23
Figure 3-5 Memory with holes and processes .....	24
Figure 3-6 Memory management with Linked lists.....	24
Figure 3-7 Fragmented memory layout .....	27
Figure 3-8 Fragmented memory with unused blocks .....	29
Figure 3-9 List view before and after allocating "X" amount of memory.....	30
Figure 3-10 After de-allocating Y amount of memory .....	30
Figure 3-11 Merging free memory nodes.....	31
Figure 3-12 Node re arranging and merging used memory nodes.....	31
Figure 3-13 Amalgamating all nodes.....	31
Figure 3-11 How memory is stored in the disk.....	34
Figure 3-12 Converting to relative addresses, with multiple blocks .....	35
Figure 3-13 Converting to actual addresses with multiple blocks.....	37
Figure 4-1 Performance testing boundary, for setup 1 .....	46
Figure 4-2 Performance comparison of data insertion to list.....	47
Figure 4-3 Performance comparison of data deletion from the list .....	48
Figure 4-4 Performance Comparison of Data insertion to Map .....	49
Figure 4-5 Performance comparison of data deletion from the map .....	50
Figure 4-6 Performance testing boundary, for setup 2 .....	51
Figure 4-7 How data is stored during performance test (for STL list) .....	51
Figure 4-8 How data is stored during performance test (for STL Map).....	52
Figure 4-9 Performance comparison of data persistence of a list.....	52
Figure 4-10 Performance comparison of data loading from a list .....	53
Figure 4-11 Performance comparison of data persistence of a map.....	54
Figure 4-12 Performance comparison of data loading in a map.....	55



Figure 4-13 Persistence time change against block size for PDL..... 56  
Figure 4-14 Loading time change against block size..... 56



University of Moratuwa, Sri Lanka.  
Electronic Theses & Dissertations  
[www.lib.mrt.ac.lk](http://www.lib.mrt.ac.lk)

# List of Acronyms

API	<i>Application Programming Interface</i>
CPU	<i>Central Processing Unit</i>
J2EE	<i>Java 2 Enterprise Edition</i>
ODMG	<i>Object Data Management Group</i>
PDL	<i>Persistent Data Library</i>
POJO	<i>Plain Old Java Object</i>
PTL	<i>Persistent Template Library</i>
RAM	<i>Random Access Memory</i>
RPM	<i>Revolutions Per Minute</i>
STL	<i>Standard Template Library</i>
TCP	<i>Transmission Control Protocol</i>
UDP	<i>User Datagram Protocol</i>



University of Moratuwa, Sri Lanka.  
Electronic Theses & Dissertations  
[www.lib.mrt.ac.lk](http://www.lib.mrt.ac.lk)