# Glacier

## Highly durable, decentralized storage despite massive correlated failures

Andreas Haeberlen, Alan Mislove, Peter Druschel
Department of Computer Science, Rice University

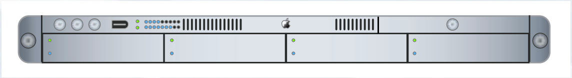
# Overview

- Classical Concept
- Glacier
  - Concept
  - Distribution of data
  - Recovery
  - Environment
  - Security
- ePost experiment
  - Setup
  - Results
- Glacier for a Real File Server
- Competitors?
  - Academic
  - Commercial/Industrial

# «Classical» System

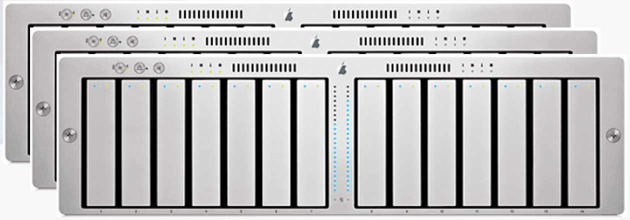
pool of workstations
(different OS)

central fileserver



with primary store with
redundancy (RAID-array)

central
backup store (one or more)



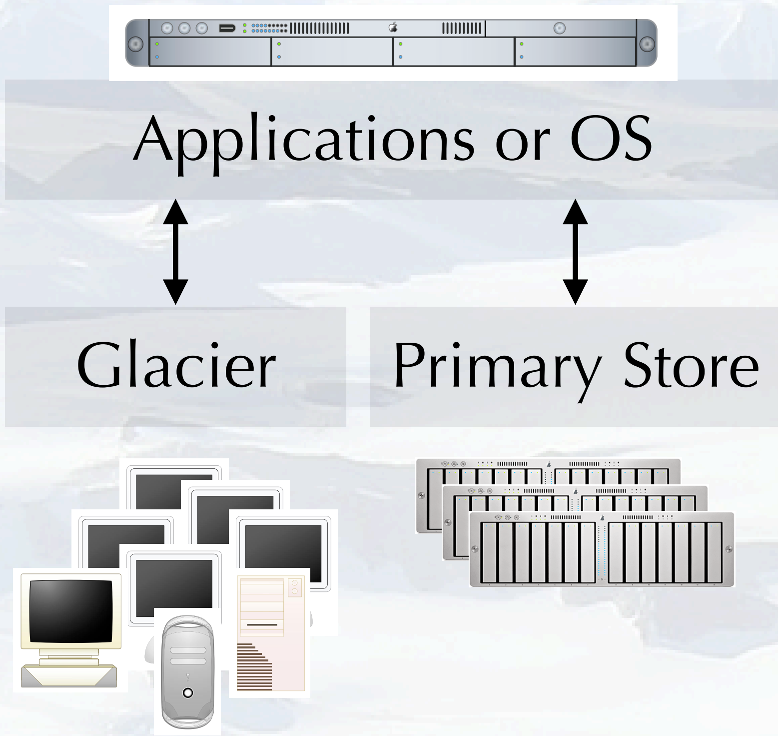
## Problems/Disadavantages

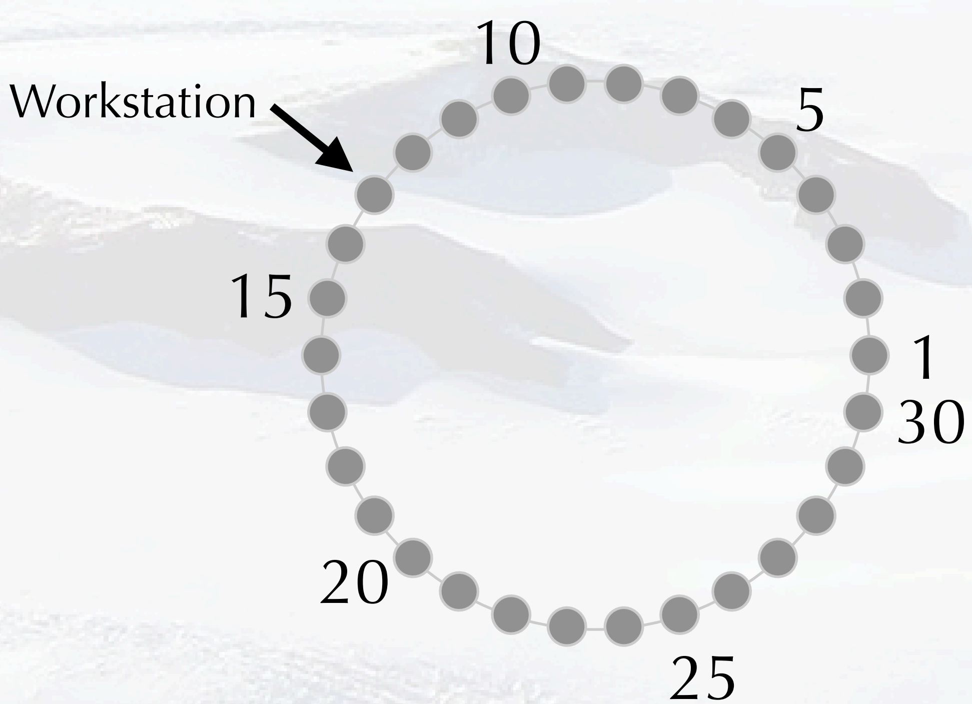
- Fileserver, primary store and backup store run the same OS → same vulnarabilities
- Additional redundancy through more separated backup stores is expensive
- Disk capacity of workstations is huge and underused (often up to 90% free)
  → could be used for decentralized storage

# Overview

- Classical Concept
- Glacier
  - Concept
  - Distribution of data
  - Recovery
  - Environment
  - Security
- ePost experiment
  - Setup
  - Results
- Glacier for a Real File Server
- Competitors?
  - Academic
  - Commercial/Industrial

# Glacier




- Glacier assigns every object (e.g. file) to store a key *k*.
- Every object is recoded into $N$ fragments so that $r<N$ of them contain sufficient information to restore the object.
- Every fragment is identified by a fragment key $(k,i,v)$ where $i$ is a fragment index and $v$ a version number.

- Every participating workstation is assigned a node id.
- The node id space is circular.

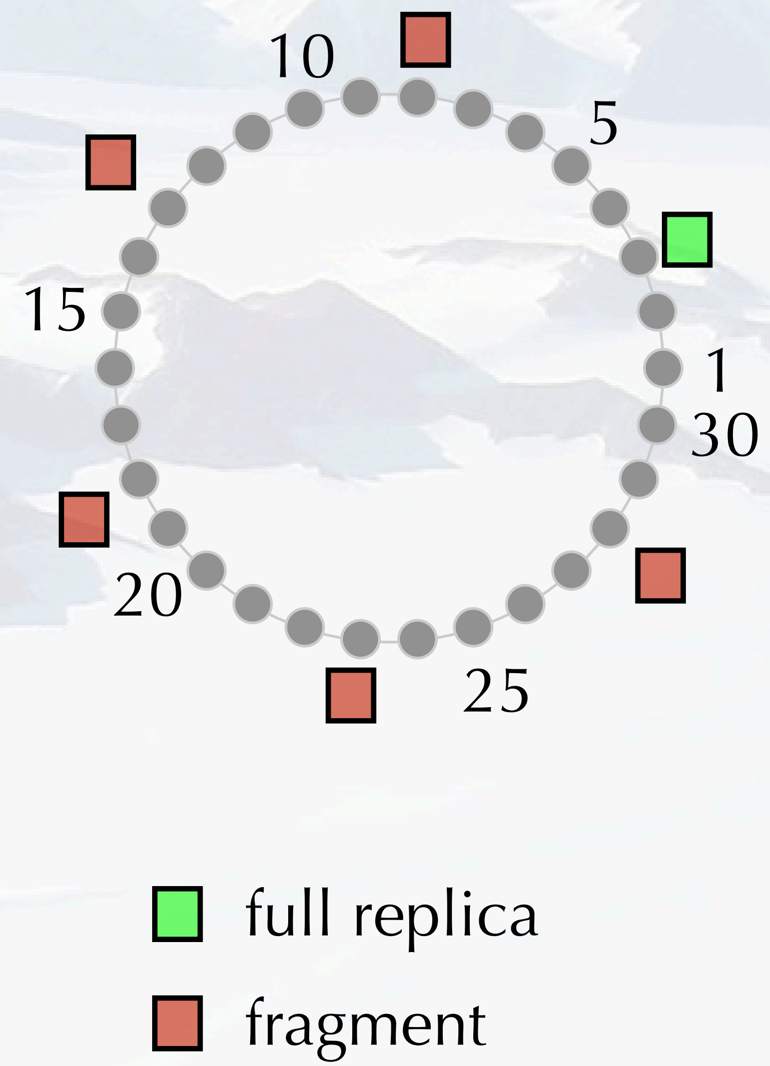
→ The fragments are distributed on the workstations.

# Fragment Placement

- The fragment placement function *P(k,i,v)* distributes the fragments on the workstations:

$$P(k,i,v) = k + \frac{i}{N+1} + H(v)$$

At postion *k* a full replica of object *k* is stored. The *n* fragments are stored at equidistant points in the id space.



# Offline Workstations

## Problem

Certain replicas and/or fragments cannot be stored.

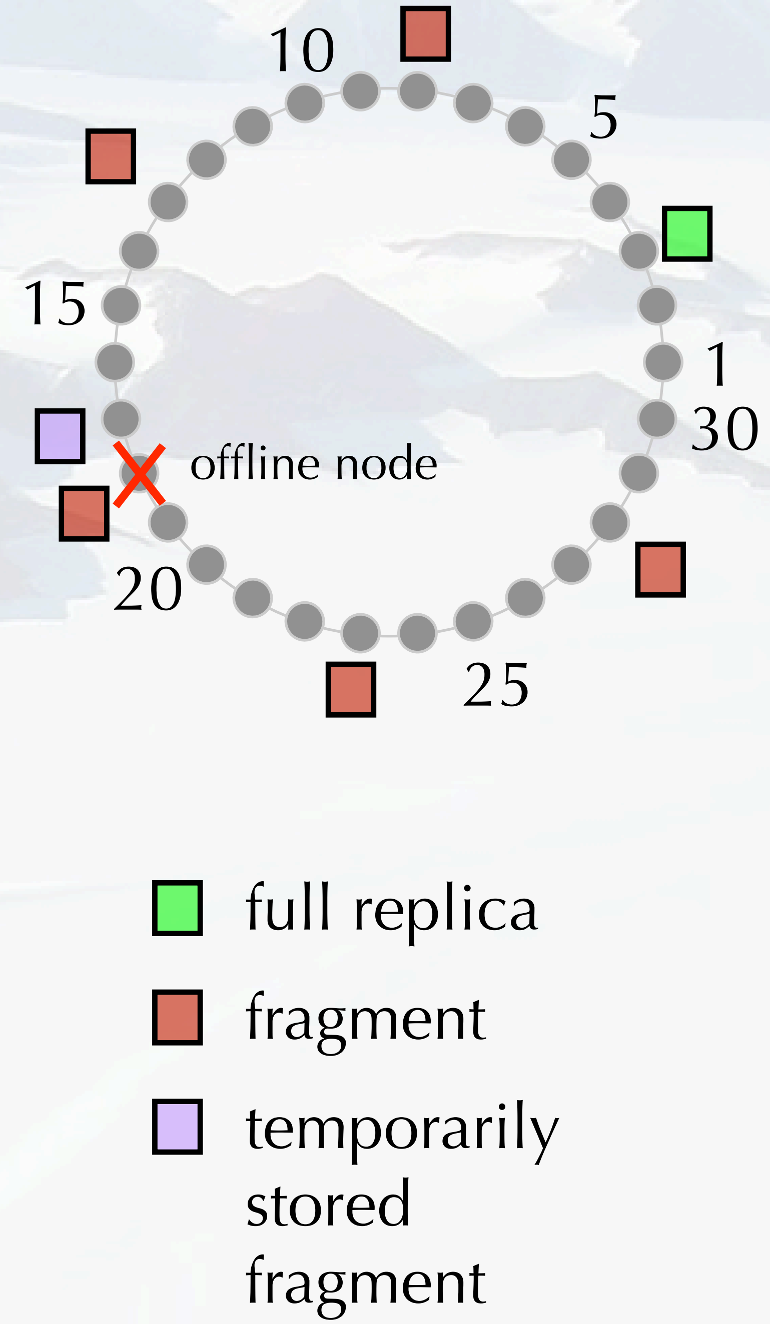
## Solution

New Objects

- Glacier sends probe messages to find nodes near (regarding the node id) the missing node.
- The replicas and/or fragments are stored on one of these neighbouring nodes.

Existing Objects

- Regular maintenance task on every node which copies objects to «correct» nodes as soon as these are again online.

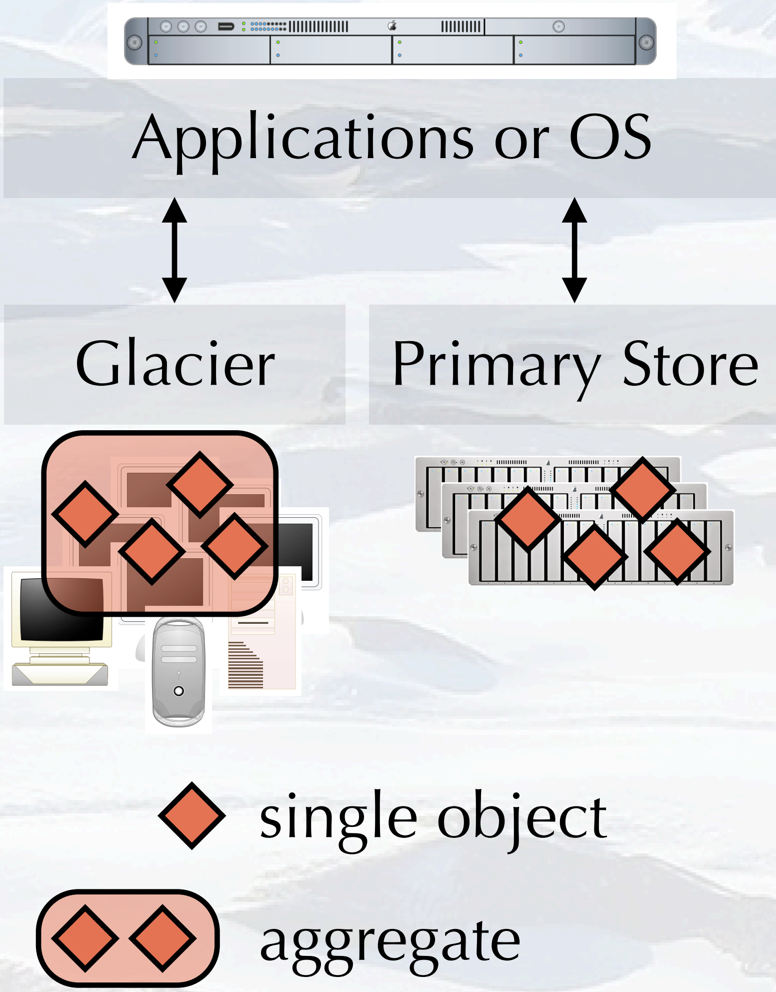


# Garbage Collection

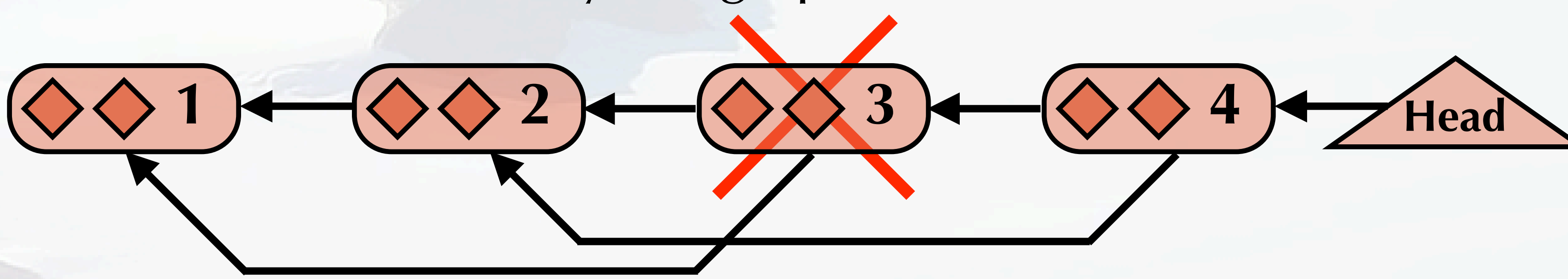
- No delete function implemented due to security reasons.
- Instead every object has a lease period $l$:
  - Lease period has to be renewed by owner (appliction or OS).
  - Every version of an object is identified by a version number $v$ and is stored independently.
    Frequent changes → massive storage use.
  - If lease period has expired, storage can be reclaimed.

$$P(k, i, v) = k + \frac{i}{N+1} + H(v)$$

# Aggregation




- Storing every single object (file) independently leads to high overhead.
- Glacier forms aggregates of groups of objects at the primary store:
  - Aggregates are distributed on to the nodes.
  - Aggregates contains several keys of other previously stored aggregates to facilitate recovery → directed acyclic graph.




The indegree of every aggregate is kept above a fixed number $d_{min}$.

  - Head contains link to newest aggregate.
  - An aggregate directory mapping objects (files) to aggregates is kept at the primary store.

# Recovery

## Case 1: Massive failures of nodes or connectivity

but primary store still running

- Normal maintenance tasks assure that fragments are distributed as soon as the nodes are reachable again.
- Every node only distributes a limited number of fragments at the same time to prevent congestion on the network.

## Case 2: Massive failures including primary store

- Waiting until connecitivity between the remaining nodes and a new primary store is established (manual intervention by sys admin necessary).
- Using the head of the aggregate graph the keys of all currently used aggregates can be retrieved.
- The data of the retrived aggregates is copied to the new primary store
- The aggregate directory is rebuilt.

# Environment

What does Glacier need to work?

## Connectivity

- Network with enough bandwidth between nodes.
- Reliable end-to-end communication (e.g. TCP/IP) between nodes.

## Security

- Encryption of transmission data and stored data (symmetric, e.g. 3DES, AES, IDEA).
- Message authentication between nodes (e.g. public/private key system).

## Overlay network

- Provides mapping of keys to nodes responsible for these keys (Pastry).

# Security (I)

## Integrity

- Remote deletion impossible
  → attacker can only overwrite fragments and replicas on directly controlled nodes.
- Every fragment and replica is encrypted and contains a signed hash (e.g. SHA-1)
  → manipulation would be detected (but replica or fragment is lost).

## Durability

- Attacker must destroy at least $r$ of $N$ fragments and all replicas to destroy data.
  → Difficult to find the hosts responsible for these fragements due to
  - encrypted communication between hosts
  - pseudo random selection of storage nodes
- Attacker must disable the network to stop fragment and replica distribution.

## Space filling attacks

- Insertion of large objects into Glacier
- Insertion of large objects on to the storage nodes

→ Must be prevented by OS and/or applications.

# Security (II)

## Variables

| | |
|---|---|
| $\|O\|$ | Size of object. |
| $N$ | Number of fragments stored per object. |
| $r$ | Number of fragments containing enough data for reconstruction. |
| → | The storage overhead $S$ is determined by the code ($N/r$). |
| | The message overhead is determined by $N$. |
| $f_{max}$ | Failure rate of any node. |

## Durability of an object

Probability that at least $r$ Fragments survive (data can be reconstructed).

$$D = P(s \geqslant r)$$

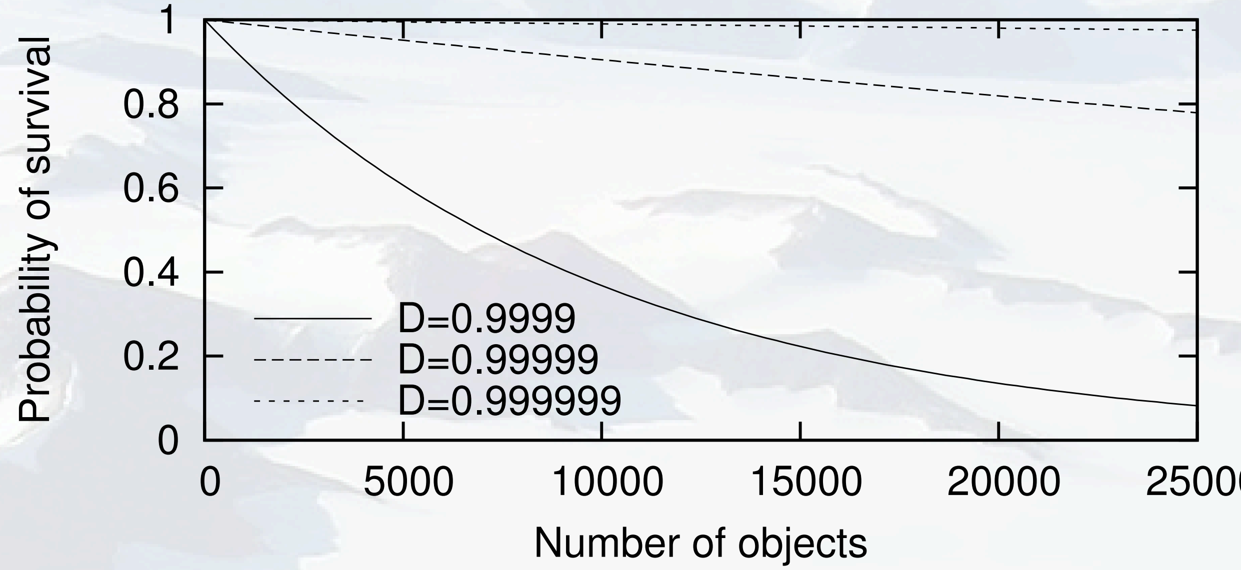
$$= \sum_{k=r}^{N} \binom{N}{k} (1 - f_{max})^k \cdot f_{max}^{N-k}$$

| Failure $f_{max}$ | Durability $D$ | Code $r$ | Fragments $N$ | Storage $S$ |
|---|---|---|---|---|
| 0.30 | 0.9999 | 3 | 13 | 4.33 |
| 0.50 | 0.99999 | 4 | 29 | 7.25 |
| 0.60 | 0.999999 | 5 | 48 | 9.60 |
| 0.70 | 0.999999 | 5 | 68 | 13.60 |
| 0.85 | 0.999999 | 5 | 149 | 29.80 |
| 0.63 | 0.999999 | 1 | 30 | 30.00 |

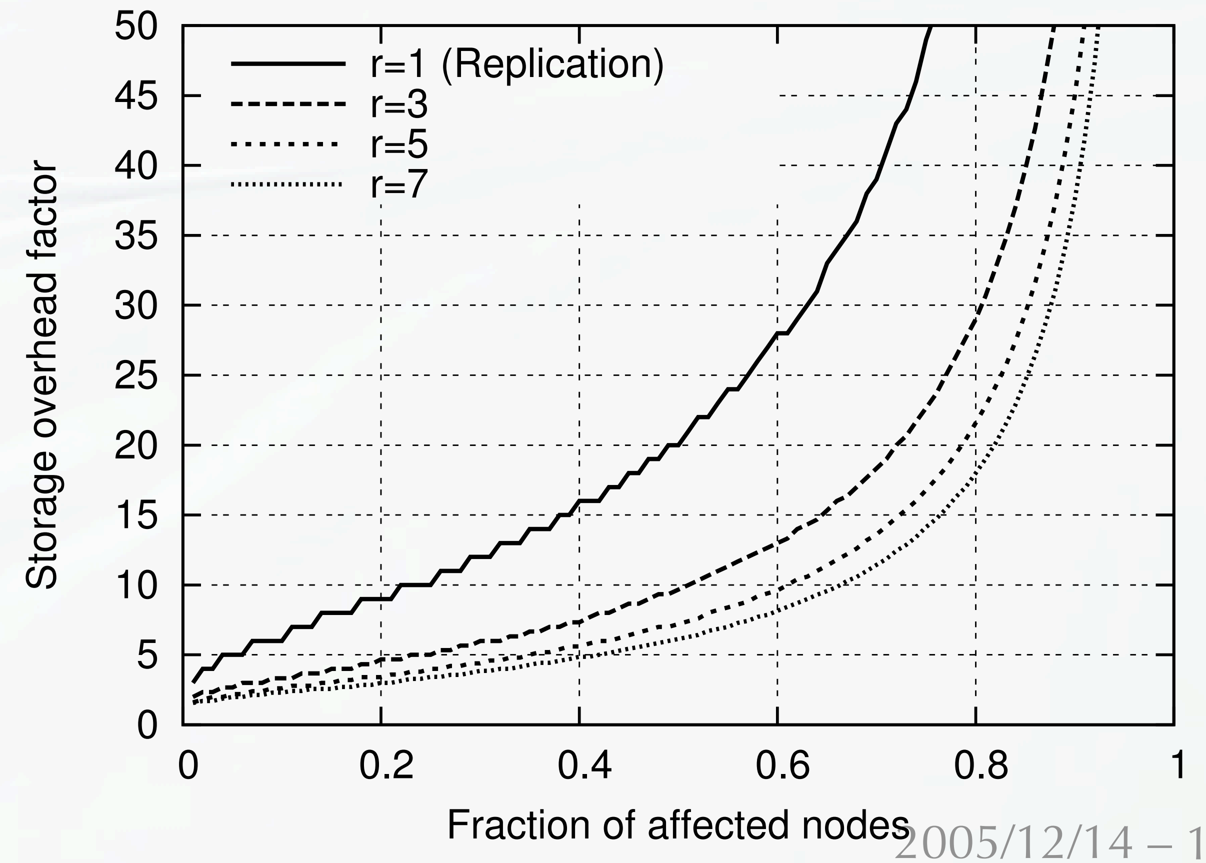
# Security (III)

Durability of an object collection. A collection of $n$ objects survives with a probability of

$$P_D(n) = D^n$$




How much storage overhead do we need for 99.9999% durability in case of a certain failure rate at a given code?



# Overview

- Classical Concept
- Glacier
  - Concept
  - Distribution of data
  - Recovery
  - Environment
  - Security
- ePost experiment
  - Setup
  - Results
- Glacier for a Real File Server
- Competitors?
  - Academic
  - Commercial/Industrial

# ePost: Setup

## ePost

- A cooperative serverless email system.
- 35 nodes.
- 8 active users (use ePost as main mail system).
- 7 passive users (forward all their incoming mail to ePost for storage).

## Glacier setup

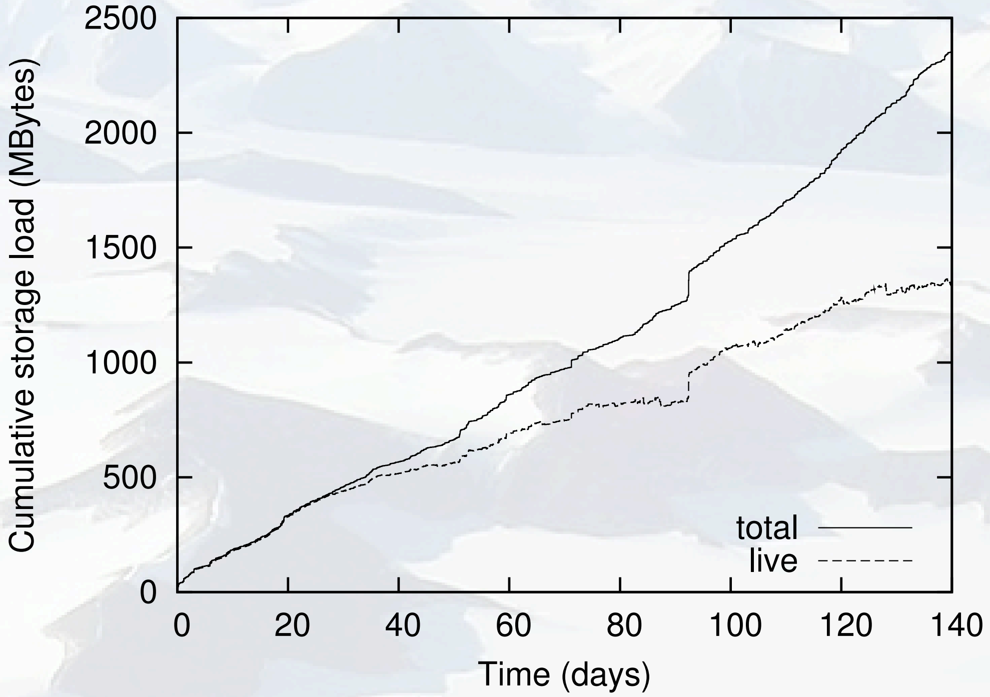
- $N = 48$ fragments per object.
- $r = 5$ (any five fragments are sufficient for restoring).

→ At $f_{max}=0.6$ we get a minimum object durability of $D=0.999999$.

# ePost: Results (I)

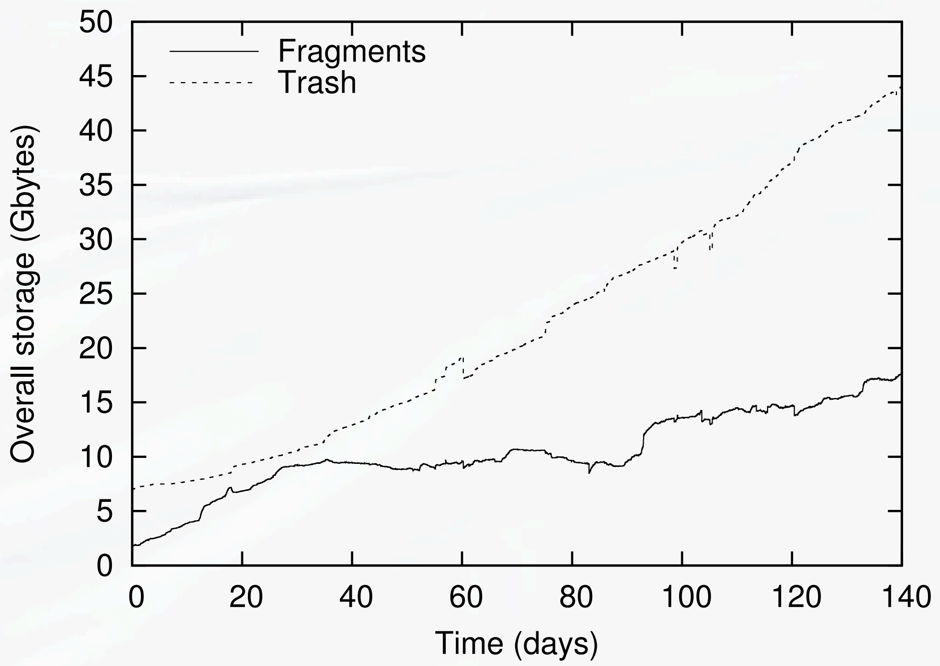
## Effective data

The real amount (without overhead) of ePost data stored using Glacier:



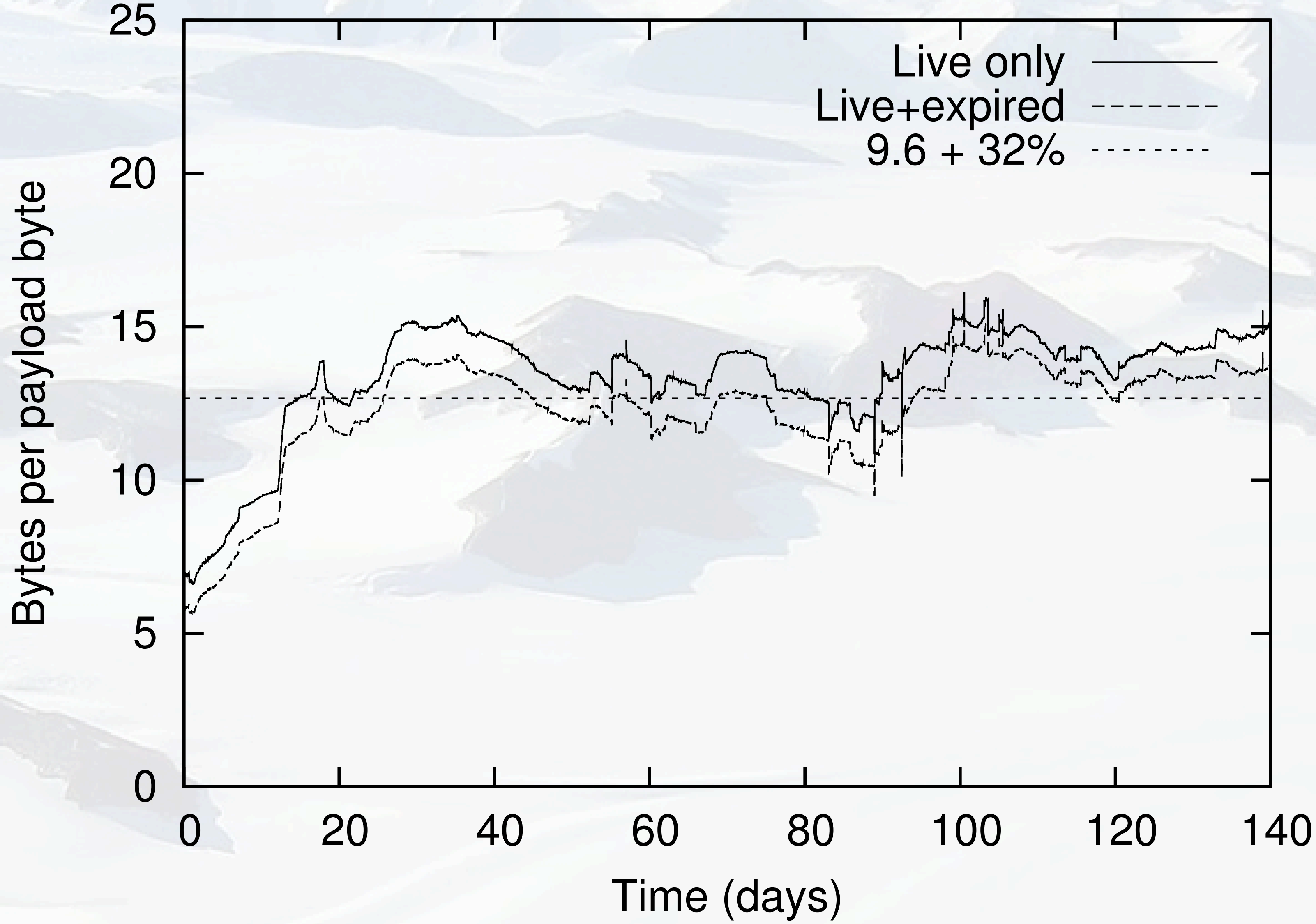

## Used storage

The storage consumed by Glacier for the above displayed real data:



# ePost: Results (II)

Storage factor



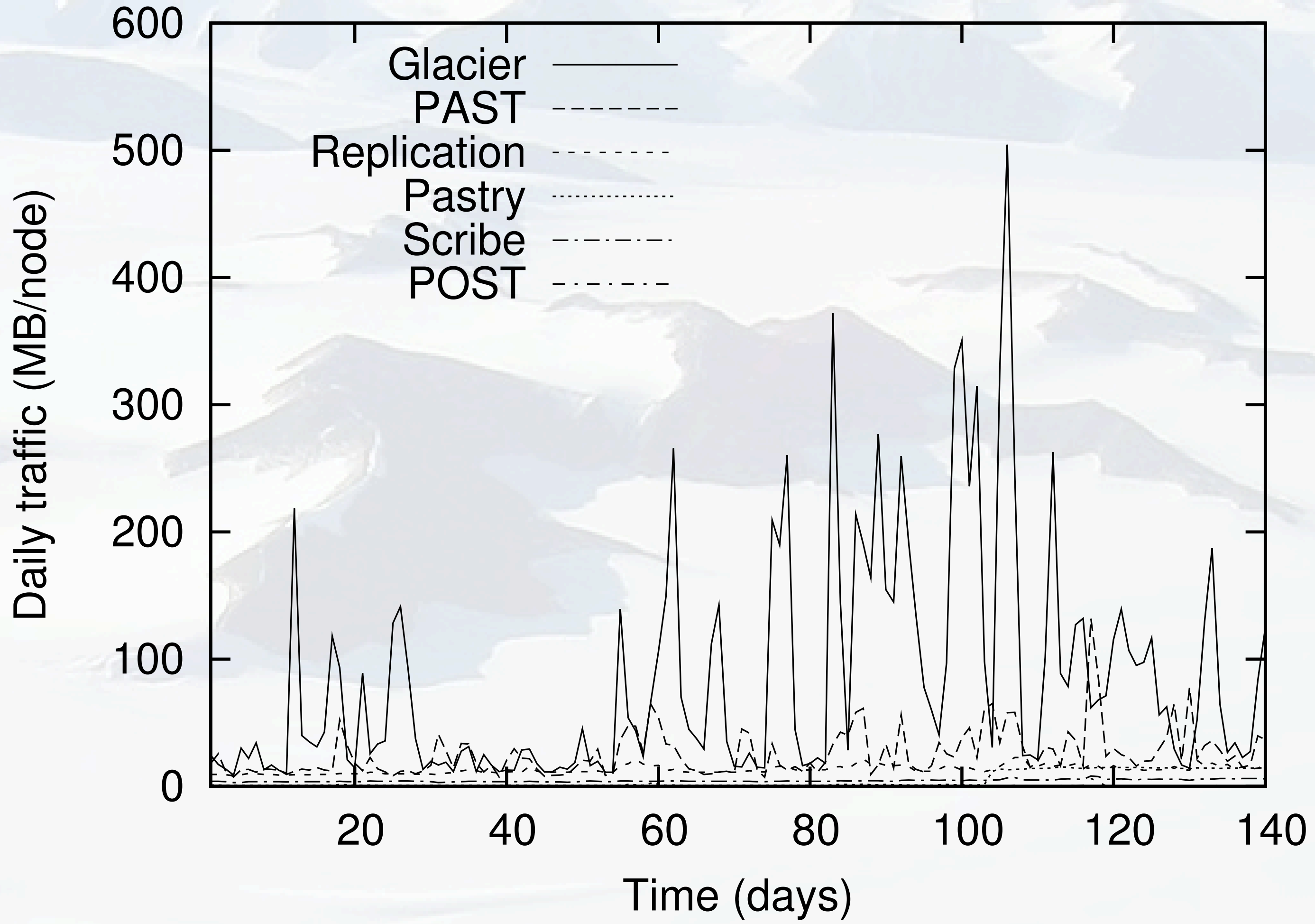

Average overhead factor: 11

# ePost: Results (III)

## Network load

The amount of traffic generated per node and day.
Please note: Between the days 80 and 120 a lot of nodes failed, therefore a lot of data had to be transfered.

This traffic is normally quite well spread over the day because Glacier limits the maximum number of file transfers at the same time:




$$\frac{500 \text{ MB}}{24 \cdot 3600 \text{ s}} = 0.00578 \text{ MB/s} = 0.046296 \text{ Mbit/s}$$

# ePost: Recovery

## «Setup»

- 22 randomly selected nodes were disabled → 13 nodes remaining.

## Result

- After one our the all the data was recovered and copied on a new primary store.

# Glacier: Conclusions/Open Points

- It is possible to use the workstation storage as an additional backup storage.
- Glacier provides a lot of security in case of large scale failure.
- Glacier works for a limited amount of data and objects (some GB).
  But how is it for larger systems with TB of data and a large amount of small objects?
- How does Glacier react on a classical file server situation where a lot of files are constantly changed?

# Overview

- Classical Concept
- Glacier
  - Concept
  - Distribution of data
  - Recovery
  - Environment
  - Security
- ePost experiment
  - Setup
  - Results
- Glacier for a Real File Server
- Competitors?
  - Academic
  - Commercial/Industrial

# Glacier as Backup for a Real File Server (I)

## Assumptions

- 1 TB normal user data has to be backuped.
- Average workstation: 100 GB free disk space.

## Glacier setup (the same as for ePost)

- $N = 48$ fragments per object.
- $r = 5$ (any five fragments are sufficient for restoring).

## Results extrapolated from the ePost results

- Storage overhead factor 11 → 11 TB of workstation disk capacity is need

→ at least 110 workstations necessary!

- ePost average network load: 0.0463 MBit/s for about 1.5 GB user data

→ average network load for 1 TB user data: 30.9 MBit/s!

# Glacier as Backup for a Real File Server (II)

## Mobile office problem

- More and more companies are only using laptops → less diskspace, less often connected to the company network.

## Backup verification problem

- Splitting
- Encryption

→ Verification and extraction of backup data is very difficult. Would you implement it, if you were responsible?

# Overview

- Classical Concept
- Glacier
  - Concept
  - Distribution of data
  - Recovery
  - Environment
  - Security
- ePost experiment
  - Setup
  - Results
- Glacier for a Real File Server
- Competitors?
  - Academic
  - Commercial/Industrial

# Academic Competitors

## DISP: Practical, Efficient, Secure and Fault Tolerant Data Storage for Distributed Systems

- Daniel Ellard (Harvard University), James Megquier (Gnuterra Corporation), 2003
- Nearly the same concept as Glacier except for the sophisticated fragment placement concept and primary store.
- No real world test results available, but basic tests:
  - Pentium 1.8 GHz, Gigabit Ethernet, FreeBSD 4.8
  - 100 MB data encrypted on disk

  → decryption, SSL encryption for transfer, verification (SHA-1) at receiver

  → Result: about 8 MByte/s

# «Real» Systems

## Is there anything?

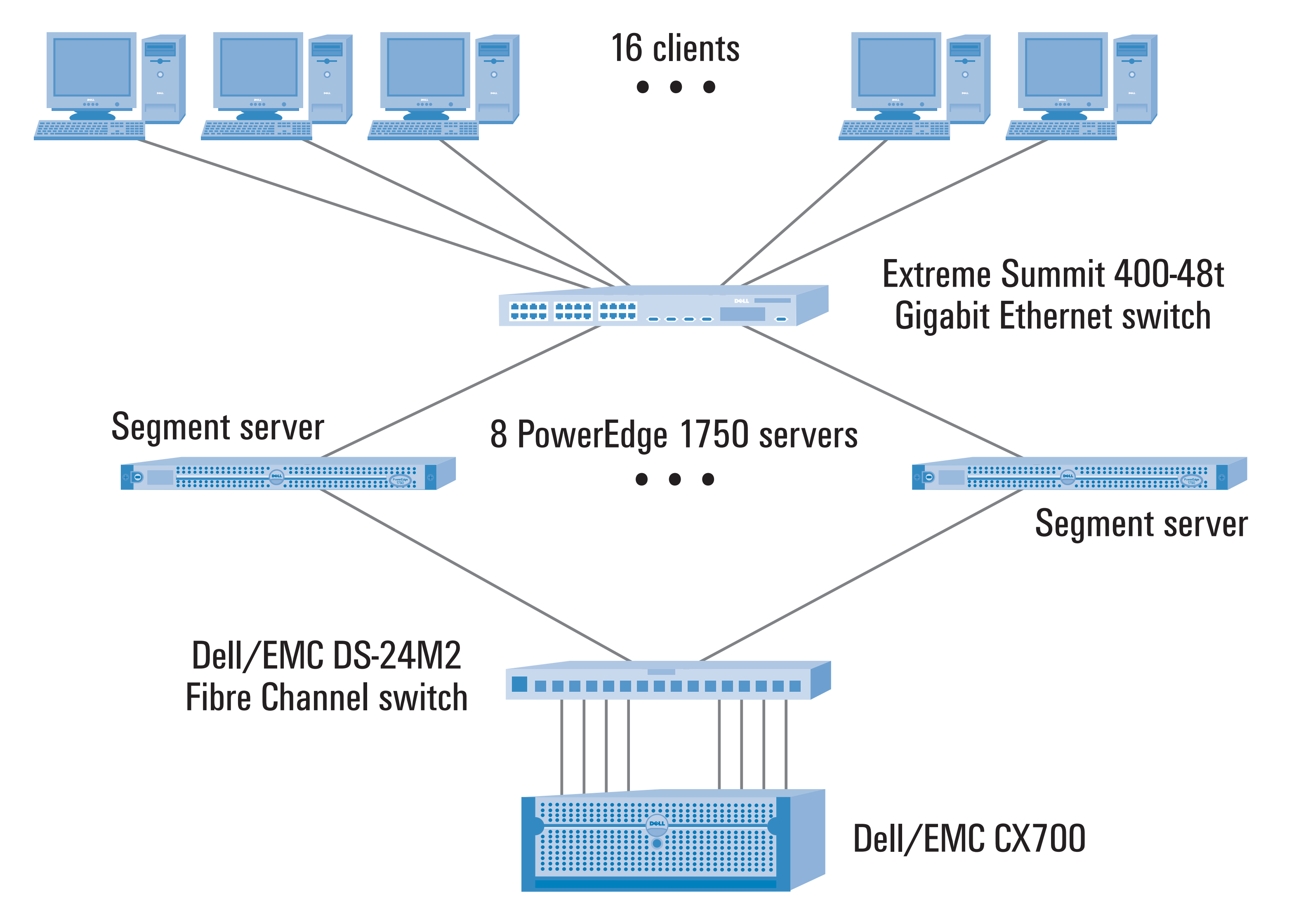
- No system found which uses working clients for backup.

## But there are similar systems:

- Use of cheap PCs for storage instead of expensive SAN-Systems.
- Expensive storage systems are connected to several storage servers which in turn realize the connection to the workstations.
- Main differences to the academic systems presented:
  - Storage PCs/Servers are only used for storage.
  - Dedicated LAN for storage purposes.

# Expensive System (I)

## Dell high performance storage with IBRIX file system



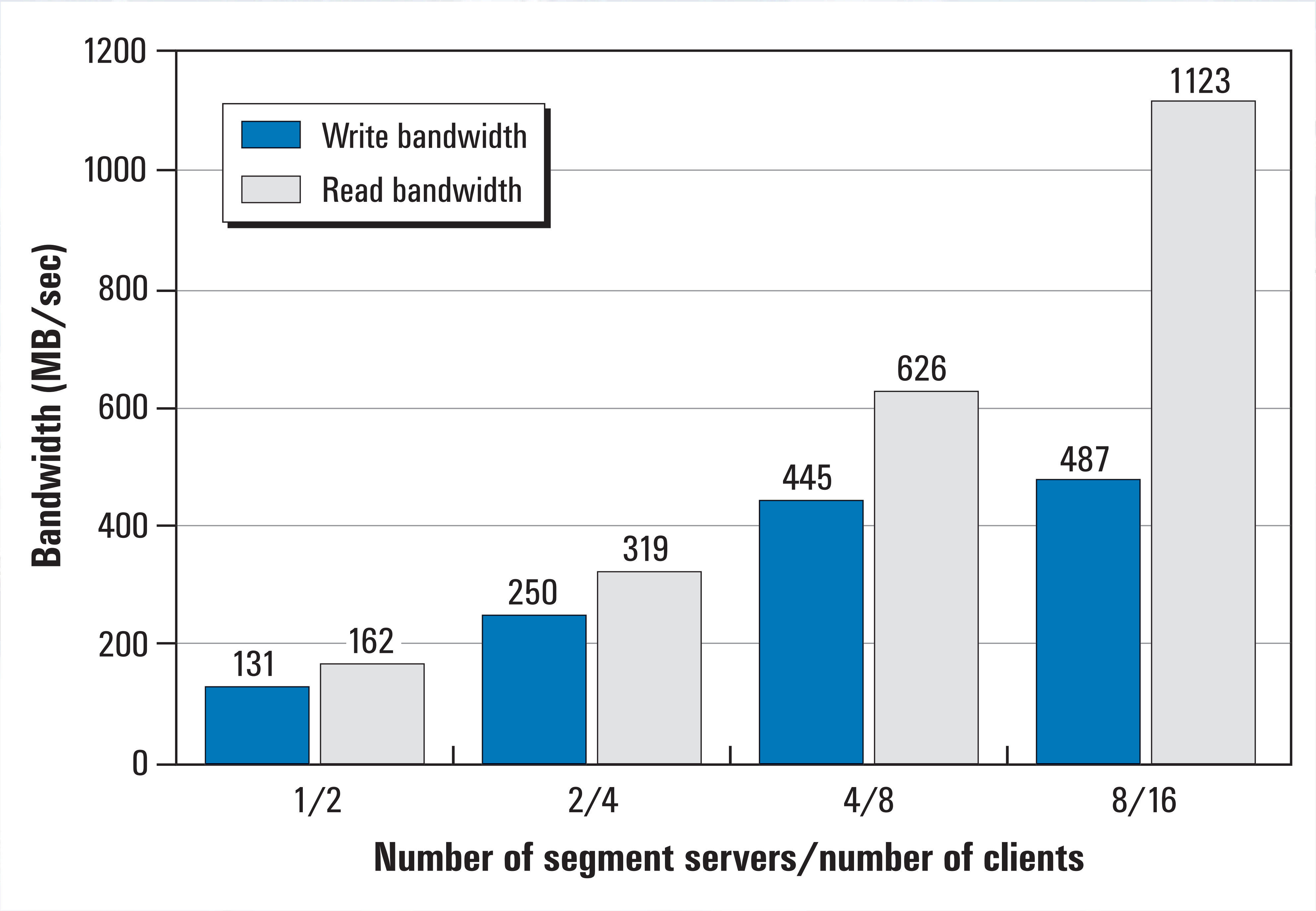

- Clients:
  2 x 3 GHz, 2 GB Ram
- Segment Servers:
  2 x 3 GHz, 2 GB Ram
  Fibre channel card

## Concept

- Segment servers store the metadata (i.e. where the files are stored).
- IBRIX allows that data storage is completly independent of namespace and file hierarchy → Excellent optimization for application is possible.

# Expensive System (II)

The amount of data transfered between clients and storage:



# Expensive System (III)

## What about safety?

- Up to 240 disks in storage units combined to RAID-1, RAID-3, RAID-5 or RAID-10 disk groups.
- Certified server disks
- Verification of written data independently by two processors.
- Large caches with batteries to write cache to dedicated RAID-5 protected disks in case of complete power failure.
- «Normal» features like hot swappable power supplies, disks, storage processors etc.

# Lustre (I)

## A Scalable, High-Performance File System

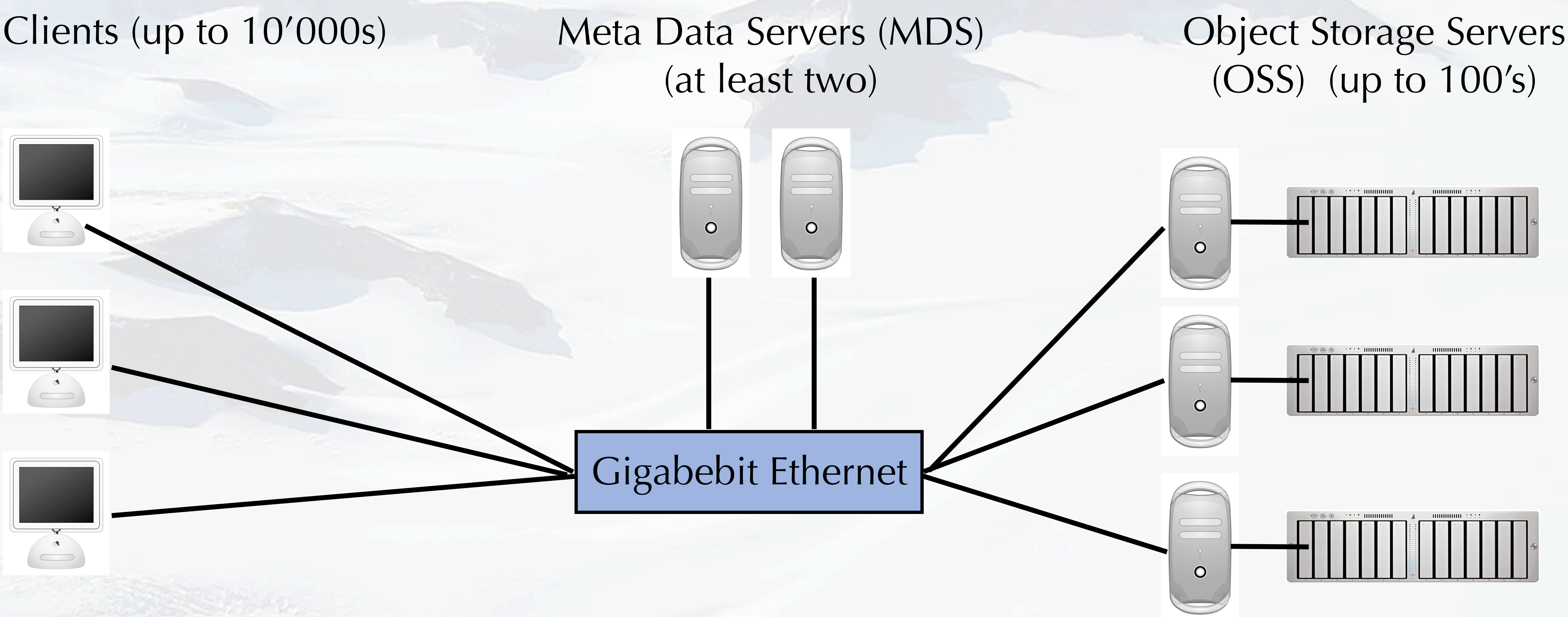
- Developed and maintained by Cluster File Systems, Inc
- Open source  (GPL)
- OS: Several Linux distributions
- Hardware platforms: IA-32, IA-64, X86-64, PPC
- Networking: TCP/IP, InfiBand and others

# Lustre (II)

## Concept

Seperation of

- Workstations accessing data
- Meta Data Servers (MDS) storing filesystem metadata
- Object Storage Servers (OSS) storing actual data



# Lustre (II)

## Performance

Example: Tungsten Supercomputer at the National Center for Supercomputing Applications (NCSA) at the University of Illinois:

- 104 Object Storage Servers
- 120 TB storage
- Over 11.1 Gigabyte/s I/O throughput using Lustre

## Security

- Automatic failover for Meta Data Servers
- Replication of data accross several Object Storage Servers including automatic failover for read and write access
- Possibility to integrate Kerberos authentication and encrypted data transfer

# Lustre (III)

A possible realization
(not as fast as at NCSA):

PetaBox TB80

- 40 nodes, each with 2 TB storage,
  1 GHz CPU, Gigabit Ethernet
- Gigabit Ethernet Switch: 48 Ports
- Only 3.2 KW power needed (ecological computing)
- about USD 2000 per Terabyte, all inclusive
  (disks, CPU, board, networking, rack, cooling).
  → USD 160’000 for 80 TB