

09PH  
4917



# **Store Architecture in a Persistent Operating System**

**DAVID HULSE**

Thesis submitted for the degree of Doctor of Philosophy  
at the University of Adelaide in the Faculty of Engineering

January 1998  
Department of Computer Science

# Table of Contents

<b>Table of Contents .....</b>	<b>ii</b>
<b>Abstract.....</b>	<b>vi</b>
<b>Statement of Originality .....</b>	<b>vii</b>
<b>Acknowledgments .....</b>	<b>viii</b>
<b>Chapter 1 Introduction.....</b>	<b>1</b>
1.1 Persistence .....	1
1.1.1 Uniform Treatment of Data .....	3
1.1.2 Location Independence of Data .....	3
1.1.3 Resilience of Data.....	3
1.2 Supporting Orthogonal Persistence .....	3
1.3 Persistent Operating Systems .....	6
1.4 A New Persistent Operating System.....	7
1.5 Thesis Structure .....	8
1.6 Contribution.....	9
<b>Chapter 2 Operating System Support for Persistence.....</b>	<b>10</b>
2.1 Introduction .....	10
2.2 Multics.....	10
2.2.1 The Multics Virtual Memory .....	11
2.2.2 Memory Management Features of the GE 645 Processor .....	12
2.2.3 Computation in Multics .....	13
2.2.4 Analysis .....	15
2.3 Monads .....	16
2.3.1 Capability-Based Addressing.....	17
2.3.2 Virtual Addressing Environment .....	17
2.3.3 Procedure Calling Mechanism.....	18
2.3.4 Persistence .....	18
2.3.5 Analysis .....	19
2.4 Clouds.....	19
2.4.1 Analysis .....	22
2.5 Choices .....	23
2.5.1 Analysis .....	24
2.6 Conclusion.....	25

<b>Chapter 3 Grasshopper .....</b>	<b>27</b>
3.1 Introduction .....	27
3.2 Basic Abstractions .....	27
3.3 Address Space Composition .....	29
3.3.1 Locus Mapping .....	30
3.3.2 Container Mapping .....	30
3.3.3 Mapping Mechanisms in other Operating Systems .....	30
3.3.4 Address Resolution .....	31
3.4 Invocation .....	33
3.5 Persistence Architecture .....	34
3.5.1 Managing Virtual Address Spaces .....	34
3.5.2 Management of Container Address Spaces .....	35
3.5.3 Physical Page Sets .....	36
3.5.4 Local Container Descriptors .....	39
3.5.5 Secondary Storage Devices .....	41
3.5.6 Operation of the Persistence Architecture .....	42
3.5.7 Modifications to the Persistence Architecture .....	52
3.6 Conclusion .....	52
<b>Chapter 4 A Persistent Computation Environment .....</b>	<b>54</b>
4.1 Introduction .....	54
4.2 The Persistent Computation Environment .....	54
4.3 Implementation of the PCE .....	55
4.3.1 The Computation Layer .....	56
4.3.2 The Persistent Address Space Layer .....	57
4.4 Relating Grasshopper and the PCE .....	58
<b>Chapter 5 Integrating Persistence with Virtual Memory .....</b>	<b>59</b>
5.1 Introduction .....	59
5.2 Stability and Resilience Mechanisms .....	61
5.3 Challis' Algorithm .....	62
5.4 Shadowing .....	64
5.4.1 The After-Look Mechanism .....	65
5.4.2 The Before-Look Mechanism .....	66
5.4.3 Comparison of Shadowing Techniques .....	68
5.5 Logging .....	69
5.5.1 The Log-Based Approach .....	70
5.5.2 The Log-Structured Approach .....	74
5.6 Summary of Mechanisms .....	76
5.7 Conclusion .....	77
<b>Chapter 6 Logging .....</b>	<b>78</b>
6.1 Introduction .....	78
6.2 The Database Cache .....	78
6.3 ARIES .....	81
6.3.1 Analysis Pass .....	83

6.3.2	Redo Pass.....	83
6.3.3	Undo Pass .....	84
6.3.4	Conclusion .....	84
6.4	The Sprite Log-Structured File System .....	85
6.4.1	Information Retrieval.....	86
6.4.2	Management of Free Space.....	88
6.4.3	Conclusion .....	89
6.5	Texas.....	90
6.6	Recoverable Virtual Memory .....	92
6.7	Conclusion.....	95
6.7.1	No Undo/Redo Logging.....	95
6.7.2	Undo/Redo Logging.....	96
6.7.3	No Undo/No Redo Logging.....	96
6.7.4	Recovery .....	97
6.7.5	Space Management.....	97
<b>Chapter 7 The Log Server .....</b>		<b>99</b>
7.1	Introduction .....	99
7.2	Log Server Design Issues .....	99
7.2.1	Management of Free Space.....	100
7.2.2	Structure of the Physical Log.....	103
7.2.3	Structure of Logical Logs .....	104
7.2.4	Removing Segments From the Log Sequence .....	104
7.2.5	Finding the End of the Log .....	105
7.3	Log Server Interface .....	106
7.4	Log Server Implementation .....	109
7.4.1	Structure of the Logical and Physical Logs .....	109
7.4.2	Segment Allocation and Freeing.....	111
7.4.3	Checkpoints .....	112
7.4.4	Reliable Storage of Segments .....	113
7.5	Conclusion .....	115
<b>Chapter 8 A Restricted Implementation of the PCE .....</b>		<b>116</b>
8.1	Introduction .....	116
8.2	Using a Logical Log as Stable Storage.....	117
8.2.1	Data Structures.....	119
8.2.2	Structure of the Logical Log.....	121
8.3	Implementation of the PCE .....	124
8.3.1	Virtual Memory Management.....	125
8.3.2	Checkpoint .....	127
8.3.3	Recovery .....	129
8.3.4	Cleaning.....	133
8.3.5	Checkpointing the B-Tree.....	134
8.4	Conclusion.....	135
<b>Chapter 9 Managing Consistency.....</b>		<b>136</b>

9.1	Introduction .....	136
9.2	Causality .....	138
9.2.1	Happened Before .....	139
9.2.2	Causal History.....	140
9.2.3	Cuts.....	141
9.3	Mechanisms for Capturing Causality .....	143
9.3.1	Lamport Clocks.....	143
9.3.2	Vector Clocks .....	145
9.4	Checkpoint and Recovery Mechanisms.....	147
9.4.1	Consistent Checkpointing .....	149
9.4.2	Optimistic Checkpointing .....	156
9.5	Conclusion.....	161
<b>Chapter 10 Optimistic Process Checkpointing.....</b>		<b>164</b>
10.1	Introduction .....	164
10.2	Tracking Causal Dependencies in the PCE .....	165
10.3	Tolerating an Inexact Ordering of Events.....	167
10.4	Page-Based Optimistic Checkpointing.....	170
10.4.1	Overview of Operation .....	170
10.4.2	Performing Checkpoints with Larger Granularity .....	172
10.4.3	Taking Advantage of Resident Pages .....	176
10.4.4	Preserving Causal History .....	177
10.5	Algorithms .....	180
10.5.1	Description of Data Structures.....	180
10.5.2	Initial Read Access .....	182
10.5.3	Initial Write Access .....	183
10.5.4	Page Eviction.....	184
10.5.5	Checkpoint.....	185
10.6	Conclusion.....	188
<b>Chapter 11 Performance Measurements .....</b>		<b>189</b>
11.1	Introduction .....	189
11.2	Test Configuration.....	190
11.3	The Logging Subsystem .....	190
11.4	Optimistic Checkpointing.....	191
11.5	The 007 Database Benchmark.....	194
11.6	Conclusion.....	197
<b>Chapter 12 Conclusions.....</b>		<b>198</b>
12.1	Introduction .....	198
12.2	Persistent Operating Systems .....	198
12.3	Store Design .....	200
12.4	Maintaining Consistency .....	202
12.5	Conclusion.....	204
12.6	Future Work.....	205
<b>References .....</b>		<b>207</b>

# Abstract

Many systems supporting *orthogonal persistence* have been constructed on top of the abstractions provided by conventional operating systems such as Unix. These abstractions are not ideal for the construction of persistent systems since they do not allow the hardware to be used in the most effective manner. This observation motivated the design of *Grasshopper*, a new operating system intended to support persistence as a basic abstraction.

This thesis describes the construction of a persistent environment in which multiple concurrent computations can execute and manipulate data stored in a shared address space. Experiments related to this work were carried out using the Grasshopper operating system as a framework. The persistent computation environment, known as the PCE, comprises two major facets of work. The first is a log-structured persistent store, which is used to hold stable representations of the state of the PCE. Such states include the persistent data stored within the shared address space and the meta-data describing the execution state of the computations. The result is a fully persistent system in which both data and computations may be recovered in the event of a failure.

As computations exchange information through the shared address space, causal dependencies are created between them. To preserve the consistency of the PCE, these dependencies must be preserved across failure. This is the focus of the second facet of work, which concerns the design of a new optimistic checkpointing strategy that guarantees to preserve causal dependencies in the event of a failure. It allows computations to checkpoint independently and uses an algorithm described by Johnson and Zwaenepoel to identify recoverable sets of checkpoints lazily.