OceanStore:

An Oceanic Data Utility
for Ubiquitous, Highly-Available, Reliable, and Persistent Storage

John D. Kubiatowicz
673 Soda Hall #1776
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776
kubitron@cs.berkeley.edu

1. Information about Principal Investigator (NSF Form 1225)

(submitted via FastLane)

2. List of Suggested Reviewers

1. Professor Kai Li
   35 Olden Street, Room 310
   Princeton University
   Princeton, New Jersey 08544
   Tel: (609) 258-4637, Fax: (609) 258-1771
   EMail: li@cs.princeton.edu

2. Garth Gibson
   Room 8113 Wean Hall
   School of Computer Science
   Carnegie Mellon University
   Pittsburgh, PA 15213-3891
   Tel: (412) 268-5890, Fax: (412) 268-3010
   EMail: garth@cs.cmu.edu

3. Professor John V. Guttag
   MIT Room 38-401
   Massachusetts Institute of Technology
   Cambridge, MA 02139-4307
   Tel: (617) 253-4600, Fax: (617) 258-7354
   EMail: guttag@eecs.mit.edu

4. Professor Ed Lazowska
   Department of Computer Science and Engineering
   University of Washington
   Tel: (206) 543-4755, Fax: (206) 543-2969
   EMail: lazowska@cs.washington.edu

5. Professor Tom Anderson
   316 Sieg Hall
   Department of Computer Science and Engineering
   University of Washington
   Tel: (206) 543-9348; Fax: (206) 543-2969
   EMail: tom@cs.washington.edu

3. PECASE Information Form (NSF Form 1317A)

(signed and dated, submitted with the signed cover sheet).

4. ESPCoR Certification Form (NSF Form 1404)

This section is not applicable to the PI

5. Deviation Authorization

This section is not applicable to the PI

6. NSF Approval for exemption from CAREER eligibility requirements

This section is not applicable to the PI

7. Cover Sheet (NSF Form 1207)

(submitted via FastLane and as a signed original)

8. Project Summary

This document sets forth an NSF CAREER/PECASE grant proposal for research into Extremely Wide-Area Storage Systems (EWASS). Ideally, such storage systems are highly-available from anywhere in the network, exploit automatic replication for disaster recovery, employ strong security by default, and provide performance that is similar to that of local storage under numerous circumstances. Further, such systems are self-repairing, providing automatic serviceability and maintainability.

We envision a utility model in which users pay a monthly fee to one particular "utility provider" while consuming resources from many providers. The utility providers buy and sell capacity amongst themselves in direct analogy to the deregulated electric industry. A demand for capacity or bandwidth in one region of the country would encourage entrepreneurs to bring additional services online. One of the key requirements for such utilities is *nomadic* data, i.e., data that is free to migrate and be replicated anywhere within the system. Nomadic data lends itself to fluid analogies (data is free to "flow" wherever it is needed) and permits flexible exploitation of *spatial locality*. This property is in contrast to existing distributed systems that confine their data to a small number of physical servers within the network.

Given the fluid analogy, the EWASS prototype described within this document is called the "Oceanic Data Utility" or "OceanStore". We outline five assumptions for the OceanStore utility:

- **The "Mostly Well-Connected" Assumption:** Most of the network is comprised of high-bandwidth links and periods of disconnection are brief – even at the leaves.
- **The "Promiscuous Caching" Assumption:** Data can be cached anywhere, anytime.
- **The "Operations-based Interface with Conflict Resolution" Assumption:** Applications utilize an operation-based interface that is oriented toward *conflict resolution*.
- **The "Untrusted Infrastructure" Assumption:** The infrastructure is fundamentally untrusted. This means that only ciphertext is stored in the network.
- **The "Responsible Party" Assumption:** Each repository of data has at least one responsible party that is charged with knowing where the data actually resides and ensuring that it is properly replicated.

These assumptions define the OceanStore infrastructure. In constructing this infrastructure, we propose to utilize a number of technologies, some of them mature, others relatively new:

- Organization of data as a series of "pools" of information. Each pool will include a randomized tree indexing structures (e.g., treaps[7]) connected via Bloom-filter[15] summaries.
- Transaction-like operation structures to describe updates and permit infrastructure-side conflict resolution (similar to Bayou[25]), combined with incremental cryptographic techniques[10][11] to operate directly on encrypted information in the infrastructure.
- Use of erasure-tolerant coding[54] and replication to enhance the survivability of data.
- Online *Introspection*[16] to detect and exploit patterns of access in order to optimize the position of nomadic data.

We propose two distinct phases of OceanStore prototype, one read-only, the other complete with conflict resolution. This phased style of implementation permits incremental testing and analysis of components of the system that are somewhat orthogonal. Further, we propose to use the OceanStore infrastructure to enhance education at Berkeley in several ways. Among them:

- Collaborate with other faculty to combine the persistent storage system with experimental wireless infrastructures in order to exploit online content during class and to provide novel faculty/student interactions at nonstandard venues such as cafés.
- Provide a fundamental infrastructure for undergraduates and graduates exploiting the consequences of ubiquitous computing.

OceanStore provides a foundation for ubiquitous computing, and as such will doubtless have a major impact on educational paradigms.

9. Table of Contents (NSF Form 1359)

1. Information about Principal Investigator (NSF Form 1225) 1
2. List of Suggested Reviewers 2
3. PECASE Information Form (NSF Form 1317A) 3
4. ESPCoR Certification Form (NSF Form 1404) 3
5. Deviation Authorization 3
6. NSF Approval for exemption from CAREER eligibility requirements 3
7. Cover Sheet (NSF Form 1207) 3
8. Project Summary 4
9. Table of Contents (NSF Form 1359) 5
10. Project Description and Results from Prior NSF Support 1

10A. Results from prior NSF Support 1
10B. Career Development Plan 1
10.1 Introduction 1
10.1.1 The Promise of a Utility Infrastructure 1
10.1.2 The Oceanic Data Utility 2
10.2 Technical Discussion 3
10.2.1 Assumptions of the OceanStore Infrastructure 3
10.2.2 Five Major Challenges of the OceanStore Model 5
10.2.3 Data Naming and Location: The Cascaded Pools Hierarchy for Indexing 5
10.2.4 On the Interaction Between Security, Reliability, and Conflict Resolution 7
10.2.5 Introspective Computing Infrastructure: Gathering of Tacit Knowledge 9
10.2.6 Data Economy: The "Glue" for an Information Infrastructure 10
10.2.7 Related work 10
10.3 Educational Activities and the Impact of OceanStore 11
10.3.1 Educational Impact of OceanStore 11
10.3.2 Current Educational Activities of the PI 12
10.4 Project Deliverables 12
10.4.1 Theoretical Results and Algorithms 12
10.4.2 Prototypes 13
10.4.3 Testing and Validation Framework 13
10.4.4 Denouement 13
10.5 Plan of Work 14
10.5.1 Two-phase implementation 14
10.5.2 Proposed Schedule 14
10.6 Collaboration and Technology Transfer 15
10.7 Prior Research and Educational Accomplishments 15

11. References 1
12. Biographical Sketch of Principal Investigator, John D. Kubiatowicz 1

12.1 Vitae 1
12.2 Publications 1
12.3 Collaborators in the Last 48 Months Not Listed Above 2
12.4 PhD Students Advised 2
12.5 PhD Advisor 2

10. Project Description and Results from Prior NSF Support

10a. Results from prior NSF Support

Not previously funded as a PI or co-PI by NSF.

10b. Career Development Plan

10.1 Introduction

The past decade has seen astounding growth in the power and sophistication of electronic devices. Computational power, DRAM capacity, and disk storage capacity have been doubling every 18 months for many years. Coupled with unprecedented decreases in cost and power consumption for such components, these technological advances have placed computer technology in the hands of increasingly unsophisticated users as well as spawning a bewildering array of portable computing devices. Unfortunately, this situation is a disaster waiting to happen: the prospect of naive users with dozens of individual devices, each of which has gigabytes of inconsistent, unprotected, and insecure data is frightening to contemplate. A user could make entries to a personal calendar on one device followed by incompatible entries on a different device, and *never know the difference.* Or, they may entrust important financial or medical information to a portable device, only to misplace it later. In fairness, devices such as the Palm Pilot have introduced notions of "synchronization" in order to deal with multiple, inconsistent copies of data. However, the interfaces for synchronization are ad hoc at best and not well-suited to generalization.

Research into effective collaborative techniques (e.g., systems such as Presto[27]) have stressed non-hierarchical storage systems in which objects are distinguished by semantic properties rather than position in arbitrary hierarchies; physical location is one particularly inflexible type of hierarchy[1]. Naive users (as well as sophisticated ones) do not really *want* to worry about inconsistencies in gigabytes of information spread over countless physical devices. Their *expectations* are that the results of updating a calendar or reading and saving email in one place will be reflected everywhere else. In addition, naive users (as well as sophisticated ones) have neither the time nor inclination to put reliable backup facilities in place. Their *expectation* is that storage devices will never fail. Finally, few users worry about the security and privacy of their information. Their *expectation* is that no-one will eavesdrop on their information. None of these expectations reflects the reality of our current infrastructures[2]. However, such expectations are important if computational infrastructure is to be taken for granted[83].

10.1.1 The Promise of a Utility Infrastructure

While technology has created this problem, it also seems to have provided the physical framework for a solution: within the last decade, the backbone of the Internet has reached an astounding level of connectivity, bisection bandwidth, and aggregate data resources. Further, increasing levels of connectivity are being provided to users through cable-modems, DSL, and wireless modem technology. The possibility that everyone will have access to the Internet anytime from anywhere (with varying levels of bandwidth and reliability) is no longer science fiction. Unfortunately, despite the level of *physical connectivity* enjoyed by Internet devices, many of these devices are still disconnected at the *protocol level*, i.e., are not able to be combined together to achieve unified services. At most, small subsets of devices (owned by individual companies) serve together as oases of connectivity to provide services. The great opportunity for high-availability, reliability and scalability afforded by millions or billions of devices is lost, *because of insufficient mechanisms for sharing.*

Contrast this with utility infrastructures such as the electric distribution grids that serve large regions of the United States[3]. These distribution networks are fed from many individual power plants, owned and operated by different corporations. Yet, despite this administrative fragmentation, the aggregate system provides an extremely simple service model: each consumer pays a single company for the electricity that he or she consumes. Although the electrons that power the light-bulbs in a consumer's home may come from many sources, this complexity is hidden. In the background, service providers buy and sell capacity from one another (to meet varying demands) in a form of "electricity market." This aggregate utility structure has many advantages over a more fragmented structure:

- A single physical distribution network can be used to connect all users of the service.
- Providers have more options for balancing the load during times of unusual demand.
- Consumers leave the details of electricity generation and distribution to experts.

[1] Of course, this was one of the original reasons for introducing the relational database model[21].
[2] Many have discovered this painful reality only after their data has been overwritten, lost, or stolen.
[3] For instance, much of the Eastern seaboard is serviced by a single, unified power-distribution grid.

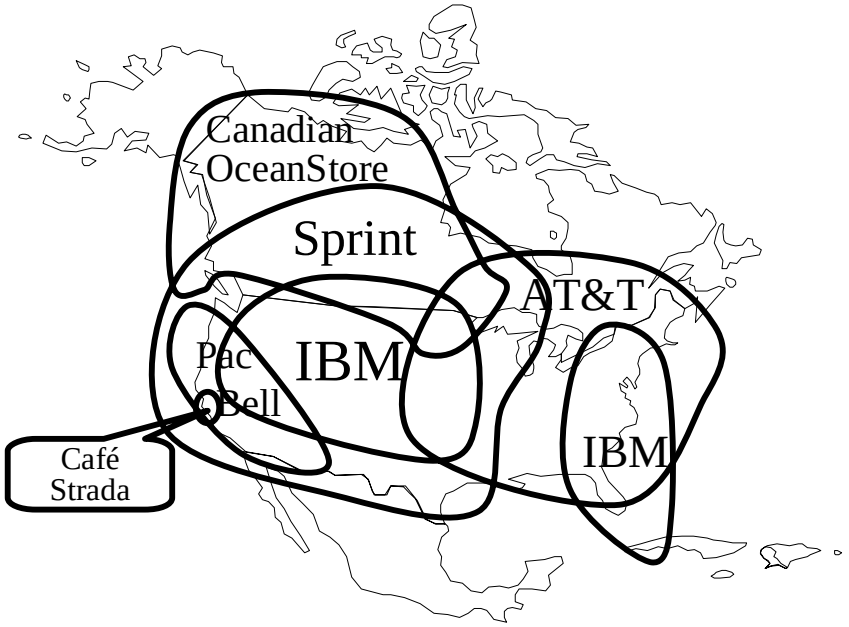


Figure 1: The Oceanic Data Utility

10.1.2 The Oceanic Data Utility

Exploiting a direct analogy with the electric distribution infrastructure, this proposal is about elevating *persistent data storage* to the level of a utility service. We envision a utility model in which consumers pay a monthly fee in exchange for access to their persistent storage. Such a utility would be highly-available from anywhere in the network, employ automatic replication for disaster recovery, use strong security by default, and provide performance that is similar to that of local storage under many circumstances. Further, the self-repairing nature of such systems would make them easy to service and maintain from the standpoint of the utility providers.

Actual services are provided by a confederation of companies, as illustrated in Figure 1. Each user pays a fee to one particular "utility provider" although they consume storage and bandwidth resources from many different providers. The utility providers buy and sell capacity amongst themselves to make up the difference. Insufficient capacity or bandwidth in one particular region of the world could encourage entrepreneurs to bring resources online. Further, small cafés[4] or airports could bring servers on their premises to give customers better performance; in return they would get a small dividend for their participation in global utility.

Ideally, a user would entrust all of his or her data to the utility infrastructure; in return, the utility's economies of scale would yield much better availability, performance, and reliability than would be available otherwise. Properly constructed, the system envisioned here could make the standard protocols for EMail, the Web, filesystems, databases, and software distribution completely obsolete. Further, one of the insidious problems with archival storage, namely the rapid decay of storage media and the equally rapid obsolescence of physical storage formats[5], is directly addressed via information utilities: utility providers simply upgrade or replace their servers as desired; the replication protocols recognize this as a failure and automatically restore the level of replication.

One of our key premises is that users are "mostly connected, most of the time". As a result, once users have entrusted their information to the utility, they can access this information from all of their wireless and wired devices. For instance, on-board devices in cars and boats can access personal databases to track fuel utilization and engine maintenance schedules, access maps and preplanned courses, manipulate phone and email databases, etc. Further, the distributed nature of the information utility means that data sharing is fundamental to the model (sharing between users is identical to sharing between different devices of a single user).

Given its widely-distributed nature, a data utility provides the *opportunity* to continuously adapt to changing aspects of data locality and utilization. The key property required for such adaptation is that data be *nomadic*, i.e.,

[4] Café Strada, referenced in Figure 1, is a small outdoor café near the Berkeley campus.
[5] Consider, for instance, vaults full of information that NASA has collected from the Voyager spacecraft.

free to go wherever it is needed. Of the "traditional" remote file systems that I have found in the open literature (for instance [3][76]), data is typically confined to particular servers, in particular regions of the network. Further, caching is usually confined to components directly on the path between clients and endpoint servers[6]. Experimental systems such as XFS[6] allow "cooperative caching"[15], in which the collective memory of a set of servers can be pooled to form a distributed file cache, but this is limited to systems connected by a fast local LAN. In contrast, the Oceanic Data Utility in this proposal permits generalized caching, anywhere anytime. This is similar to the system-level goals of the Rumor filesystem[41] and the Bayou object store. Nomadic data lends itself to a number of "fluid" analogies: the aggregate collection of servers in the world form an "ocean" of data; this data quickly "flows" to where it is needed; individual caches could be thought of as comprising "lakes," "pools," or "rain-barrels" of data. As a result, we think of this system as the *Oceanic Data Utility* or *OceanStore* for short. Since data is nomadic, an OceanStore system has many options for locality management; this encourages one particular form of *introspective computing*, namely the continuous monitoring of behavior to discover tacit organization and the subsequent use of this "meta-information" for locality management.

10.2 Technical Discussion

Information in the OceanStore is divided into a series of repositories. A repository is the basic element of access control; all documents in a repository are assumed to be encrypted with the same encryption key. In addition to repositories, data can be clustered into an arbitrary number of "collections"[27], which relate data together by an arbitrary set of semantic properties. Documents within collections can be encrypted with many different keys.

10.2.1 Assumptions of the OceanStore Infrastructure

The OceanStore infrastructure involves five different assumptions. We contend that these assumptions follow naturally from the requirements of an extremely-wide scale information utility. By discussing them first, we can provide context for the issues that are discussed later.

The "Mostly Well-Connected" Assumption: First, we assume that large regions of the network are connected by high-bandwidth, high-connectivity networks. Low bandwidth/highly unreliable links are assumed to be close to the leaves of the network. Given its large, distributed nature, OceanStore shares many of the properties that lead to weakly consistent systems such as Coda[3], Ficus[40] and Bayou[63]; however, extrapolating current trends, *we will assume that periods of complete disconnection from the network are short in duration.* Let's call this the "mostly well-connected" assumption.

One interesting consequence of this assumption is that mechanisms such as multicast may be used within the high-bandwidth interior to achieve faster consistency between replicas; this is a departure from the pair-wise updates built into the "anti-entropy" protocols of Bayou[63] and Ficus[64]. Further, the high-bandwidth interior provides ample opportunities for rearranging data for optimum locality as well as providing opportunities to scatter encoded fragments of the data to gain both reliability[36][54] and latency reduction[8][17][18]. Finally, one prominent complaint from users of weakly consistent systems is that they never know for sure when their data is fully committed[7]. The mostly-connected assumption provides greater opportunities to bound periods of inconsistency between replicas. Of course, this is merely a statement about common-case optimizations; any system that we develop will have to degrade gracefully when we are not fully connected.

The "Promiscuous Caching" Assumption: Second, we assume that a user's data can be cached anywhere, anytime. Let us call this the "promiscuous caching" assumption. Information that is in heavy demand could, in principle, be replicated many times in different physical regions of the network. In a system as large as OceanStore, we need to exploit locality to reduce latency and bandwidth utilization. In fact, the need for such caching is clearly validated by the recent explosion of internet companies devoted to Web caching. The challenge is intelligent management of such a vast array of caches. Note that the *potential* benefit from caches is quite large, since many access patterns are likely to have physical locality (if for no other reason than users have physical locality). However, caching introduces overhead in keeping replicas consistent. Ideally, communication traffic between replicas should be confined to *necessary* communication, i.e., traffic resulting from actual updates.

The "Operation-based Interface with Conflict Resolution" (OICR) Assumption: Our third assumption is that we are willing to modify our applications to use an operations-based interface. In the presence of unconstrained

[6] One exception to to this is web caching companies, such as Akamai[4], which achieve wide-spread caching for read-only data on the web. However, these companies work within existing protocols and must employ ad hoc solutions.
[7] From personal communication with users.

replication, issues of update semantics, cache consistency, and commit behavior clearly rear their ugly heads. These issues must be addressed *directly* rather than through ad hoc mechanisms. To this end, we make the observation that physically based consistency mechanisms, such as those defined for multiprocessors[53][28][29][33][37], are rarely "ideal" for all (or even any) applications. Typically, such coherence mechanisms have fixed "block" or "cache-line" sizes that are unrelated to the communication granularity of any applications. This can lead to a number of performance problems, such as *false sharing*, when a cache-line bounces between different physical locations simply because it contains two unrelated objects. In addition, no consistency protocol is appropriate for all applications: if too weak, certain types of synchronization may be difficult to perform; if too strong, it may greatly restrict performance. This problem has lead some multiprocessor researchers to support multiple consistency protocols (for instance, [19][52][3]). Most of these techniques invoke compiler support for passing application semantics to the underlying system; however, they are still hampered by the fact that the underlying system does not understand the context for reads and writes.

In contrast, databases have long taken an "operations-oriented" approach, namely grouping related reads and writes into *transactions*[38]. Transactions encapsulate related read and write operations together as atomic units. The fact that the underlying system is aware of this encapsulation permits a lot of flexibility in implementing the desired actions, while retaining a strict notion of correctness (namely serializability [12]). Further, false sharing cannot occur, precisely because consistency is performed at the level of application-level objects rather than arbitrary physical units of communication.

Unfortunately, serializability is overly restrictive. It provides a single-node abstraction which is often unnecessary for many applications. One possible relaxation of the serializability constraint is to allow operations from geographically separated users to conflict, and to invoke server-side mechanisms for *conflict resolution*. Conflict resolution mechanisms encapsulate a series of read and write operations, together with a specification of what to do when these operations conflict with others. This approach, taken by Bayou[25][79][30], is one that we agree with, i.e., that an operations-based interface with conflict resolution is crucial for good performance. Unlike Bayou, however, we believe that this interface should be flexible enough to permit applications to select from the full gamut of consistency mechanisms, from database-level serializability to extremely weak file-level merging. Note that the "Operation-based Interface with Conflict Resolution" (OICR) assumption implies that we are willing to modify our applications in order to get the full benefits of the interface; none-the-less, it will be an important that traditional file-level operations can be mapped on top of this interface with slight loss of performance.

The "Untrusted Infrastructure" Assumption: A fourth assumption that distinguishes OceanStore from most other projects is that infrastructure is fundamentally *untrusted*. Some servers may crash without warning (although we assume that this is infrequent) and others may be run by industrial spies or tabloid reporters. While one particular utility provider may be "responsible" for the integrity of a given client's data (the fifth assumption, below), we must assume that none of them can be trusted with the cleartext versions of data. Others have come to this conclusion with local file servers[13]. This lack of trust is inherent in utility model; only endpoints can be trusted with data, and all information that enters the infrastructure must be encrypted. However, the cryptographic operations must be completely transparent and directly incorporated into the storage system for ease of application development and deployment[13]. Unfortunately, the fact that data is encrypted within the infrastructure has a number of important consequences:

- Information sharing must be accomplished by passing permanent keys to collaborators, rather than acquiring temporary session keys to cleartext repositories.
- Conflict resolution must somehow operate on encrypted information.
- Data location and cache optimization mechanisms must somehow deal with ciphertext.
- Arbitrary, proxy-like computation within the infrastructure is difficult, if not impossible.

These consequences must be addressed by any OceanStore solution.

The "Responsible Party" Assumption: The final assumption that we make is that there is one particular administrative entity within the infrastructure that is "responsible" for each data repository. Since we assume that clients pay one particular "utility provider" for their OceanStore service, this "home utility provider" could serve as responsible party for all of the user's data. The responsible party is the entity that is ultimately responsible for tracking the latest copies of data and ensuring reliability and availability of a user's data. The responsible party is an abstraction. We make no assumptions that the responsible party is manifested in any particular *physical* location, but rather that it embodies the tracking of information and ability to respond to that information in achieving overall reliability. In order to avoid making the responsible party a bottleneck, we will structure the "common case"

mechanisms to operate independently of this entity. For instance, with sufficiently high levels of replication (or backing of replicas on stable servers), we can reduce the extent to which the responsible party must participate.

There are several advantages to assuming the existence of a responsible party. First, the presence of an ultimate authority enables the use of probabilistic algorithms for some common case mechanisms. For instance, we can structure our data location facilities to locate data *with high probability*, falling back on our responsible party when we are unable to locate data otherwise. Second, the responsible party represents a well-defined entity in which to embody polices for replication, availability, and survivability. Third, the responsible party can make guarantees about reliability and availability and be financially responsible when these guarantees are not metz; this is the only way that people would be willing to trust their information to the infrastructure.

10.2.2 Five Major Challenges of the OceanStore Model

Having explored the assumptions of the model, we can now identify major issues that arise in the design of OceanStore. Although this Principle Investigator has straw-man solutions for some of them, others remain topics of research. We will address some issues in this section, then follow with specific solutions in subsequent sections.

Data Naming and Location: First and foremost, the presence of promiscuous caching greatly complicates the data naming and location problem. Copies can reside almost anywhere, and this requires an extremely flexible data location facility that can quickly locate nearby copies without consuming huge amounts of bandwidth in the network. The "responsible party" assumption is an important assumption in that it permits "mostly correct" indexing structures. This is an important problem that we will explicitly address in Section 10.2.3.

Framework for Conflict Resolution: A second issue is discovering an appropriate framework for conflict resolution. Here we will be searching for an operations-based interface for conflict resolution, as mentioned previously. However, given the "untrusted infrastructure" assumption, this conflict resolution policy must operate on encrypted text. Further, it must interact properly with any redundancy mechanisms employed for reliability. Although we present a straw-man solution in Section 10.2.4, this interaction provides an exciting opportunity for research. Note that the mere existence of encryption necessitates strategies for key-management.

Replication Policies and Mechanisms: A third issue involves selecting forms of replication and redundancy that are compatible with encryption and with conflict resolution within the untrusted infrastructure. In particular, there is a tradeoff between survivability and performance: with greater coding and replication, data is very unlikely to be destroyed, since a large number of individual servers must be corrupted in order to destroy data. However, this level of replication greatly complicates and delays the update process.

Fluid Locality Optimization and Introspective Computing: A fourth issue revolves around the opportunity presented by promiscuous caching. What techniques should be used to optimize the placement of replicas and the system as a whole? Note that any placement decision must maintain data reliability and security. This particular option is "graduated" in the sense that the success of OceanStore does not depend on "perfect" optimization policies. However, the better policies that are produced, the better performance that can be expected.

Data Economic Policies: A final issue is the techniques and policies that would turn an operating OceanStore system into a functioning utility. It is important that such policies be self-stabilizing and self-correcting in addition to being economically attractive to participating entities. This is optional from the standpoint of this proposal, but important for any future wide-scale deployment of the OceanStore.

System such as OceanStore have a number of pieces that fall under the category of "technology synthesis", i.e., pieces that are built in straightforward ways from existing technologies. The data naming and location pieces fall into this category, as do some of the details of the conflict-resolution mechanisms, and filesystem and server technologies. Many of these are a "simple matter of programming," although we do expect some research results from interactions between pieces. However, some of the most interesting individual research results will likely ensue from the interactions between conflict resolution and ciphertext (Section 10.2.4) and the use of introspection for locality management (Section 10.2.5).

10.2.3 Data Naming and Location: The Cascaded Pools Hierarchy for Indexing

The first of the major issues to be discussed is the data naming and location problem. Since copies can reside anywhere within the infrastructure, it is important to have a flexible location mechanism. As shown in Figure 2, information within OceanStore will be divided into a series of "pools" of varying sizes. (These are the shaded ovals). Each pool inhabits a particular physical location in the network and hence may reside on a single node, an

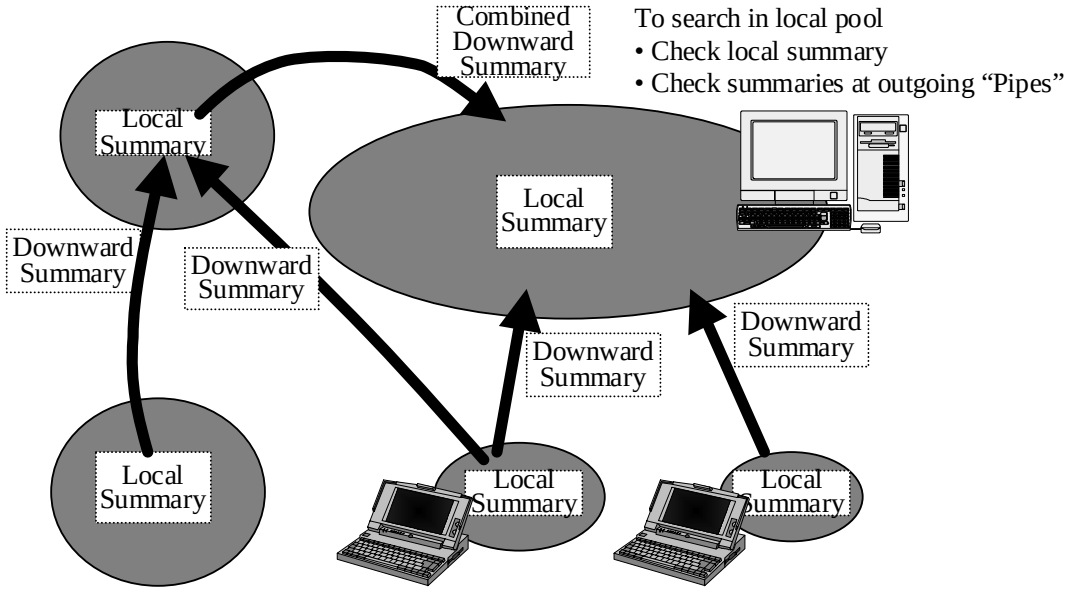


Figure 2: A Cascaded Hierarchy of Pools with summaries along outflow points.

SMP, or a cluster – perhaps all the nodes in a given building. In principle, we imagine that the total number of pools could be much larger than the available space of IP addresses, since there could be small caching servers scattered "everywhere". Attached to each pool may be one or more computational devices utilizing information directly from the local pool. Many of the smaller information pools may reside directly on the devices that they are servicing. For instance, Figure 2 shows two laptops talking directly to small pools – pools that reside within their internal memory and disk. This figure also shows a workstation talking to a much larger pool; in this later case, the pool may reside on servers attached to the local LAN. Since the persistent storage interface is operation-based (Section 10.2.4), access to persistent storage may occur either in the local pool or in remote pools; it is up to the Introspective caching heuristics to direct this decision. Pools are connected by a series of "pipes" that serve as conduits for information. Both items and queries flow from from pool to pool along these conduits. Pools and their interconnecting pipes form the complex, fully connected web that comprises OceanStore. The exact topology of this structure is unspecified, but directly reflects physical proximity, with additional, higher-dimensional links to pools at greater distances. This structure must be built "on the fly", since portions of the pool structure are in constant flux (e.g., as users move around). We anticipate that introspective mechanisms of Section 10.2.5 will assist here.

Unique Object IDs: Given this organization, an obvious question arises: how should information be located and manipulated? First and foremost, we must separate *naming* from *location.* Names are unique, human-readable character strings that represent files, databases, etc[8]. For now, we will call a nameable information entity an "object". Names are immutable and independent of the location of the objects that they represent. Objects, on the other hand, are opaque (i.e., encrypted), change location frequently, can be replicated, and change their content during update operations.

To aid in locating objects, each object is given a unique identifier, generated at the time of its creation. In addition, each *version* of each object is given a unique signature generated with a secure one-way hash mechanism, such SHA-1[61]. Hence, we envision object identifiers (OIDs) generated in two different ways: either from extrinsic properties (i.e., names as character strings) or intrinsic properties (i.e., signatures over the contents of the object). We will use both types of OID and construct a location facility that is capable of taking arbitrary OIDs and finding corresponding objects. The OID-to-object mapping is a function, meaning that each object may have many OIDs, but that each OID must be associated with only one object. In addition, This mapping must be self-verifiable, in that each object contains sufficient information to derive all valid OIDs that might be mapped to it.

We note that the process of resolving names is much like that of locating objects. Rather than imposing artificial structure on the names or location facilities, we simply convert names into unique identifiers via SHA-1. We then use the same location facility (described below) to resolve names into unique identifiers and to resolve identifiers

[8] We are not going to make any claims about how these names are assigned.

into objects. To locate a named object, the name is first hashed into an OID, then *resolved* into a naming object by invoking the location facility. This object holds a globally unique OID[9], which stands for the object that we desire. We then invoke the location facilities again to find the object. The two levels of indirection allows us to separate the actual name (which may be a transient thing) from a persistent pointer to the object. Note that the traditional notions of "directories", "repositories", or "collections"[27] of information are simply objects which contain OIDs.

Fast Local Search: Given an OID for a name, directory, or object, how do we locate it? Assuming that introspective mechanisms are working well (Section 10.2.5), neighboring pools should "often" contain the information that is needed by the local node. Thus, we equip each pool with a fast randomized index structure (such as a treap[7] or similar structure[56]) that permits quick lookup of OIDs in the local pool. These data structures retain good average properties despite frequent insertion and deletion. As a result, lookup within any particular pool is a fast, exact process. During a search, we start with the local pool. If we do not find what we are seeking in the local pool, then we send our search to neighboring pools; given the presence of long-distance conduits between pools, this search can potentially cover much distance with a short number of hops. However, we limit the number of hops that we are willing to travel to look for items in this way. If the local search fails we resort to an exact, hierarchical structure (such as Globe[81]). Such a structure is exact in that a well-defined search algorithm is guaranteed to find references to the objects of interest. The presence of this "last line of defense" is one of the consequences of having a "responsible party". We will update this structure infrequently, only when objects stray far enough from their original locations that such fuzzy "links" would not longer locate the object.

Gradient Search with Bloom Filter Potential Function: To direct our search mechanism, we want something that can serve as an OID *potential function* to drive a form of gradient search. To do this, we associate a potential function at each of the outgoing links of a pool. During a search, we look at the values of this function to decide which of the links would be most effective to traverse.

As an initial take at such a potential, we propose to use a variant of Bloom filters[15]. A Bloom filter is a large vector of bits that summarizes the presence of keys within an index structure, in our case OIDs within a pool. To generate a summary, one applies N *different* hash functions to each OID in the pool, generating N different integers per OID (N is a parameter). The values are used, modulo the length of the vector, to set bits in the vector. During a search, one can check if an item *might* be in a pool by generating N hash values and checking to see if all N of the corresponding bits are set in the summary; if not, then the corresponding pool definitely *does not* contain the item. Note that this admits false positives, i.e., can indicate that an item is in the index when it actually is not. Bloom filters have had a long history in the database community (for distributed joins[55], and Web caching[31], among other things); the tradeoffs in filter size versus probability of false positives has been well characterized.

As shown in Figure 2, we pass the summary vectors to nearest neighbors in the pool structure. In passing summary vectors to neighbors, a local node combines its own summary with a weighted set of summaries from other neighbors. One simple way to do this is to "OR" together the summaries from neighbors with the local summary and pass this to the next pool (this technique is used in [22]). Unfortunately, this technique treats all distances equally. A better option, which we propose to use, is to employ "weighted" summaries. We think of each bit in the summary vector as a real number. Then, we combine summaries by "ORing" together vectors from the neighbors, multiplying the result by ½, then adding in the local summary. (We do this by representing each "bit" of the summary by multiple bits, i.e., a fixed-point representation for numbers less than one.) This technique produces a potential function which is most strongly affected by close pools and less affected by pools that are farther away: the most significant bits of the vector summarize the pool that is one "hop" away, the next bits any pools that are two hops away, etc. These "weighted" summaries let us choose directions to search which appear to lead to our desired object in the shortest number of hops. Note that we can recognize cycles during this summary combination process, and arrange so that each node is represented in only one bit position.

10.2.4 On the Interaction Between Security, Reliability, and Conflict Resolution

In operating upon an object, we can choose to apply requested operations locally or migrate the object to the requestor before performing other actions. Whether an object is accessed in place or migrated first is decided by the introspective component (Section 10.2.5) and is completely transparent to the client. Section 10.2.2 described why an operations-oriented interface with conflict resolution is desirable for accessing the OceanStore infrastructure. In fact, OceanStore takes its cue from Bayou[25] with respect to conflict resolution policies. Updates to the persistent object store are packaged as "operations" which include a set of updates to apply to the object store, a series of criteria for detecting conflicts, and merge procedure to resolve conflicts when detected.

[9]Derived from the time, location, and initial name at the moment of creation (and anything else required for global uniqueness).

To simplify recovery from failures, OceanStore employs a version-based consistency scheme[72]. In general, database systems construct their transaction management mechanisms around one or more stable logs and one or more transient queueing structures[12][42][59]. Typically, a set of such logs are associated with each database server. In OceanStore, however, objects are far more mobile. Consequently, logging activities for each object are centered around *object control records*, which are data structures associated with each replica and which contain, among other things, a directory of pointers to other replicas and a log of pending updates or conflict resolutions (as in Bayou).

As mentioned in Section 10.2.3, every object in OceanStore has a unique OID that is generated at the time of creation. When this OID is dereferenced by the data location service, the result is an object control record for one of the replicas, as well as the most recent data for that replica. In addition, the object control record contains unique OID signatures for versions of the object that were previously "published" or archived. These OIDs can be dereferenced to yield old, read-only versions of the object. This versioning mechanisms gives OceanStore the properties of permanent archival storage (much like Intermemory[36]), but as an integral part of the consistency mechanism. Note further that all archival documents have a signature which is not only used to locate them, but which can be used to authenticate them as well.

Conflict Resolution on Encrypted and Encoded Data: Updates to information are incremental in nature. This presents a problem in the presence of ciphertext, since we do not want to decrypt our data, apply modifications, and re-encrypt data for every slight change of the database. This would be a tremendous burden on OceanStore servers. Further, the difficulty of dealing with untrusted servers is that they must perform conflict resolution, logging, and merging entirely without access to cleartext. On first glance, this would appear to be an impossible problem.

The first of these can be tackled via a new branch of cryptography, called *Incremental Cryptoraphy*[9][10][11]. Some recent results include techniques which permit the generation of new versions of encrypted documents from older ones, given incremental changes in the cleartext. Similar techniques permit the incremental generation of new signatures from older ones. We hope to exploit both of these techniques.

The second issue is more problematic, i.e., the performing of database modifications (or conflict resolutions) directly on encrypted data, within the untrusted infrastructure. This is necessary, since we assume that our replicas are distributed widely – it would be unreasonable to bring them the client to be updated[10]. We have two options that we propose to handle this:

1. Make use of "tamper-resistant hardware"[80]. This is computational hardware that a user is willing to trust with his or her keys. One could imagine that such hardware consisted of complete systems on a chip which would destroy their contents if ever opened, and which included a cryptographic signature that could be verified by the user. To be effective, these devices would have to be sprinkled throughout the network
2. Make use of oblivious computation techniques for "function hiding" [67][68]. These techniques permit the execution of encrypted functions on encrypted data. "Encrypted functions" are functions which can be directly executed in untrusted domains, but whose functionality cannot be figured out by examining them (or for which it is as difficult to figure out what they are doing as it is to break an encryption key).

The first of these is straightforward, but far less desirable than the second[5]. Although applying oblivious functions techniques to general computations is computationally expensive, our hope is to find fast, specific instances of such functions for conflict resolution within an untrusted infrastructure.

Replication Mechanisms: How many replicas should be created, and how should they be distributed? Among other things, a minimum of three replicas yields the potential for quorum-based approaches to replica management[12]. Further, three replicas yields a bit of redundancy even after a single failure, i.e., during the time between when the first replica fails and when another replica can be created. Clearly, the more replicas that are created, the more survivability that is achieved. Unfortunately, this has great cost, both in storage space and in consistency overhead.

As demonstrated by the Intermemory project[36], much greater replication efficiency can be gained by using erasure codes; the latest of these codes (*turbo codes* [54]) can be encoded and decoded linear time. The Intermemory system survives hundreds of server failures without losing data by encoding each piece of information and scattering it among a thousand nodes. Amazingly, the total storage overhead of this redundancy is only a factor of five above the uncoded size of the information. As suggested in [17][18], erasure codes can also be used to reduce latency by scattering pieces of data to many sites and requesting from all of them simultaneously; this particular "digital fountain" approach seems quite attractive for bulk migration of information within the high-bandwidth backbone of

[10] In fact, requiring that replicas be brought to a single point in the network (the updating client) in order to be updated would violate requirements for reliability.

the internet. So, why not store all information within OceanStore in erasure coded form? Unfortunately, this would greatly complicate the update process. To perform a small modification to a piece of information that has been encoded in this way, we would have to collect information from many nodes, modify it, then scatter it to many nodes again. Despite this complication on updates, however, erasure codes are quite attractive for increasing the survivability of archival information[11].

Consequently, we propose a hybrid approach, in which archival versions of data are stored in coded, dispersed form, as are snapshots of these the latest copies of the data. Given such a snapshot and an appropriate "redo" log, we can recover from a great number of failures. In an untrusted infrastructure, widespread coding is one way to make sure that data survives failures or active attempts at data corruption. The exact frequency of snapshots is a design parameter (depending on the size of log space, commit rate of information, etc).

Index Searches on Encrypted Data: A final item of importance is the issue of performing index searches on encrypted data. Given the vast array of information that is likely to be stored in the OceanStore, we clearly want to perform associative searches on this data. In fact, the very premise of Presto-like interfaces[27] is that we locate data associatively, by its attributes, not by its position in some arbitrary hierarchy that existed at the time of its creation.

Given that fact that data objects are opaque from the standpoint of the infrastructure, we would appear to have an extremely difficult situation. What sort of search could we possibly hope to accomplish? In fact, there are at least four possibilities, each of which we are currently considering:

1. Generate additional "indices" at the time of creation or modification of information. This is possible because entities (clients) that manipulate data have the encryption keys and hence access to cleartext. Indices would consist of pairs of attributes and OIDs. We assume that each group of users shares the same index database, and that it is encrypted with a common key.
2. Include cleartext attributes with data objects (ala Presto[27]).
3. Make use of tamper-resistant hardware techniques, mentioned above.
4. Use techniques for function hiding such as described in [67][68]. Such techniques were discussed above; they have the potential to permit scan operations to be performed by untrusted hardware on encrypted data without revealing information. Alternatively, we could encrypt document attributes in a well defined position in every object, thereby simplifying the type of computation we require on encrypted data.

Since database style searches are extremely desirable for a system as large as OceanStore, some mechanism for this is important. Note, however, that there are many security vulnerabilities in exporting indices; we will have to be extremely careful here.

10.2.5 Introspective Computing Infrastructure: The Exploitation of Tacit Knowledge

OceanStore presents an exciting opportunity for dynamic optimization along several axes, such as the number of replicas, location of replicas, and the physical storage formats for these replicas. The intelligent cache management problem is a generalized version of what is often called *hoarding*[46][77] in mobile computing environments. In many of these systems, users are responsible for specifying which files should be cached on their laptops. In a system as large and varied as OceanStore, however, requiring significant user input to the hoarding process is completely implausible. To achieve a good level of performance at a reasonable cost on the user's part, some form of *introspection*[36] is required. Introspection is the intelligent observation and response to changing patterns of usage. Preliminary versions of data collection and analysis for exactly this purpose has been explored in the Seer project at UCLA[50][51]. Others have explored the use of adaptive monitoring to tune the performance of filesystems[62]. One of the key research questions to be resolved for OceanStore is what information should be collected and how it should be analyzed in order to best utilize the caching resources.

Figure 3 illustrates two principle components to the introspective analysis. Although this bears some similarity to the Seer mechanisms[51], it has much greater scope. The data monitoring components collect information from both the user's reference stream, from neighboring or related pools of information, and from incoming search requests. Monitoring components reside in all positions of the infrastructure. Among other things, every object carries with it a web of "tacit" information that links it with other objects that are related to it or have small "semantic distance"[51] relative to it. In OceanStore, objects can become related to one another through a number of events: access by the same user in the same task or session, simultaneous activity by a group of users in a region of the world, or relationship via client-server style communication. In addition, information is collected about the relationship between different users of replicas of the same object.

[11] This is, in fact, the intended application of the Intermemory System, mentioned above.

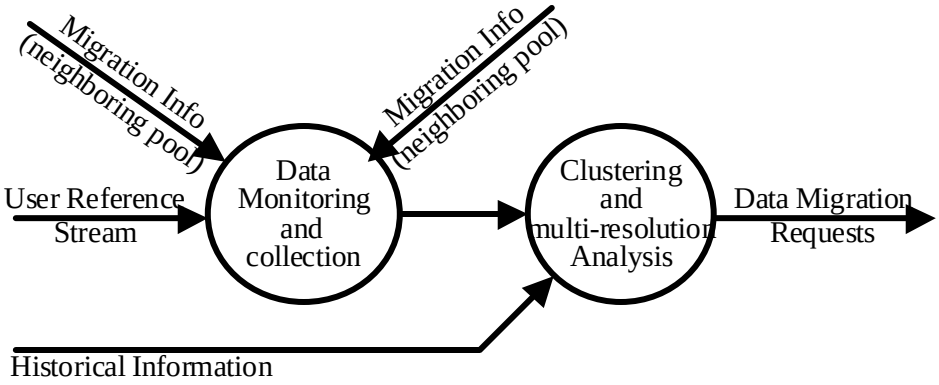


Figure 3: Introspective Analysis for Locality Management

Given both online information and historical information, the analysis component clusters semantic distance information to determine which objects are related to one another and should be colocated. A number of clustering algorithms could be brought to bear with this problem. In addition multiscale statistical analysis has shown promise in many venues (see, for instance [48]). This seems appropriate in the introspective domain as well, since persistent objects will be hierarchically organized by nature. We will use the multiscale analyses to construct phantom "directories" which are like normal directories, but which contain files that are related semantically. Note that phantom directories can be clustered into other phantom directories, etc. Such clusters bear much similarity to the "collections" mention in Section 10.2.3 in the context of Presto[27]: the introspective components can be thought of as continually generating new semantic structure on existing data.

With the clustering analysis in hand, the introspective components make recommendations for data migration. They attempt to colocate objects that form clusters and to migrate or replicate objects so that they move closer to consumers. In addition, they monitor objects that are being updated by multiple parties. If appropriate, they attempt to optimize network and coherence traffic by reducing the number of active replicas, centralizing the remaining replicas relative to consumers of data, and migrating primary replicas to portions of the network that are the source of most of the update traffic.

10.2.6 Data Economy: The "Glue" for an Information Infrastructure

We anticipate that each migrating object and search request will contain explicit, non-forgeable tags that are used for accounting. One possibility is that each utility provider keeps track of the amount of money "owed" to it by each other utility provider. This requires some form of secure software metering scheme, such as in [70], and a market which sets the value of resources. The exact form of this market is unknown at this time. The Data Economy is not a focus of the current proposal, but we do anticipate discussions and collaborations around this issue.

10.2.7 Related work

The Intermemory project[36] provides a read-only, archival storage infrastructure, with data spread among many servers world-wide in order to make sure that it is permanently available. Similar to the Intermemory system, we use erasure codes to ensure the survivability of archival versions of data and for snapshots. Use of erasure codes for latency reduction was suggested in [17][18]; such techniques may be used in OceanStore to quickly relocate replicas, since the snapshots are erasure coded anyway.

Many have worked on distributed file systems: NFS[76] and AFS[69] are well known distributed filesystems. These more traditional systems force data to be cached along the path between client and server; as a result, the servers become a bottleneck and a single point of failure. XFS[6] introduced "cooperative caching"[15] as a technique for collective sharing of memory for file caches. Weakly-consistent file systems, such as Coda[47], Bayou[63], Ficus[40][41] have targeted problems of transparency in the face of long periods of disconnection. Our philosophy of conflict resolution most closely resembles that of Bayou. However, all of these systems rely on secure servers and network connections to ensure the security of data. The systems do not exploit the advantages of the "mostly-connected" assumption, i.e., high-bandwidth backbone with multicast, bounded periods of inconsistency, and the potential for high-availability and disaster recovery. Recently, several researchers have explored the security

and performance implications of exploiting intelligence directly within disk devices[1][16][45][65] and with remote, network attached storage[35][34].

There is an extensive literature on multiprocessor coherence[53][28][29][33][37][19][52][3]. However, most multiprocessor coherence mechanisms are based on fixed-size coherence units. Many in the database community have tackled the problems of consistency in large distributed databases; to name a few: [12][32][60][73] and methods of relaxing coherence [39]. OceanStore builds on many of these techniques. In [39], Jim Gray warns of some of the dangers of update anywhere, anytime.

Name lookup services such as DNS[58], LDAP[43], and Globe[81] all resolve names into unique identifiers. However, these schemes impose fixed hierarchy on either the names or locations at which objects can be cached. Both DNS and LDAP assume relatively static mappings of names to identifiers as well as imposing hierarchy on the names themselves. Globe employs a relatively fixed, hierarchical search tree which has the property that some items may be physically close but distant in the search tree. Ocean Store provides much more flexibility in its location facilities and provides for a large, distributed collection of information pools to work together as a unified cache.

The idea of a data economy is not new, but perhaps applied on a much wider scale in OceanStore. Mariposa[73] was one of the first projects that we are aware of that proposed use of economic models to drive the use of resources in a distributed database. Most recently, the Telegraph project[78] has been exploring agoric distributed databases.

10.3 Educational Activities and the Impact of OceanStore

One of the key goals of the OceanStore project is to develop an infrastructure that is sufficiently robust that it can be the persistent storage medium of choice for researchers and students in the Berkeley Northside Systems Group[12]. Attending to the needs of a user group, even one that is relatively forgiving, has a tendency to quickly shake out many major problems.

10.3.1 Educational Impact of OceanStore

With this level of commitment, the educational implications of OceanStore are numerous. Omnipresent access to persistent information has the potential to revolutionize classroom interactions. For instance, a wide array of portable devices can be used to enhance interactions between students and faculty members by providing access to course materials and multimedia demonstrations. At Berkeley, this Principal Investigator is already using the Web to dispense assignments, announcements, and grades to his students. Many other forms of interaction become possible with direct, consistent, secure and continuous access to information. Direct broadcasts of lectures to portable devices are possible, as are more flexible methods of online test-taking or individualized assignments. Other more interesting possibilities include the real-time merging of notes being taken by students in the room (on their portable devices, such as CrossPads, that happen to be tied into the infrastructure).

More importantly, the OceanStore infrastructure has the potential to greatly impact kindergarten through 12th grade education. Students could have permanent storage repositories that stay with them for their whole grade-school experience and beyond. What if they were encouraged to keep every word that they have ever looked up or encountered in a personal database? Portable, tablet-style devices combined with OceanStore could encourage life-long journal writing and capturing of on-the-spot inspirations. Mathematics could be taught with actual, personalized information about the fuel consumption of parent's cars or the number of miles biked during a vacation in the mountains. With a persistent, highly-available infrastructure, students could learn how to collect and analyze information from real-life experiences, then access that information in the classroom. Further, digital libraries could be exported via the OceanStore infrastructure; these could be combined with tools that allowed students to link notes directly to library text and to see the notes taken by other students. This combination of standard texts and interactive note-taking would allow much more effective studying than traditional methods. It is the intention of the Principal Investigator to explore how local school districts can get access to portable devices as well as OceanStore storage space in order explore these possibilities.

There are two ongoing efforts at Berkeley that complement the OceanStore project. First, Berkeley has a project called "PostPC" which is exploring modes of computing beyond the "traditional" desktop metaphor. In support of this, we gave out Palm Pilots to all incoming computer science graduate students last year. It is our intention to continue programs such as this in successive years[13]. As a result, we have a number of students that are exploring communication infrastructure and applications for palm computing devices (e.g., the Palm Pilot) and laptops. Already, we are exploring physical communication infrastructures (such as infrared, short-distance wireless, etc)

[12] BNSG is comprised of a set of 10 faculty who collaborate together on a regular basis; it includes people with specialties in hardware design, computer architecture, operating systems, networking, and mobile computing.

[13] Whether it will be Palm Pilots, laptops, or other PDAs, etc, it hard to say yet.

throughout our research building to bring intermittent connectivity to these small devices. Hence, it is the intention of the Principal Investigator to make the OceanStore infrastructure available to these devices as soon as possible. Then, this new infrastructure will be used in a number of ways:

- Placing notes generated at student/faculty meetings within the OceanStore infrastructure.
- Exploring the use of NotePals[24] to collect and combine the notes students during lectures. The OceanStore data storage system would provide a natural framework around which to coordinate the combination of human-generated notes with actual copies of the papers or lecture notes.
- Use of on-line class materials during lectures.

Note that Berkeley has a very strong tradition of undergraduate involvement in research. One of the aspects of the PostPC project is that it lends itself to undergraduate experimentation. In fact, we may experiment with giving undergraduate students mobile computing devices (there is another pending proposal to do this); use of OceanStore for the dissemination of shared information is natural.

Second, the OceanStore infrastructure described within this proposal is one of the underpinnings of the multi-faculty *Endeavor* effort here at Berkeley. Endeavor is a new effort consisting of approximately 15 faculty members united by the common goal of driving computing to its next level: that of a vast, ubiquitous, unified federation of devices interacting with one another to achieve greater levels of human interaction and information reliability than ever seen before. Central to this effort is the presence of a unified data infrastructure that is present everywhere, all the time. With its many facets, the Endeavor project is focused on ways in which ubiquitous computing can improve the quality of life for everyone; in particular, one of the pieces of the Endeavor project is an intelligent classroom infrastructure, integrating numerous sources of information and electronic control to enhance instruction. I will be participating in this effort both as architect of the OceanStore infrastructure and as a generator of novel educational paradigms which result.

10.3.2 Current Educational Activities of the PI

The Principal investigator is currently involved in the instruction of both graduate and undergraduate students. The primary subject of these classes is Computer Architecture: the design and implementation of complete computer systems, with a focus on hardware techniques. As demonstrated by his PhD work (see discussion in Section 10.7), he is a firm believer in the "integrated systems viewpoint": that optimal computer systems can only be designed with an unfailing grasp of all levels of the system, from transistors, through operating systems, and into applications. This is a viewpoint that he reflects within his classes through examples from compiler technology, operating systems, and theoretical discussions. Teaching is somewhat of a passion with the PI, and students recognize this with high teaching ratings.

The graduate computer architecture class is lecture-oriented, with a research-level final project. The PI has recently revised this class to reintroduce research papers as a forum for discussing cutting edge concepts. Lecture periods were given to spirited discussion of research papers, and were quite successful. In the upcoming semesters, he intends to continue with new material and more interesting projects.

The undergraduate computer architecture class is also lecture-oriented, with a semester-long series of laboratory assignments. To this class, he has recently introduced new testing methodologies and increased the software component of the class (in the interest of designing for testability). Although the class is primarily oriented toward "traditional" RISC design techniques, the PI gave a few lectures in modern, out-of-order execution techniques, which resulted in final projects with working simulated processors employing these techniques.

10.4 Project Deliverables

There are three classes of deliverables for this proposal: techniques, prototypes, and validation frameworks. We will deal with each of these in turn. Although there may appear to be a large array of deliverables, we will provide a logical, incremental prototyping methodology in Section 10.5.

10.4.1 Theoretical Results and Algorithms

This project will have five different deliverables which are theoretical/algorithmic in nature:

- ***Techniques for data location and naming in fluid systems:*** We give a straw-man proposal in Section 10.2.3, but there remains more work to be done here.

- ***Introspective techniques for data distribution*:** We propose to explore techniques for continuous gathering of multi-scale information about data usage and employing this for intelligent prefetching and rearrangement of data, as described in Section 10.2.5.
- ***Global-scale mechanisms for replication and reliability:*** We propose to explore techniques for use of erasure codes and replication which is compatible with encryption and conflict-resolution mechanisms, and which achieve high-levels of reliability (or, conversely, low-correlated probability of failure for all replicas). This was described in Section 10.2.4.
- ***A language and protocol for conflict resolution in the presence of ciphertext*:** We propose to provide an operation-level interface for conflict resolution that is appropriate for wide-scale systems containing ciphertext, as in Section 10.2.4. This is anticipated to be the most challenging aspect of this proposal.
- ***A realistic model for an electronic data economy:*** We propose to explore viable techniques and rules to turn the technology of OceanStore into a viable utility. This issue was touched up on Section 10.2.6. We hope to leverage efforts underway in the Telegraph project[78] at Berkeley for handling data economies.

We expect all five of these aspects to be compatible with one another. Note that each of them represent well-defined partitions of the research space, we will present a schedule and partitioning of work in Section 10.5.

10.4.2 Prototypes

This project has three types of prototypes that we anticipate:

- ***Prototype Utility Providers (OceanStore Servers):*** Prototype development for the OceanStore utility is greatly aided by the presence of the Millennium infrastructure at Berkeley[57] and its related software infrastructure[71]. The Millennium testbed contains a thousand workstation nodes, organized as a cluster of clusters. This testbed will be physically distributed throughout the Berkeley campus, with individual clusters connected by gigabit-per-second networks. Hence, this provides an ideal physical infrastructure on which to test many of the OceanStore precepts: it will have numerous individual entities, connected by a high-bandwidth core network, and with lower-bandwidth connections to clients.
- ***Legacy OceanStore Clients and Applications:*** In addition, we will provide a file-system front-end for OceanStore that runs as a user-level NFS server under one or more UNIX platforms. This will demonstrate the viability of using OceanStore with legacy applications, and will aid in building a base of users. Ultimately, we hope to bootstrap Berkeley expertise (in students and staff) to provide native interfaces under Microsoft Windows, thereby extending our potential base of users even further; however, we do not promise this as a deliverable here.
- ***Native OceanStore Clients and Applications:*** The OceanStore client layer will consist of a user-level library talking to a local caching server. These two components will interact with one another to provide all of the client-side caching support, introspective monitoring and analysis, and conflict resolution interfaces. We hope to leverage the Endeavor and PostPC projects to develop OceanStore aware applications, as alluded to in Section 10.3.

The first of these comprises the OceanStore infrastructure, while the second and third support a base of applications.

10.4.3 Testing and Validation Framework

We anticipate that proofs of correctness will be available by design for some of the OceanStore components, in particular the *conflict resolution* and *replication* mechanisms. However, theoretical correctness and actual correctness (or stability) are rarely one and the same. Thus, one of the deliverables that we hope to provide is a complete testing and validation framework that acts to inject bad information into the system, cause utility provider nodes to fail, and otherwise impact an operating OceanStore system. Further, one important metric would conceivably be the degree to which OceanStore performance degrades gracefully under increased load or failures. Such a testing framework could be achieved through the use of ancillary processes, called “bashers” or “daemons” [20], whose sole task is to stretch the limits of the operating domain and check that the results are reasonable. Such antagonistic mechanisms are often used in the domain of hardware design, but are not often applied to large-scale systems. Thus, one of the results that we hope to achieve is a characterization of the reliability and performance of OceanStore under a variety of adverse circumstances.

10.4.4 Denouement

Assuming that the OceanStore project is successful, the Principal Investigator intends to seek additional funding for a wide-scale prototype of the OceanStore utility. This would include large servers on major Internet (and Internet-2)

backbones, as well as smaller caching servers. In addition, as mentioned in Section 10.3, OceanStore would be tightly integrated with the Endeavor project at Berkeley (a multi-faculty effort to explore ubiquitous, highly-available computing); one aspect of this would include demonstrations of wireless infrastructure backed by OceanStore caching servers.

10.5 Plan of Work

10.5.1 Two-phase implementation

By its very nature, the OceanStore project is amenable to a two-phase implementation approach, with an incremental series of technology demonstrations within each phase. These two phases could be roughly characterized as "technology synthesis" (infrastructure building) and "research results", although the introspective technologies span both categories. Phase-one demonstrations of technology include:

- **A fluid data location and naming service:** The techniques of Section 10.2.3 can lead to a large-scale read-only caching service. At this stage, initial prototyping of the client-side interfaces for a normal UNIX-style filesystem will provide instant access to legacy applications for experimentation.
- **Introspective technology for locality optimization:** Given a working naming and location service, some of the introspective technologies can be prototyped.
- **Replication and coding:** Also given a working naming and location service, techniques for replication and data coding can be prototyped.

These first three pieces can be exercised with simplified (read-only) versions of the client interfaces. This prototype would provide a powerful infrastructure for delivery of read-only content. The second phase would then include:

- **Techniques for conflict resolution with ciphertext:** This is one of the more difficult deliverables in this proposal. Hence, we anticipate that it will require the most "design and thinking time". Fortunately, this can also be added after the above three components have been explored. During this prototyping stage, we anticipate that the full operation-oriented interface will be complete. Hence, we anticipate the implementation of the native OceanStore operation-oriented interfaces and incorporation into PostPC and Expeditions infrastructures.
- **Data economy for the utility infrastructure:** This may be added at any time to the infrastructure. Our primary goal in including this is merely for completeness.

Thus, we have a well-defined progression of prototype infrastructures.

10.5.2 Proposed Schedule:

Year 1: The first year would see initial phase-I implementation as well as intense phase-II thinking:

- **Initial Design and implementation:** Design and implementation of data location technology, initial introspection techniques, and replication technologies. Initial, read-only UNIX interfaces.
- **Preliminary Analysis:** Preliminary analysis of initial prototype technologies.
- **Design:** Initial theoretical design of conflict resolution mechanisms in presence of ciphertext.

Year 2: Second year should see some limited deployment of phase-I prototype

- **Analysis:** Efficacy of phase-I prototype begins.
- **Design and Implementation:** Implementation of initial version of validation framework. More sophisticated introspective technologies and better replication policies. Phase-II technologies.
- **Educational:** Preliminary use in classroom activities for information access.
- **Publishing:** Aggressive publishing of papers

Year 3: Iteration and phase-II analysis

- **Design and Implementation:** Finishing of full OceanStore client interface libraries for one or more portable device. PostPC applications development.
- **Credible utility deployment:** Reasonable technology and mechanisms to make the OceanStore utility a realistic technology. Collaboration with economist and industrial leaders anticipated.
- **More Extensive Deployment:** Deployment on Millennium infrastructure throughout Berkeley.
- **Publishing and software release:** Publishing in several conferences as well aggressive proselytizing for use of software technologies.

Year 4: Wrap up (less well defined and dependent on success of previous years).

- **Wide scale deployment:** Assuming that all goes well, develop plans for a much larger scale deployment. Proselytizing of companies: convince them to support OceanStore technology?
- **Implementation:** Support for wider array of clients, more interesting software. Extensive collaboration with user-interface researchers.
- **Educational:** Student theses and ancillary work

10.6 Technology Transfer and Collaborations

The OceanStore infrastructure was designed with technology transfer as an important goal; in fact, the utility infrastructure was specifically constructed so that it could be comprised of resources from many different companies. Fortunately, Berkeley has a number of direct industrial collaborations with Pacific Bell, Sprint, IBM, Microsoft and others. We have a tradition of collaboration and technology transfer both on campus and at industrial retreats. As a result, it is the intention of the Principal Investigator to disseminate OceanStore protocols, methodologies, and software to the public as soon as they are ready. The OceanStore infrastructure is confederation of (potentially) antagonistic parties; hence, only a neutral party, such as Berkeley, has the potential to generate global support for the infrastructure.

In addition, the OceanStore project will build on a number of projects here at Berkeley. Association with the Endeavor project alone entails 12 – 15 collaborators, many of whom would be clients of OceanStore technology. To be more specific, however, both Joan Feigenbaum (AT&T labs) and Doug Tygar (UC Berkeley) have expressed interest in collaborating on the security implications of OceanStore, and in particular the issue of computing on encrypted data. Joe Hellerstein and Michael Franklin (both UC Berkeley) have expressed interest in collaborating on some of the database aspects of the OceanStore infrastructure. Joe Hellerstein also has an interest in agoric data federations (i.e., data economies). We anticipate much collaboration and sharing of techniques and research results in this domain. Further, Anthony Joseph and Eric Brewer (both UC Berkeley) have expressed interest in collaborating on some of the global systems issues of OceanStore. Thus, the architecture itself has the potential to spawn a number of creative and fruitful collaborations.

10.7 Prior Research and Educational Accomplishments

The Principal Investigator has coauthored publications in the area of multiprocessor design; a selected list of publications is included with the attached biographical sketch. His experience and knowledge spans a particularly wide range of topics, approaching multiprocessor design from the "integrated-systems" standpoint: operating systems, compilers, and hardware. While a graduate student at MIT, he was one of the chief architects as well as the principal implementor of the Alewife machine, a cache-coherent, shared-memory multiprocessor that integrates hardware support for shared-memory and message-passing communication [3]. During the Alewife project, this author proposed modifications to the industry-standard SPARC-V7 processor, some of which found their way into the SPARC-V9 specification. Further, he designed and implemented the one million-transistor, *Communications and Memory-Management Unit (CMMU)* that provided a communication infrastructure for Alewife. He was also responsible implementing roughly one third of the Alewife operating and runtime system (which was written from scratch). Alewife became operational in 1994 and subsequently developed a small but dedicated user community.

Alewife was one of the first machines to explore the consequences of exporting multiple communication styles directly to user level, combining the efficiency of direct access to hardware with OS-level protection. Ultimately, this was the topic of the Principal Investigator's doctoral dissertation [49]. In designing hardware and software systems for Alewife, the he became intimately familiar with the extensive literature of shared-memory access models (which bare striking similarity to database consistency models). From 1987 to 1989, he was on the staff of Project Athena at MIT, and as such had occasion to grapple with distributed file systems and related security concerns. Further, during the following decade, he worked as a consultant for CLAM associates, a company that provides high-availability clustering products for large corporations. Thus, this author is qualified to explore hardware and software tradeoffs in the highly-available, persistent, secure, flexible utility model of OceanStore.

Awards and Distinctions: The Principal Investigator has received the following distinctions:

- Okawa Foundation award for faculty development. September 1998.
- MIT Nomination for ACM Dissertation Award, July 1998.
- The George M. Sprowls Award best PhD thesis in EECS at MIT. July 1998.
- IBM Graduate Fellowship. September 1992 – June 1994
- Best Paper: International Conference on Supercomputing (ICS). July 1993

UNIVERSITY OF CALIFORNIA, BERKELEY

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO

SANTA BARBARA • SANTA CRUZ

ELECTRICAL ENGINEERING & COMPUTER SCIENCES
231 CORY HALL
BERKELEY, CA 94720-1770
(510) 642-0253 / 2845 (fax)

July 22, 1999
File Number 99-230

Program Officer
CISE/NCR
National Science Foundation
4201 Wilson Boulevard
Arlington, VA 22230

To Whom It May Concern:

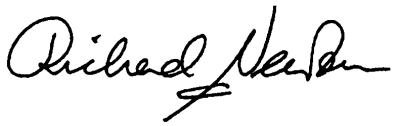
In this memorandum, I document the significant investment the Department of Electrical Engineering and Computer Sciences has made in the career development of Professor John Kubiatowicz. The University of California, Berkeley has an extensive mentoring program for young faculty, requiring a mentor to be identified at the time of appointment. Professor Kubiatowicz's senior faculty mentor is Professor Dave Patterson. Furthermore, Professor Kubiatowicz has received the following generous support to establish a vigorous and significant research activity in his research specialty:

- The normal faculty teaching load for his first year was reduced to 50%.
- A startup research fund of $25,000 per year for two years, to be used for equipment, software, travel, student support or any other purpose in support of research; three months of summer salary support for his first two years; funding for two research assistants for two years.
- Three personal computers for his office and his students, and fully paid relocation expenses.

Kubiatowicz has been enthusiastically supported by the department to tightly integrate his teaching and research programs. He has already produced good results: a successful revision of the graduate Computer Architecture class which produced a number of interesting class research projects; and the formation of a new research group into the wide-ranging effects of *introspection* on everything from microprocessor design to global-scale networking services.

Professor Kubiatowicz joined this department as his first full-time tenure track position effective 1 July 1997. I have read his Career Development Plan, and speak for all of my colleagues in stating that I fully and enthusiastically endorse it. Personally, I believe John Kubiatowicz is one of our real rising stars. His work is innovative, of high risk and of potentially very high reward in an area that is bound to be of increasing importance in the years ahead.

Yours truly,



A. Richard Newton
Professor, and
Chair, Department of Electrical Engineering and Computer Science

11. References

[1] ACHARYA, A., UYSAL, M., SALTZ, J. "Active Disks: Programming Model, Algorithms, and Evaluation," *Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems, 1998*

[2] ADVE, S. AND HILL, M. "Weak-Ordering: A New Definition," Proceedings of the 17th International Symposium on Computer Architecture, June 1990

[3] AGARWAL, A., BIANCHINI, R., CHAIKEN, D., JOHNSON, K., KRANZ, D., LIM, B-H., MACKENZIE, K., AND YEUNG, D. "The Alewife Machine: Architecture and Performance," In *Proceedings of the 22nd International Symposium on Computer Architecture*, June 1995.

[4] AKAMAI. See, for instance, http://www.akamaitech.com

[5] ANDERSON, R., KUHN, M. "Tamper Resistance – a Cautionary Note," *The Second USENIX Workshop on Electronic Commerce*, November, 1996

[6] ANDERSON, T., DAHLIN, M., NEEFE, J., ROSELLI, D., PATTERSON, D., AND WANG, R. "Serverless Network File Systems," *Procceedings of the 15th ACM Symposium on Operating System Principles*, December 1995.

[7] ARGON, C.R, SEIDEL, R.G. "Randomized search trees," *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science* (FOCS). 1978

[8] ARPACI-DUSSEAU, R.H., ANDERSON, E.A., TREUHAFT, N., CULLER, D.E., HELLERSTEIN, J.M., PATTERSON, D.A., YELICK, K. "Cluster I/O with River: Making the Fast Case Common," *To appear in IOPADS '99.*

[9] BELLARE, M., GOLDREICH, O., AND GOLDWASSER, S. "Incremental Cryptography: The Case of Hashing and Signing," *Advances in Cryptography — Crypto '94 Proceedings, Lecture Notes in Computer Science* Vol 839, Springer-Verlag, 1994

[10] BELLARE, M., GOLDREICH, O., GOLDWASSER, S. "Incremental Cryptography and Applications to Virus Protection," *Proceedings of the 27th ACM Symposium on tyhe Theory of Computing*, May 1995

[11] BELLARE, M., and Micciancio, D. "A New Paradigm for collision-free hashing: Incrementality at reduced cost," *Advances in Cryptology — Eurocrypt 97 Proceedings, Lecture Notes in Computer Science Vol 1233.* Springer-Verlag, 1997

[12] BERNSTEIN, P.A., HADZILACOS, V., AND GOODMAN, N. "Concurrency Control and Recovery in Database Systems," Chapter 8. Addison-Wesley Publishing Company, 1987.

[13] BLAZE, M. "A Cryptographic File System for Unix," *First ACM Conference on Communications and Computing Security*, November 1993

[14] BLAZE, M. "Key Management in an Encrypting File System," *Proceedings Summer 1994 USENIX Technical Conference*, June 1994

[15] BLOOM, B. "Space/Time Tradeoffs in Hash Coding with Allowable Errors," *Communications of the ACM* 13, 7 (July 1970)

[16] BROWN, A., OPPENHEIMER, D., KEETON, K., THOMAS, R., KUBIATOWICZ, J., AND PATTERSON, D.A. "ISTORE: Introspective Storage for Data-Intensive Network Services," *Proceedings of the 7th Workshop on Hot Topics in Operating Systems* (HotOS-VII), March 1999.

[17] BYERS, J.W., LUBY, M., MITZENMACHER, M. "Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads," *International Computer Science Institute (ICSI)* Technical Report TR-98-021, July 1998

[18] BYERS, J.W., LUBY, M., MITZENMACHER, M., REGE, A. "A Digital Fountain Approach to Reliable Distribution of Bulk Data," *Proceedings of ACM SIGCOMM '98,* September 1998

[19] CARTER, J.B., BENNETT, J.K, AND ZWAENEPOEL, W. "Implementation and Performance of Munin," *Proceedings of the 10th Symposium on Operating Systems Principles* (SOSP) 1991.

[20] CLARK, DOUGLAS W. "Large-Scale hardware Simulation: Modeling and Verification Strategies," *CMU Computer Science: A 25th Anniversary Commemorative"* (R.F. Rashid, ed.) Invited Paper. ACM Press/Addison-Wesley, 1991

[21] CODD, E.F. "A relational Model of Data for Large Shared Data Banks," *CACM 13:6*, June 1970

[22] CZERWINSKI, S., ZHAO, B.Y., HODES, T.D., JOSEPH, A.D., AND KATZ, R.H. "An Architecture for a Secure Service Discovery Service," *Proceedings of the 5th Annual International Conference on Mobile Computing and Networks* (MobiCom '99), August 1999.

[23] DAHLIN, M., ANDERSON, T., PATTERSON, D., AND WANG, R. "Cooperative Caching: Using Remote Client Memory to Improve File System Performance," *Proceedings of the 13tth ACM symposium on Operating Systems Design and Implementation*, November 1994

[24] DAVIS, R.C., LANDAY, J.A., CHEN, V., HUANG, J., LEE, R.B., LI, F., LIN, J., MORREY, C.B. III, SCHLEIMER, B., PRICE, M., AND SCHILIT, B.N. "NotePals: Lightweight Note Sharing by the Group, for the Group," *Proceedings of CHI '99*, May 1999.

[25] DEMERS, A.J., PETERSEN, K., SPREITZER, M.J., TERRY, D.B., THEIMER, M.M., AND WELCH, B.B. "The Bayou Architecture: Support for Data Sharing among Mobile Users," *Proceedings of the Workshop on Mobile Computing Systems and Applications*, Santa Cruz, California, December 1994.

[26] DESHPADE, A.V., Olston, C. "Asopiram: Simulating an Agoric Distributed DBMS," *Berkeley Graduate Database Class Project*, Spring 1999

[27] DOURISH, P. EDWARDS, W.K., LAMARCA, A. AND SALISBURY, M. "Presto: An Experimental Architecture for Fluid Interactive Document Spaces," To appear in *ACM Transactions on Computer-Human Interaction.* 1999

[28] DUBOIS, M. AND SCHEURICH, C. "Memory Access Dependencies in Shared-Memory Multiprocessors," *IEEE Transactions on Software Engineering*, 16(6):660-673, June 1990

[29] DUBOIS, M., SCHEURICH, C., AND BRIGGS, F. "Memory Access Buffering in Multiprocessors," *Proceedings of the 13th Annual International Symposium on Computer Architecture*, pages 434-442, June 1986

[30] EDWARDS, W.K., MYNATT, E., PETERSEN, K., SPREITZER, M., TERRY, D., AND THEIMER, M. "Designing and Implementing Asynchronous Collaboration Applications with Bayou," *Proceedings of the 10th ACM Symposium on User Interface Software and Technology*, October 1997

[31] FAN, L., CAO, P., ALMEIDA, J., AND BRODER, A. "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *Proceedings of SIGCOMM '98*, 1998

[32] FOX, A. *et all*, "Cluster-Based Scalable Network Services," *Proceedings of the 1997 Symposium on Operating Systems Principles*, October 1997.

[33] GHARACHORLOO, K., LENOSKI, D., LAUDON, J., GIBBONS, P., GUPTA, A., AND HENNESSY, J. "Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors," *Proceedings of the 17th International Symposium on Computer Architecture*, May 1990.

[34] GIBSON, G.A., NAGLE, D.F., AMIRI, K., BUTLER, J., CHANG, F.W., GOBIOFF, H., HARDIN, C., RIEDEL, E., ROCHBERG, D., ZELENKA, J. "A Cost-Effective, High-Bandwidth Storage Architecture," *Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems, 1998*

[35] GIBSON, G.A., NAGLE, D.F., COURTRIGHT II, W., LAQNZA, N., MAZAITIS, P., UNANGST, M., ZELENKA, U. "NASD Scalable Storage Systems," *Proceedings of USENIX 1999, Linux Workshop,* June 1999

[36] GOLDBERG, A.V. AND YIANILOS, P.N. "Towards an Archival Intermemory," *Proc. IEEE International Forum on Research and Technology Advances in Digital Libraries* (ADL'98)

[37] GOODMAN, J.R. "Cache Consistency and Sequential Consistency," Technical Report no. 61, SCI Committee, March 1989.

[38] GRAY, J. *et al.* "Granularity of Locks and Degrees of Consistency in a Shared Data Base," *IFIP Working Conference on Modelling of Data Base Management Systems.* 1997. AFIPS Press.

[39] GRAY, J., HELLAND, P., O'NEIL, P. SHASHA, D. "The Dangers of Replication and a Solution," *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1996

[40] GUY, R.G., HEIDEMANN, J.S., MAK, W., PAGE, T.W. JR., POPEK, G.J., AND ROTHMEIER, D. "Implementation of the Ficus replicated file system," *Proceedings Summer USENIX Conference*, June 1990

[41] GUY, R.G., REIHLER, P., RATNER, D. GUNTER, M., MA, W., AND POPEK, G. "Rumor: Mobile Data Access Through Optimistic Peer-to-Peer Replication," *ER'98 Workshop on Mobile Data Access.* 1998

[42] HAERDER, T., AND REUTER, A. "Principles of Transaction-Oriented Database Recovery," *Computing Serveys, 15(4);* Association of Computing Machinery, 1983
[43] HOWES, T.A. "The Lightweight Directory Access Protocol: X.500 Lite," Technical Report 95-8, Center for Information Technology Integration, U.Mich., July 1995.
[44] JOSEPH, A.D., TAUBER, J.A., AND KAASHOEK, M.F. "Building Reliable Mobile-Aware Applications using the Rover Toolkit," *Proceedings of the Second ACM International Conference on Mobile Computing and Networking* (MobiCom'96). November 1996.
[45] KEETON, D., PATTERSON, D.A., AND HELLERSTEIN, J.M., "A Case for Intelligent Disks (IDISKS)," *SIGMOD Record, Vol 27, No. 3, September, 1998*
[46] KISTLER, J.J. AND SATYANARAYANAN, M. "Disconnected Operation in the Coda File System," *ACM Transactions on Computer Systems*, 10(1), February 1992.
[47] KISTLER, J.J., KUMAR, P., OKASAKI, M.E., SIEGEL, E.H., STEERE, D.C. "Coda: A Highly Available File System for a Distributed Workstation Environment," *IEEE Transactions on Computers,* April 1990, Vol. 39, No. 4
[48] KRIM, H. WILLINGER, W., JUDITSKI, A., AND TSE, D. (editors) "Special Issue on Multiscale Statistical Signal Analysis and its Applications," *IEEE Transactions on Information Theory,* April 1999.
[49] KUBIATOWICZ, J. "Integrated Shared-Memory and Message Passing Communication in the Alewife Multiprocessor," PhD thesis, Massachusetts Institute of Technology, January 1998.
[50] KUENNING, G.H., REIHER, P., AND POPEK, G.J., "Experience with an Automated Hoarding System," *Personal Technologies,* 1(3), September 1997.
[51] KUENNING, G.H. AND POPEK, G.J. "Automated Hoarding for Mobile Computers," *Proceedings of the 16th ACM Symposium on Operating Systems Principles,* (SOSP-16), October 1997.
[52] KUSKIN, J. *et al.* "The Stanford FLASH Multiprocessor (87 kB)," *In Proceedings of the 21st International Symposium on Computer Architecture*, April 1994.
[53] LAMPORT, L. "How to Make a Multiprocessor Computer that Correctly Executes Multiprocessor Programs," *IEEE Transactions on Computers,* C-28(9):241-248, September 1979
[54] LUBY, M.G., MITZENMACHER, M., SHOKROLLAHI, M.A., SPIELMAN, D.A., STEMANN, V. "Practical Loss-Resilient Codes," *Proceedings of the 29th Symposium on the Theory of Computing* (STOC), May 1997.
[55] MAKERT, L.F., LOHMAN, G.M., "R* Optimizer Validation and Performance Evaluation for Distributed Queries," *Proceedings of the 12th International Conference on VeryLarge Data Bases,* August 1986
[56] MARTÍNEZ, C. AND ROURA, S. "Randomized Binary Search Trees," *Journal of the ACM,* 45(2), March 1998
[57] MILLENIUM. *Berkeley Millennium Project.* http://now.cs.berkeley.edu/Millennium/ January 1998
[58] MOCKAPETRIS, P.V., AND DUNLAP, K. "Development of the Domain Name System," *Proceedings of SIGCOMM '88",* August 1988.
[59] MOHAN, C., HADERLE, D., LINDSAY, B., PIRAHESH, H., AND SCHWARZ, P. "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," *ACM Transactions on Database Systems,* 17(1), 1992
[60] MOHAN, C., LINDSAY, B., AND OBERMARCK, R. "Transaction Management in the R* Distributed Database Management System," *ACM Transactions on Database Systms,* 11(4), 1986
[61] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, NIST FIPS PUB 186, "Digital Signature Standard," *Us Department of Commerce,* May 1994
[62] NEEFE MATTHEWS, J., ROSELLI, D., COSTELLO, A.M., WANG, R.Y., AND ANDERSON, T.E. "Improving the Performance of Log-Structured File Systems with Adaptive Methods," *Proceedings of the 16th ACM Symposium on Operating System Principles,* October, 1997
[63] PETERSEN, K., SPREITZER, M.J., TERRY, D.B., THEIMER, M.M., AND DEMERS, A.J., "Flexible Update Propagation for Weakly Consistent Replication," *Proceedings of the 16th ACM Symposium on Operating Systems Principles* (SOSP-16), October 1997.
[64] REIHER, P., HEIDEMANN, J., RATNER, D., SKINNER, G., AND POPEK, G. "Resolving file conflicts in the Ficus file system," *Proceedings Summer USENIX Conference,* June 1994

[65] RIEDEL, E., GIBSON, G.A., AND FALOUTSOS, C. "Active Storage For Large-Scale Data Mining and Multimedia," *Proceedings of the 24th International Conference on Very Large Databases*, Aug 1998
[66] RIVEST, R. "The MD5 message-digest algorithm," *IETF Network Working Group*, RFC 1321. April 1992
[67] SANDER, T. AND TSCHUDIN, C.F. "Towards Mobile Cryptography," *Proceedings of Security and Privacy*, May 1998
[68] SANDER, T. AND TSCHUDIN, C.F. "Protecting Mobile Agents Against Malicious Host," In the volume "Mobile Agents and Security" in the series Springer Lecture Notes in Computer Science 1419.
[69] SATYANARAYANAN, M. "Scalable, Secure, and Highly Available Distributed File Access," *IEEE Computer*, May 1990, Vol. 23, No. 5
[70] SCHNEIER, B. AND KELSEY, J. "A Peer-to-Peer Software Metering System," *The Second USENIX Workshop on Electronic Commerce Proceedings*, USENIX Press, November 1996, pp. 279-286.
[71] SIMMILLENNIUM. *SimMillennium Project.* http://www.cs.berkeley.edu/~culler/SimMillennium/
[72] STONEBRAKER, M. "The Design of the POSTGRES Storage System," *Proceedings of the 13th International Conference on Very Large Data Bases.* September 1987
[73] STONEBRAKER, M., AOKI, P.M., PFEFFER, A., SAH, A., SIDELL, J., STAELIN, C., AND YU, A. "Mariposa: A Wide-Area Distributed Database System," *VLDB Journal 5(1).* January 1996
[74] STONEBRAKER, M. DEVINE, R., KORNACKER, M., LITWIN, W., PFEFFER, A., SAH, A., AND STAELIN, C. "An Economic Paradigm for Query Processing and Data Migration in Mariposa," *Proceedings of 3rd International Conference on Parallel and Distributed Information Systems*, September 1994
[75] STONEBRAKER, M. AND HELLERSTEIN, J.M., EDS. "Readings in Database Systems," Morgan-Kaufmann publishers, 1998.
[76] SUN MICROSYSTEMS, "NFS: network file system protocol specification," Internet RFC 1094, Internet Engineering Task Force, March 1989
[77] TAIT, C.D., LEI, H., ACHARYA, S., AND CHANG, H. "Intelligent file hoarding for mobile computers," *Proceedings of MobiCom '95: The First International Conference on Mobile Computing and Networking*, November 1995.
[78] TELEGRAPH. Telegraph: A Universal System for Information. *Berkeley Telegraph Project*: http://db.cs.berkeley.edu/telegraph/
[79] TERRY, D.B., PETERSEN, K., SPREITZER, M.J., AND THEIMER, M.M. "The Case for Non-transparent Replication: Examples from Bayou," *IEEE Data Engineering*, December 1998.
[80] TYGAR, J.D., AND YEE, B. "Dyad: A System for Using Physically Secure Coprocessors," in *Technological Strategies for Protection of Intellectual Property in the Networked Multimedia Environment.* Harvard University Press and the Interactive Multimedia Association, 1994
[81] VAN STEEN, M. HAUCK, F., HOMBURG, P., AND TANNENBAUM, A. "Locating objects in wide-area systems," *IEEE Communications Magazing*, January 1988
[82] WÄCHTER, H., AND REUTER, A. "The ConTract Model," *Database Transaction Models for Advanced Applications.* Elmagarmid, A. (Ed.), Morgan Kaufmann Publishers (1991).
[83] WEISER, MARK, "The Computer for the Twenty-First Century," *Scientific American*, September 1991.

12. Biographical Sketch of Principal Investigator, John D. Kubiatowicz

12.1 Vitae

Assistant Professor
University of California, Berkeley
CS Division, EECS Department
675 Soda Hall #1776
Berkeley, CA 94720-1776 USA

work: (510) 643-6817
fax: (510) 643-7352
home: (510) 540-7168
email: kubitron@cs.berkeley.edu
http://www.cs.berkeley.edu/~kubitron

.Education

Massachusetts Institute of Technology **Cambridge, MA**
PhD Degree in Electrical Engineering and Computer Science ..... January 1998
Dissertation: *Integrated Shared-Memory and Message-Passing Communication in the Alewife Multiprocessor*
Advisor: Anant Agarwal
Minor in Physics (quantum mechanics and field theory)

12.2 Publications

Publications Related to This Proposal

BROWN, A., OPPENHEIMER, D., KEETON, K., THOMAS, R., KUBIATOWICZ, J., AND PATTERSON, D.A. "ISTORE: Introspective Storage for Data-Intensive Network Services" *Proceedings of the 7th Workshop on Hot Topics in Operating Systems* (HotOS-VII), March 1999.

MACKENZIE, K., KUBIATOWICZ, J., AGARWAL, A., AND KAASHOEK, F. "Exploiting Two-Case Delivery for Fast Protected Messaging" *Proceedings of the 4th Int'l Symposium on High Performance Computer Architecture*, February 1998

YEUNG, D., KUBIATOWICZ, J., AND AGARWAL, A. "MGS: A Multigrain Shared Memory System", *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, May 1996

BREWER, E.A., CHONG, FT. LIU, L.T., SHARMA, S.D., AND KUBIATOWICZ, J. "Remote Queues: Exposing Message Queues for Optimization and Atomicity". *Proceedings of Symposium on Parallel Algorithms and Architectures*, 1995.

CHAIKEN, D. KUBIATOWICZ, J.D., AND AGARWAL, A. "LimitLESS Directories: A Scalable Cache Coherence Scheme." *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, April 1991

Other Significant Publications

CHONG, F.T., BARUA, R., DAHLGREN, F., KUBIATOWICZ, J.D., AGARWAL, A. "The Sensitivity of Communication Mechanisms to Bandwidth and Latency". *Proceedings of the 4th International Symposium on High Performance Computer Architecture*, February 1998

AGARWAL, A., BIANCHINI, R., CHAIKEN, D., JOHNSON, K., KRANZ, D., KUBIATOWICZ, J., LIM, B-H., MACKENZIE, K., AND YEUNG, D. "The Alewife Machine: Architecture and Performance." In *Proceedings of the 22nd International Symposium on Computer Architecture*, June 1995.

KUBIATOWICZ, J. AND AGARWAL, A., "The Anatomy of a Message in the Alewife Multiprocessor", *Proceedings of the International Conference on Supercomputing*, 1993.

AGARWAL, A., KUBIATOWICZ, J., KRANZ, D., LIM, B-H, YEUNG, D., AND D'SOUZA, G. "Sparcle: An Evolutionary Processor Design for Large-Scale Multiprocessors" *IEEE Micro*, June 1993

KUBIATOWICZ, J. CHAIKEN, D., AND AGARWAL, A. "Closing the Window of Vulnerability in Multiphase Memory Transactions" *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, October 1992.

12.3 Collaborators in the Last 48 Months Not Listed Above

Massachusetts Institute of Technology: Anant Agarwal, M.Frans Kaashoek
UC Berkeley: Prof. Eric Brewer, Prof. David Culler, Prof. Joseph Hellerstein, Prof. Anthony Joseph, Prof. Randy Katz, Prof. Kathy Yelick,

12.4 PhD Students Advised

Sharad Agarwal (second year), Victor Wen (first year), Co-Advised David Oppenheimer and Aaron Brown (both in third year).

12.5 PhD Advisor

Prof. Anant Agarwal, MIT