

Lecture Notes in Computer Science

1002

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Advisory Board: W. Brauer D. Gries J. Stoer

James Jay Kistler

Disconnected Operation in a Distributed File System



Springer

Series Editors

Gerhard Goos

Universität Karlsruhe

Vincenz-Priessnitz-Straße 3, D-76128 Karlsruhe, Germany

Juris Hartmanis

Department of Computer Science, Cornell University

4130 Upson Hall, Ithaca, NY 14853, USA

Jan van Leeuwen

Department of Computer Science, Utrecht University

Padualaan 14, 3584 CH Utrecht, The Netherlands

Author

James Jay Kistler

Systems Research Center, Digital Equipment Corporation

130 Lytton Avenue, Palo Alto, CA 94301, USA

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Kistler, James Jay:

Disconnected operation in a distributed file system / James Jay Kistler. - Berlin ;

Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ;

Tokyo : Springer, 1996

(Lecture notes in computer science ; 1002)

ISBN 3-540-60627-0

NE: GT

CR Subject Classification (1991): D.4, C.2.4, E.5, F.2.2

ISBN 3-540-60627-0 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1995

Printed in Germany

Typesetting: Camera-ready by author

SPIN 10486054 06/3142 - 5 4 3 2 1 0 Printed on acid-free paper

For Chris

Foreword

Tension between *autonomy* and *interdependence* lies at the heart of every distributed system. The ability to use remote resources enhances the storage capacity and computational power of a client. But there is a price to be paid: reliance on remote resources renders the client vulnerable to failures of the network or servers. In the worst case, a client can be totally crippled by the unavailability of a remote resource. This problem is already serious today, and will only worsen with further growth in the size and complexity of distributed systems.

How can one alleviate this problem? The traditional approach has been to use *replication* at servers. Unfortunately, there are limits to the value of this approach. It comes at significant hardware cost. Worse, it is useless if a network failure isolates a client from all server replicas. The latter scenario is especially common in *mobile computing*, where intermittent connectivity is an unfortunate fact of life.

In this doctoral dissertation, Jay Kistler describes a radically different solution. His approach, called *disconnected operation*, calls upon a client to mask failures from users and applications by emulating the functionality of a server. For distributed file systems, Kistler observes that this emulation can efficiently and cheaply be performed by exploiting the file cache already maintained by the client for performance reasons. This leads to a tantalizingly simple design: it just requires pre-loading the cache with critical data, continuing normal operation until disconnection, logging all changes made while disconnected, and replaying them upon reconnection.

Of course, reality is never that simple. There are many conceptual and implementation problems that arise when one tries to implement this functionality. For example:

- How does one arrange to have the right files in the cache at disconnection?
- How does the client conserve scarce cache space while disconnected?
- How can the process of reintegrating changes be made efficient and transparent?
- What are the security implications of disconnected operation?
- How can one reconcile the conflicting demands of availability and performance on caching?

- How can disconnected operation be seamlessly integrated with server replication?
- How likely are update conflicts, and how does one detect and cope with them?

Kistler answers these and many related questions in this work. The implementation he describes is of such high quality that it has been in serious use for over four years. The evaluation of the system is thorough and it sheds much light on the above questions. In addition, the work describes a new model of computation called the “inferred transaction model” that cleanly captures the semantics of disconnected operation. This model serves as an excellent conceptual foundation for reasoning about the consistency properties of disconnected operation.

This research has had substantial impact on industry. Many software companies have efforts under way to exploit these results commercially. There is now broad consensus that disconnected operation is a key enabling technology for mobile computing.

In closing, it is a pleasure to read a dissertation that is such a model of clarity and lucid exposition. The subtleties of the subproblems are brought out with great skill, and the specific solutions adopted are convincingly substantiated. The document is indeed worthy of the research it describes. I would expect no less of my friend, colleague, and former graduate student, Jay Kistler.

*M. Satyanarayanan
Professor of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania
August 1995*

Preface

Disconnected operation refers to the ability of a distributed system client to operate despite server inaccessibility by emulating services locally. The capability to operate disconnected is already valuable in many systems, and its importance is growing with two major trends: the increasing scale of distributed systems, and the proliferation of powerful mobile computers. The former makes clients vulnerable to more frequent and less controllable system failures, and the latter introduces an important class of clients which are disconnected frequently and for long durations – often as a matter of choice.

This dissertation shows that it is practical to support disconnected operation for a fundamental system service: general purpose file management. It describes the architecture, implementation, and evaluation of disconnected file service in the Coda file system. The architecture is centered on the idea that the disconnected service agent should be one and the same with the client cache manager. The Coda cache manager prepares for disconnection by pre-fetching and *hoarding* copies of critical files; while disconnected it logs all update activity and otherwise *emulates* server behavior; upon reconnection it *reintegrates* by sending its log to the server for replay. This design achieves the goal of high data availability – users can access many of their files while disconnected – but it does not sacrifice the other positive properties of contemporary distributed file systems: scalability, performance, security, and transparency.

Disconnected operation in Coda was designed and implemented during the period of 1989 to 1993. At the time this dissertation was completed, the system had been actively used by more than 20 people over the course of two years. Both stationary and mobile workstations had been employed as clients, and disconnections had ranged up to about ten days in length. Usage experience was extremely positive. The hoarding strategy sufficed to avoid most disconnected cache misses, and partitioned data sharing was rare enough to cause very few reintegration failures. Measurements and simulation results indicated that disconnected operation in Coda should be equally transparent and successful at much larger scale.

Since 1993 the system has continued to be used as a research vehicle at Carnegie Mellon University. The number of local users has grown significantly and the code has been made available for distribution outside of CMU.

Coda researchers have extended the system with important new functionality, including weakly-connected operation, advanced hoarding support, and transactional file system semantics. Reports of this progress are beginning to appear and to generate discussion in the academic literature.

Disconnected file service is also beginning to make its mark in the commercial world, driven by the tremendous success of mobile computers in the marketplace. Products offering limited forms of disconnected file support have been available from small companies for several years, and industry heavyweights such as IBM, DEC, and Microsoft are beginning to weigh-in with efforts of their own. These initial products are all hampered to some degree by the legacy of PC operating systems, but the obstacles are rapidly being overcome. I am certain that we will see more – and better – products of this type in the near future. Indeed, I am as confident today about the future of disconnected operation as I was two years ago when I wrote the final sentence of this dissertation: *The advantages of disconnected file service are so compelling that its support will – in my opinion – be a standard feature of all widely-used operating environments of the future.*

Acknowledgments

Performing the thesis research and writing this dissertation turned out to be a larger undertaking than I ever imagined. I could not have completed it without the care and support of many wonderful people, and I'm delighted to be able to acknowledge them here.

First, I would like to thank my advisor, Satya. I couldn't have had a better mentor. He always made time to see me, no matter how busy his schedule. He was a constant source of good ideas and an infallible detector of bad ones. He challenged me when I needed to be challenged and boosted my confidence when it needed to be boosted. He taught me the importance of critical thinking and of validating one's ideas through experimentation. More than anything, though, he has been a true and steady friend.

The other members of my thesis committee were helpful throughout my career at CMU. Early on, Rick Rashid and Eric Cooper co-advised me and made me feel comfortable as I was finding my way. Later, they gave sound advice on my topic and on research in general. Mike Schroeder helped to expose and formulate the problem that my work addresses, and his careful reading and critiquing of the dissertation made it a much better document. I thank all three of them for their efforts.

I warmly thank my colleagues in the Coda group: Maria Ebling, Puneet Kumar, Qi Lu, Hank Mashburn, Lily Mummert, Brian Noble, Josh Raiff, Ellen Siegel, and David Steere. They are all very talented individuals and it was a pleasure to work with them. Their help in the design, implementation, and usage phases of my work was invaluable. Many of them also read the dissertation and gave useful feedback on it, and I thank them for that extra

effort. Special thanks go to Puneet, who never failed to help with a problem or to put a happy face on a discouraging development. I have never known anyone as cheerful or as generous as he.

I thank all of the people who used my software, who persevered through the rough stages of development and who gave me their feedback on what was good and what was bad. My users included all of the members of the Coda group, plus these other folks: Anurag Acharya, Adam Beguelin, Avrim Blum, David Eckhardt, Garth Gibson, Tammy Green, Tom Mitchell, Hugo Patterson, Henry Rowley, Karen Shay, Peter Stout, Manuela Veloso, and Matt Zekauskus.

A number of people on the CMU facilities staff helped me in setting up the Coda testbed environment, in collecting measurements, and in troubleshooting various problems. Special thanks go to Mike Accetta, Paul Parker, Mark Puskar, and Dimitris Varotsis. Bob Baron, Mike Jones, and Mary Thompson of the Mach group also assisted in these respects, and I thank them as well.

I also wish to thank the CMU-SCS faculty and administration for providing such a stimulating – and yet humane – environment in which to conduct research. I know of no other place that meets both of those objectives so well. Special thanks go to Sharon Burks as the cutter of all red-tape and to Catherine Copetas as the junk-food angel of mercy.

A large and wonderful circle of friends made my family's stay in Pittsburgh very special. In addition to the people mentioned already, I'd like to thank Robert and Edie, Harry, Linda and Phil, Claire and Craig, Rhoda and Julian, Geli and Bernd, Dave and Gretchen, Dave and Ann, Deborah, Wendy and Craig, Brian and Debbie, and Wayne and the rest of the "brew crew" for their friendship and for all of the good times.

My parents, Jim and Rita, instilled in me the desire to learn and the value of hard work. Without that grounding and their love I never could have come this far. My pal, Rigsby, took me on countless rejuvenating trips to the park and returned my affection manyfold. I thank the three of them for always being there.

My first daughter, Hannah, was born part way through my thesis research, and she gave me added incentive to finish. More importantly, she made me realize that, although I couldn't write the perfect dissertation, I couldn't be such a bad guy because I'd helped to bring her into the world. My second daughter, Nora, was born after I'd finished, and she continues to help put my work in perspective. My love and thanks go out to both of them.

Finally, my wife, Chris, deserves greater thanks than I can possibly give. For more than a dozen years she has stayed with me, patiently allowing me to pursue my dreams. She didn't complain when I asked her to move from Berkeley to England to continue my studies. Unbelievably, she didn't object when I asked her to leave London for Pittsburgh so that I could enter graduate school. In the course of almost seven years at CMU she never once complained that I was taking too long. During the really grim periods of dissertation

writing, when I doubted everything I'd done and believed I could do no more, she comforted me and gave me the strength to keep going. Without her there for me I surely would have quit. I will always be grateful to her.

*James Jay Kistler
Palo Alto, California
August 1995*

Table of Contents

1. Introduction	
1.1 Distributed Computing	
1.2 Distributed File Systems	
1.3 Disconnected Clients	
1.3.1 Impact of Large Scale	
1.3.2 Impact of Mobile Computers	
1.4 Disconnected Operation	
1.5 The Thesis	
1.5.1 Requirements for Masking Disconnection	
1.5.2 Establishing the Thesis	1
1.6 Organization of this Document	1
2. Design Rationale	1
2.1 AFS Heritage	1
2.1.1 Vice/Virtue	1
2.1.2 Client Caching	1
2.1.3 Volumes	1
2.2 Coda Charter	1
2.2.1 High Availability in an AFS-Like Environment	1
2.2.2 First- versus Second-Class Replication	1
2.3 Partitioned Replica Control	2
2.3.1 Transaction Model of Computation	2
2.3.2 System-Inferred Transactions	2
2.3.3 High Availability Transaction Specification	3
2.3.4 Optimistic Enforcement of Correctness	4
2.3.5 Replica Control Summary	4
3. Architecture	5
3.1 The View from Venus	5
3.2 Decoupling from Server Replication	5
3.3 Masking Disconnection	5
3.3.1 Cache Miss Avoidance	5
3.3.2 Replica Control	5
3.3.3 Update Buffering	5

XIV Table of Contents

3.3.4 Persistence	55
3.3.5 Security	55
3.4 Computation Correctness	56
3.4.1 Intra-Partition Transaction Processing	56
3.4.2 Merging	58
3.5 Detailed Design and Implementation	60
4. Coda Internals	61
4.1 Client Organization	61
4.1.1 Cache Manager	62
4.1.2 MiniCache	66
4.2 Server Organization	70
4.2.1 File Server	70
4.2.2 Authentication Server	74
4.2.3 Update Client/Server	75
5. Hoarding	77
5.1 Idealized Cache Management	77
5.2 Practical Cache Management	78
5.2.1 Use of Explicit Reference Information	79
5.2.2 Combining Implicit and Explicit Information	80
5.2.3 Use of Naming Information	81
5.2.4 Whole-Object Caching	82
5.3 Detailed Design and Implementation	83
5.3.1 Hoard Database	83
5.3.2 Object Priorities, Resource Allocation, and Naming Effects	93
5.3.3 Cache Equilibrium	96
6. Server Emulation	117
6.1 Disconnected Processing	117
6.1.1 User Transaction Processing	117
6.1.2 Hoard Walking	118
6.2 Transaction Logging	119
6.2.1 Log Organization and Record Format	120
6.2.2 Cancellation Optimizations	121
6.2.3 Store-Record Optimization	132
6.3 Data Persistence	133
6.3.1 Leveraging Off of the Local Unix File System	134
6.3.2 Using RVM for Meta-Data	136
6.4 Other Masking Responsibilities	146
6.4.1 Access Checking	146
6.4.2 Fid Generation	149
6.5 Mitigating Emulation Failures	149
6.5.1 Cache Misses	150

6.5.2	Resource Exhaustion	151
7.	Reintegration	153
7.1	Overview of the Algorithm	153
7.1.1	Prelude	153
7.1.2	Interlude	156
7.1.3	Postlude	158
7.2	Non-Certification Issues	159
7.2.1	Volume-Log Ownership	160
7.2.2	Transaction Validation	161
7.2.3	Back-Fetching	161
7.2.4	Atomicity of Reintegration	162
7.2.5	Closures	164
7.3	Certification	165
7.3.1	Version Certification	166
7.3.2	Value Certification	167
7.3.3	Hybrid Certification	168
7.3.4	Other Certification Issues	173
7.3.5	Summary	177
8.	Evaluation	183
8.1	Implementation Status and Testbed Environment	183
8.2	Qualitative Evaluation	186
8.2.1	Hoarding	186
8.2.2	Server Emulation	193
8.2.3	Reintegration	195
8.2.4	General Observations	198
8.3	Quantitative Evaluation	200
8.3.1	Client Storage Requirements	200
8.3.2	Task Latency	208
8.3.3	Reintegration Latency	211
8.3.4	Cross-Client Write-Sharing	215
9.	Related Work	221
9.1	Systems	221
9.1.1	FACE	221
9.1.2	AFS and Cedar	223
9.1.3	Tait and Duchamp	223
9.1.4	Ficus	225
9.2	Mechanisms	227
9.2.1	Replica Control	227
9.2.2	Pre-Fetching and Hoarding	228
9.2.3	Log Optimizations	230
9.2.4	Recovery in Physical File Systems	230

XVI Table of Contents

10. Conclusions	233
10.1 Contributions	233
10.2 Future Work	235
10.3 Final Remarks	237
Bibliography	239
Index	245

List of Figures

1.1	Effect of Caching on Service Structure	4
2.1	Vice and Virtue	12
2.2	File System View at a Virtue Workstation	13
2.3	Integration of Server Replication and Disconnected Operation	21
2.4	A Non-1UE Execution Resulting from Partitioned Read/Write Conflict	23
2.5	Partitioned Transaction Example where 1SR-ness is Data-Dependent	25
2.6	A 1VSR Partitioned History with Read/Write Conflicts	26
2.7	Multi-Item Queries Violating One-Copy Serializability	27
2.8	Coda Data Item Description	34
2.9	Composition of the 1VSR' History Class	39
2.10	A Non-1VSR' History without Write/Write Conflicts	41
3.1	Venus States and Transitions	52
4.1	Organization of a Coda Client	62
4.2	Typical Organization of Coda Servers	70
5.1	Format of a Venus File System Object (fsobj)	84
5.2	Format of a Hoard Database Entry	85
5.3	Sample HDB Entry Expansion	87
5.4	Sample Meta-Expansion	89
5.5	Name-Context States and Transitions	102
5.6	Reference Priority Recomputation	111
5.7	Status Walk	112
5.8	CheckExpansion Routine Invoked During Status Walk	113
5.9	MetaExpand Routine Invoked During Status Walk	114
5.10	Data Walk	115
6.1	Format of a link Log Record	120
6.2	A 1SR History whose Log is Reintegrateable only after Optimization	123
6.3	Organization of Venus' Address Space	140
7.1	Stages of Reintegration, Major Sub-Tasks, and RPC Traffic	154

XVIII List of Figures

7.2	Atomic Sub-Tasks of Reintegration	163
7.3	A History that is Value but not Version Certifiable	167
7.4	A History that would Fail Version Certification due to Version Granularity	168
7.5	A History that would Fail Certification if Counter Operations were Certified	174
7.6	A History that is Partially Reintegrateable	175
7.7	Canonical Twin Transaction Example	176
7.8	Set CR: The Coda Reintegrateable Histories	178
7.9	Histories Illustrating the Availability Contribution of each Reintegration Feature	180
7.10	A 1SR History that is not Certifiable due to Read/Write Conflict ..	181
7.11	An Explicit-Transaction History that is 1SR but not Certifiable ..	181
8.1	High-Water Marks of Cache Space Used	204
8.2	Optimized versus Unoptimized Cache Space High-Water Marks ..	207
8.3	Reintegration Latency with Log Optimizations Enabled	214

List of Tables

2.1	4.3 BSD File System Interface	30
2.2	Coda Transaction Types and System Call Mapping	31
2.3	Coda Transaction Specification	40
2.4	Summary of Coda Replica Control Features	49
4.1	Summary of Venus LWPs	63
4.2	VFS and Vnode Interfaces	68
4.3	Venus-to-MiniCache Interface	70
4.4	Summary of File Server LWPs	71
4.5	Vice Interface	73
4.6	CallBack Interface	73
5.1	Hoard Program Command Set	85
5.2	Sample Hoard Profiles	91
5.3	Summary of Disequilibrating Events and ISR Actions	104
6.1	Log Record Types and their Type-Specific Fields	121
6.2	Overwritten Subsequences Recognized by Venus	128
6.3	Identity Subsequences Recognized by Venus	129
6.4	RVM Library Interface Routines and Descriptions	139
6.5	Typical Recoverable Heap Quotas and Usage	141
6.6	Access Right Types and Privileges	146
7.1	Comparison of Pure Certification Strategies	169
7.2	Coda Hybrid Certification Specification	170
8.1	Vital Statistics for the Work-Day and Full-Week Traces	203
8.2	Optimized versus Unoptimized High-Water Marks at Simulation End	208
8.3	Task Latency	209
8.4	Reintegration Latency with Log Optimizations Enabled	213
8.5	Optimized versus Unoptimized Reintegration Latency	216
8.6	Write-Sharing in AFS	220