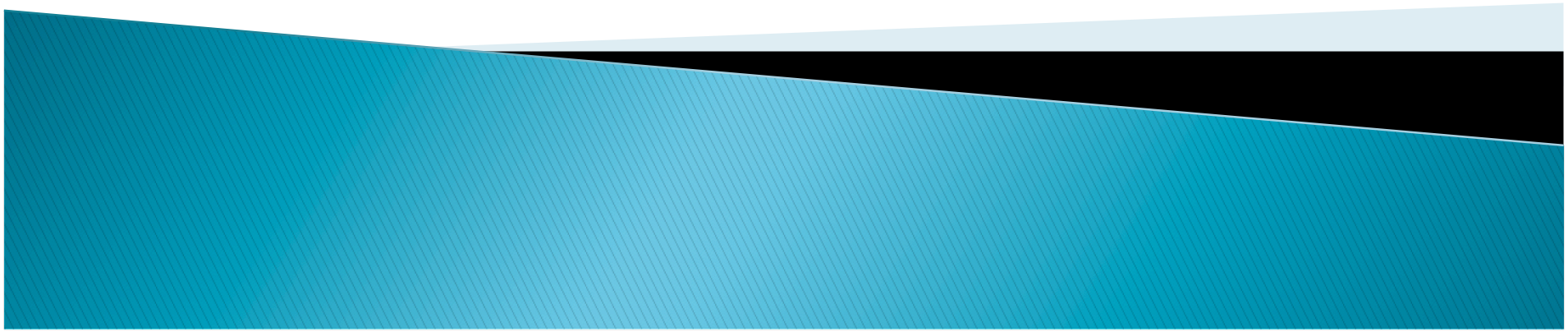
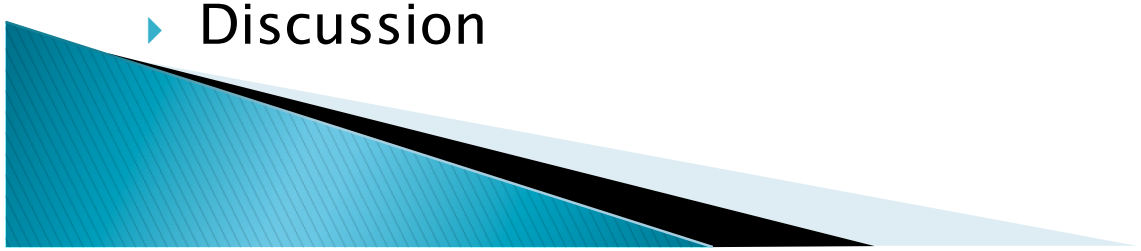
Frangipani: A scalable distributed File System

Presented by:
Alvaro Llanos E



Outline

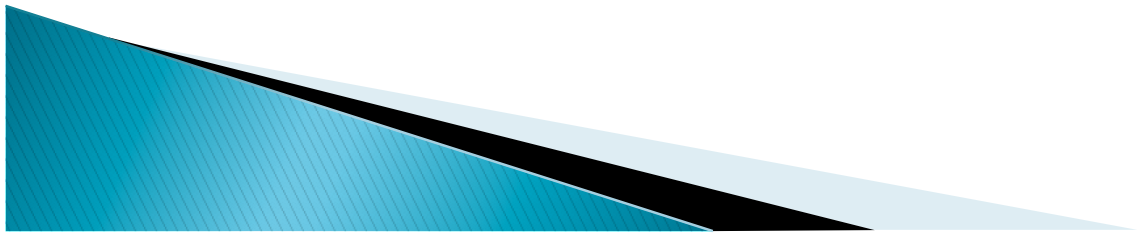
- Motivation and Overview
- Frangipani Architecture overview
- Similar DFS
- PETAL: Distributed virtual disks
  - Overview
  - Design
  - Virtual → Physical mapping
  - Failure tolerance
- Frangipani components
  - Security
  - Disk Layout
  - Logging and Recovery:
  - Cache
  - Lock service
- Performance
- Discussion



FRANGIPANI: Motivation

- Why a distributed file system?
  - Scalability, Performance, Reliability, Durability

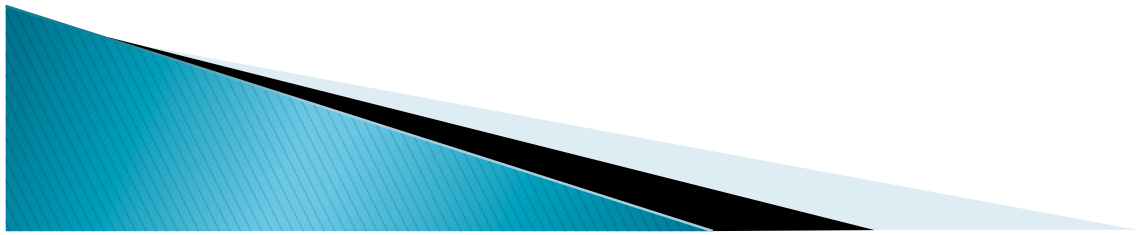
- Frangipani
  - Scalability → Easy to add more components
  - Administration → Simple – All users view the same set of files.
  - Tolerates and recover from machine, network, and disk failures.



FRANGIPANI: Overview

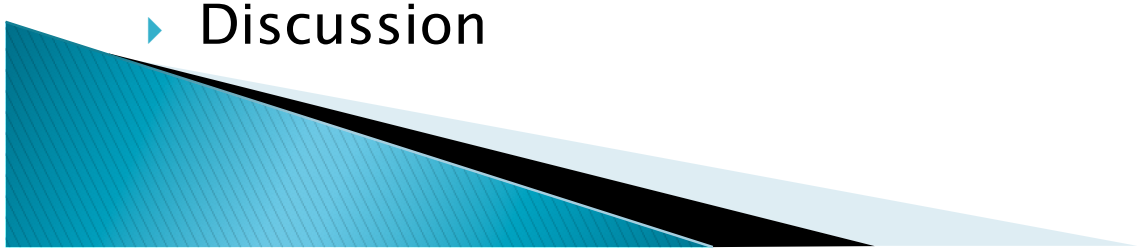
- Focused on Clustered systems
- Simple design
  - Many assumptions and constraints → Performance impact?
    - Trusted environment: Limited security considerations
    - Lack of portability: Runs in kernel level
    - Two layered design: PETAL is used

      ⇓ control : ⇑ simplicity
- **Tight design?**:
  - Support for alternative storage abstractions?

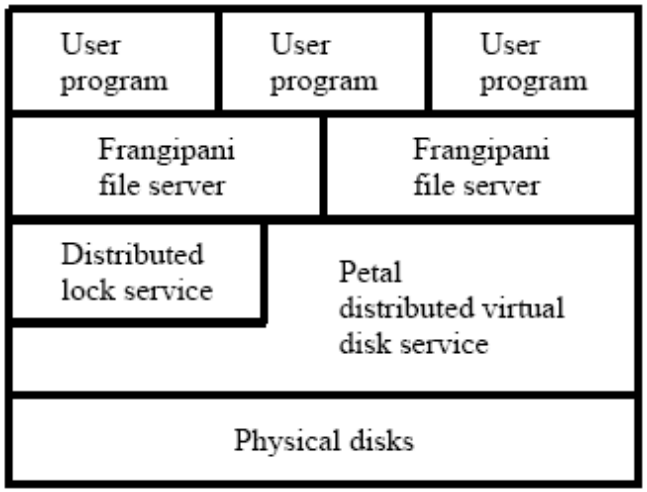


Outline

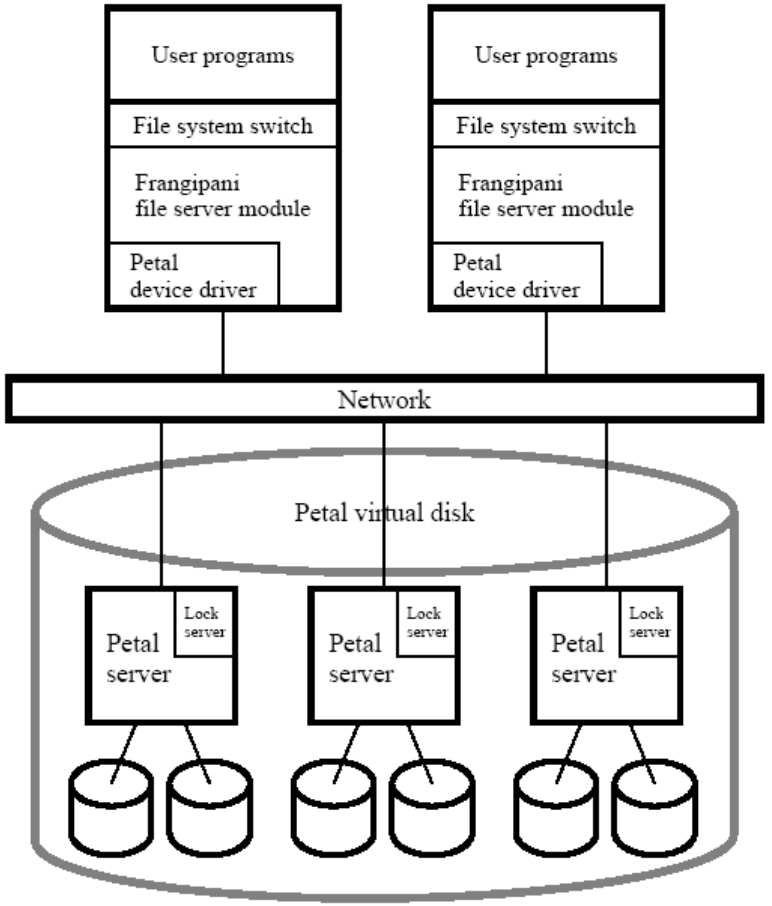
- Motivation
- Frangipani Architecture overview
- Similar DFS
- PETAL: Distributed virtual disks
  - Overview
  - Design
  - Virtual → Physical mapping
  - Failure tolerance
- Frangipani components
  - Security
  - Disk Layout
  - Logging and Recovery:
  - Cache
  - Lock service
- Performance
- Discussion



FRANGIPANI: Architecture overview

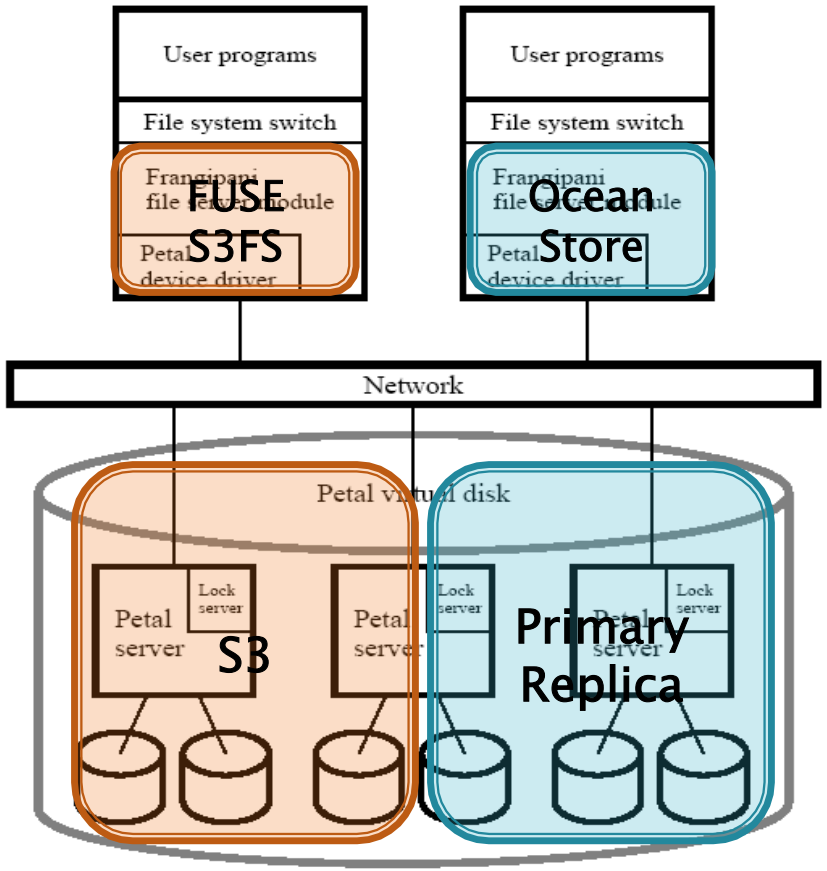


Frangipani layering



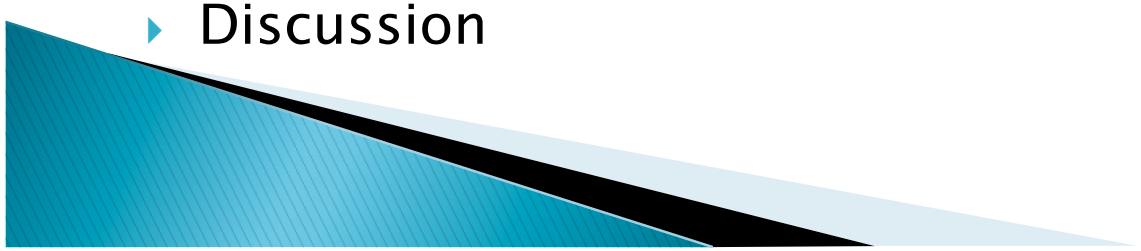
One possible configuration

Frangipani: Similar Designs



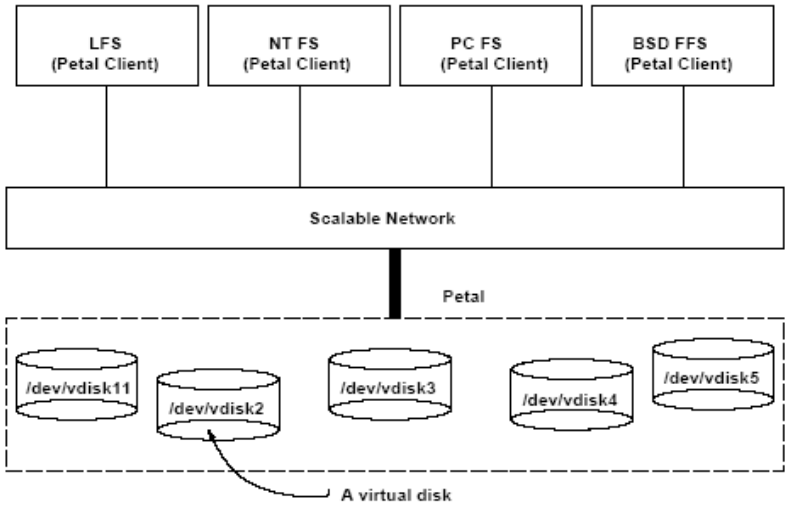
Outline

- Motivation
- Frangipani Architecture overview
- Similar DFS
- PETAL: Distributed virtual disks
  - Overview
  - Design
  - Virtual → Physical mapping
  - Failure tolerance
- Frangipani components
  - Security
  - Disk Layout
  - Logging and Recovery:
  - Cache
  - Lock service
- Performance
- Discussion

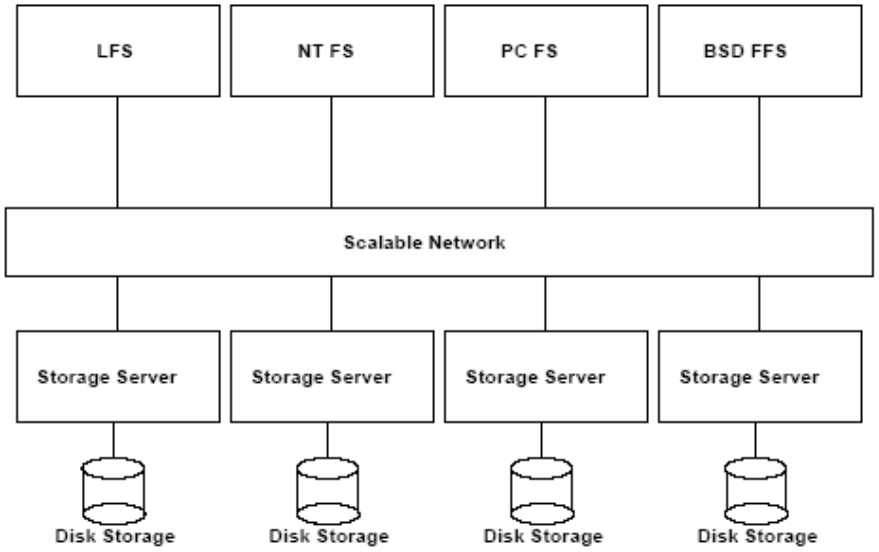


PETAL: Distributed Virtual Disks

Client view

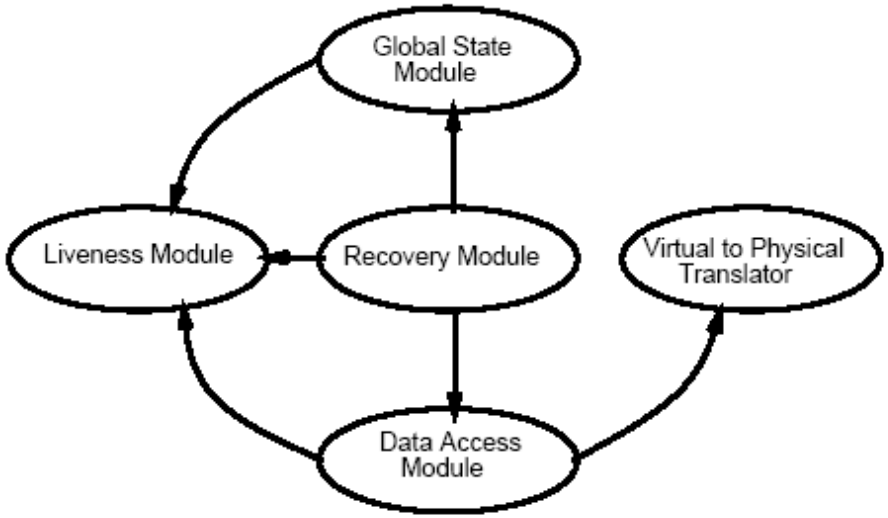


Physical view



In which context this might not be the best approach?

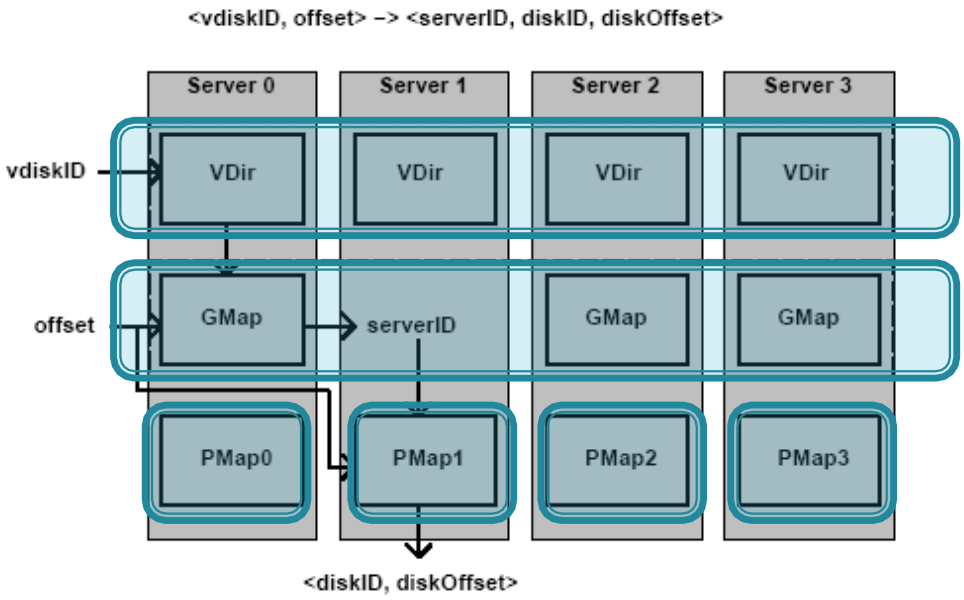
PETAL: Design



- Servers maintain most of the state.
- Clients maintain only ‘hints’
- Algorithm guarantees recovery from random failures of servers and network connectivity as long as majority of servers are up and communicated.

PETA: Virtual → Physical

- Virtual ID -> Global ID
- Global Map identifies correct server.
- Physical Map in the ser-ver translates the GID into the Physical disk and real offset.



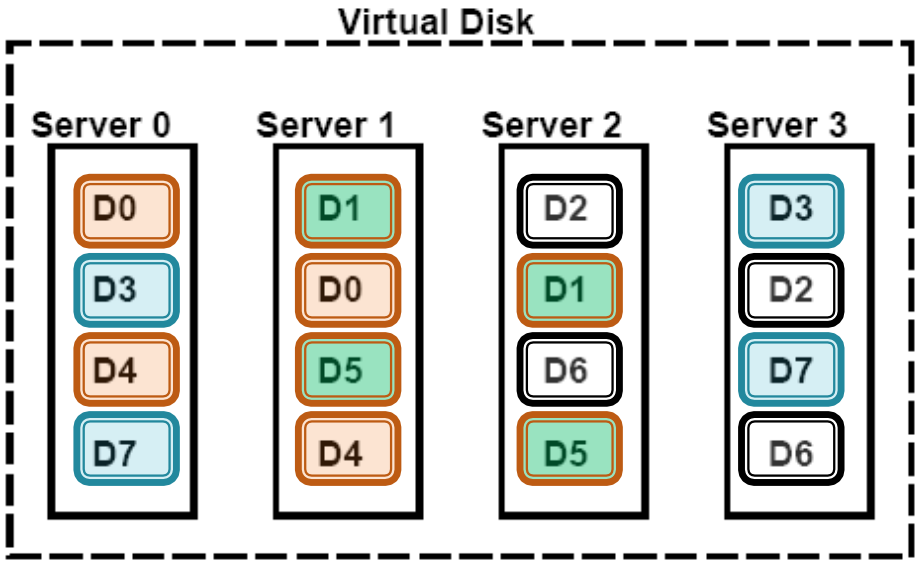
PETA: Failure tolerance

- Chained-declustered data access.

- Odds and even servers separated
 → tolerate site failures

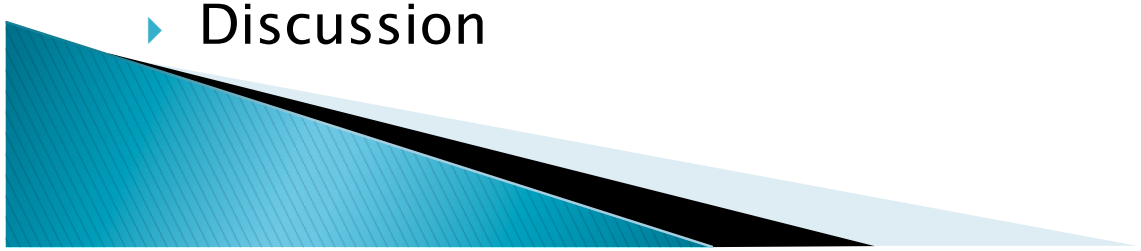
-Recoveries from server failure →
 Both contiguous servers have info.

- Dynamic load balancing.



Outline

- Motivation
- Frangipani Architecture overview
- Similar DFS
- PETAL: Distributed virtual disks
  - Overview
  - Design
  - Virtual → Physical mapping
  - Failure tolerance
- **Frangipani components**
  - **Security**
  - **Disk Layout**
  - **Logging and Recovery:**
  - **Cache**
  - **Lock service**
- Performance
- Discussion



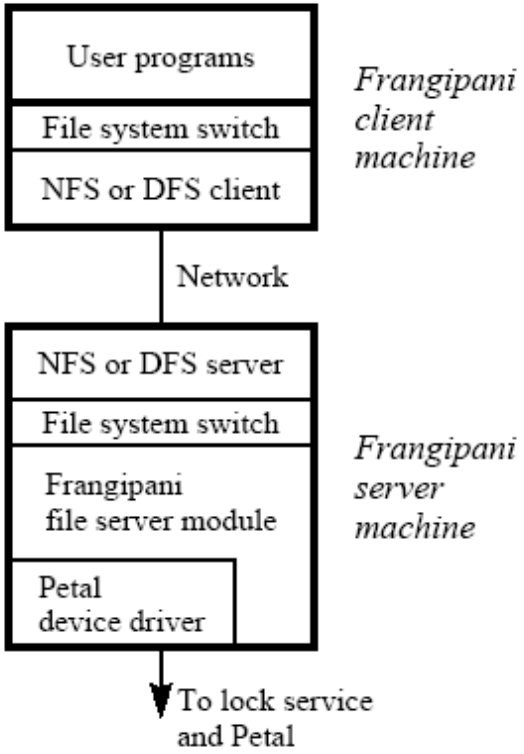
FRANGIPANI: Security

Security

Lock and Petal servers need to run on trusted servers.

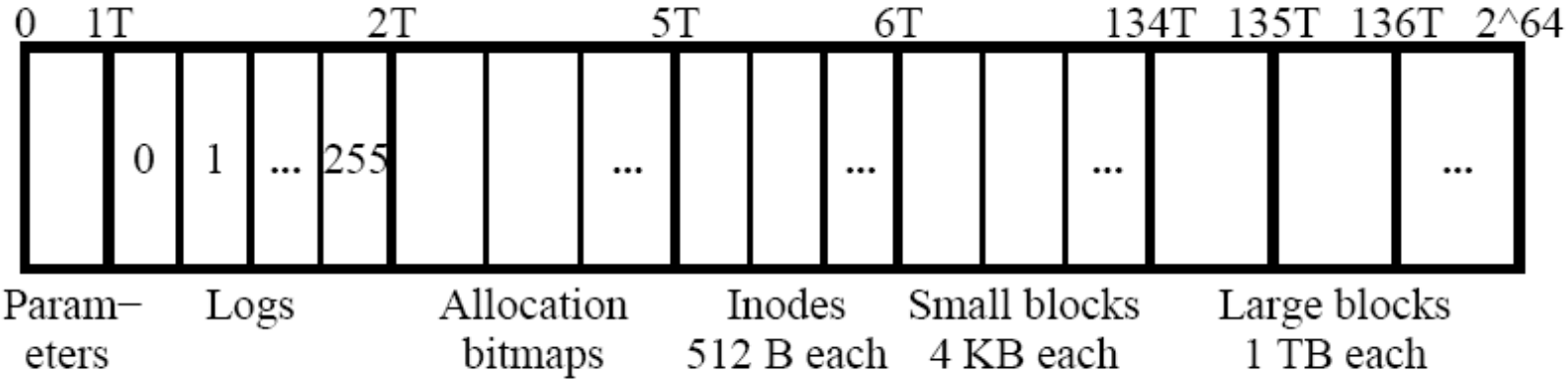
Frangipani client file systems can be in untrusted servers.

Client/Server Configuration



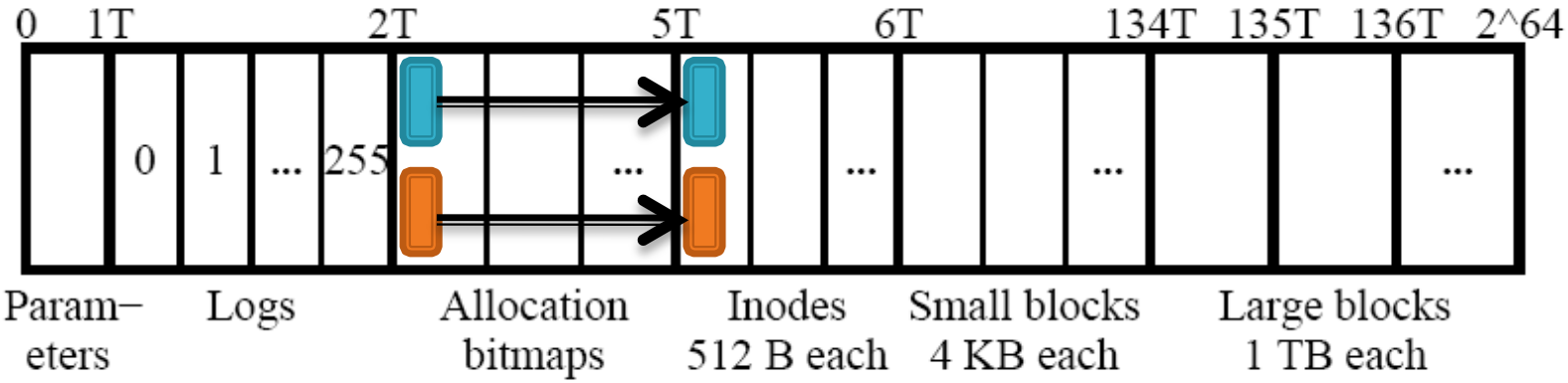
FRANGIPANI: Disk Layout

- Petal's sparse disk address space 2 64
- Each server has own log and own blocks of



- Regions:
  - First → Shared configurations parameters
  - Second → Logs
  - Third → allocation bitmaps
  - Fourth → inodes
  - Fifth → Small data blocks
  - Rest → Large data blocks

FRANGIPANI: Disk Layout



2^{64} Byte address space limit chosen based on usage experience.

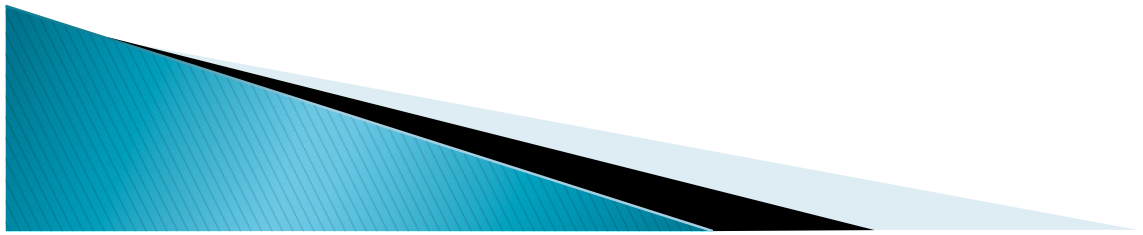
Separating inodes and data blocks completely might result in performance hit.

FRANGIPANI: Logging and Recovery

- Each Frangipani server has its own log in PETAL.
- Logs bounded in size → stored in a circular buffer → when full, system reclaims oldest 25%.
- Changes applied only if record version > block version
- Metadata blocks are reused only by new metadata blocks: Data block do not have space reserved for version numbers
- Logs store only metadata → Speeds up

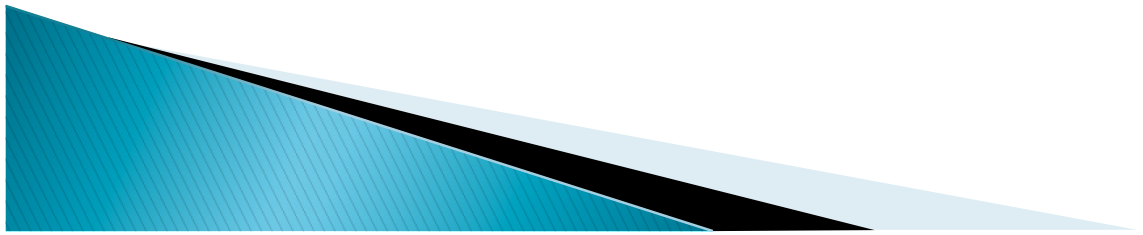
FRANGIPANI: Cache

- Dirty data is flushed to disk when downgrading locks: Write → Read
- Cache Data is invalidated if it is releasing the lock: Read → No Lock → Someone requested a Write Lock.
  - Dirty data is not sent to the new lock owner.
  - Frangipani servers communicate only with Petal.
- One lock protects inode and data blocks
  - Per-file granularity.



Frangipani: Lock Service

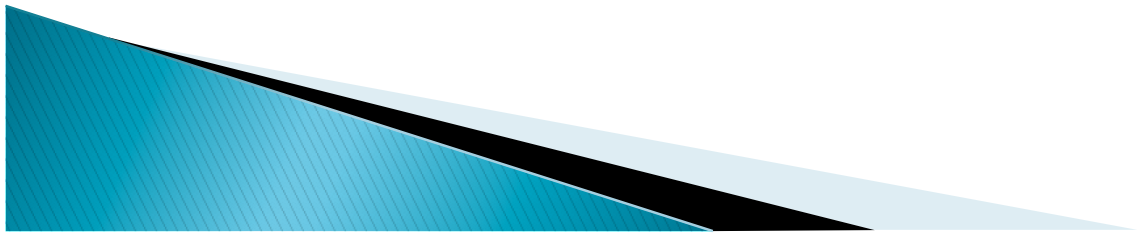
- Client failure → Leases
  - If client lease expires and there are dirty blocks → user programs get ERROR message for subsequent requests
  - Need to unmount File system to clear this error.
- 3 implementations
  - Single Centralized server
    - Failure → Performance impact
  - Lock state in PETAL:
    - it is possible to recover Locks state if server fails.
    - Poorer performance
  - FINAL Implementation
    - Cooperating Lock servers and Clerk module
    - Asynchronous calls



Frangipani: Lock Service

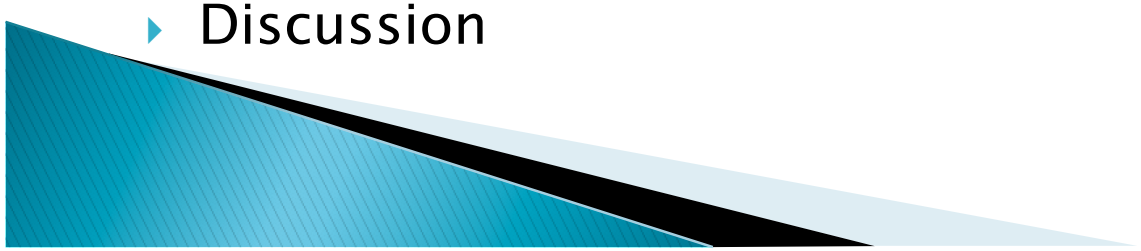
- Lock Server → Multiple Read/ Single Write locks
- 2 servers might try to write same file, both will keep acquiring and releasing locks

- Asynchronous messages: request, grant, revoke, release. (Optional Synchronous)
- Crash of Lock Servers
  - Same as Petal → heartbeats between servers → majority consensus to tolerate network partitions.
  - If lock server crashes, locks managed by it are redistributed. Lock state is retrieved from the clerks
  - If frangipani server fails, the locks are assigned to another Frangipani server and a recovery process is executed.



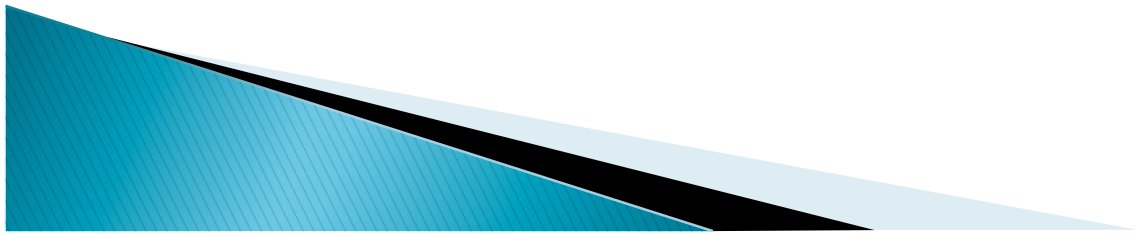
Outline

- Motivation
- Frangipani Architecture overview
- Similar DFS
- PETAL: Distributed virtual disks
  - Overview
  - Design
  - Virtual → Physical mapping
  - Failure tolerance
- Frangipani components
  - Security
  - Disk Layout
  - Logging and Recovery:
  - Cache
  - Lock service
- **Performance**
- Discussion



Frangipani: Performance

- Configuration
  - Single machine
  - Seven PETAL servers → Each one store on 9 Disks
  - Connected by ATM switch



Frangipani:Performance

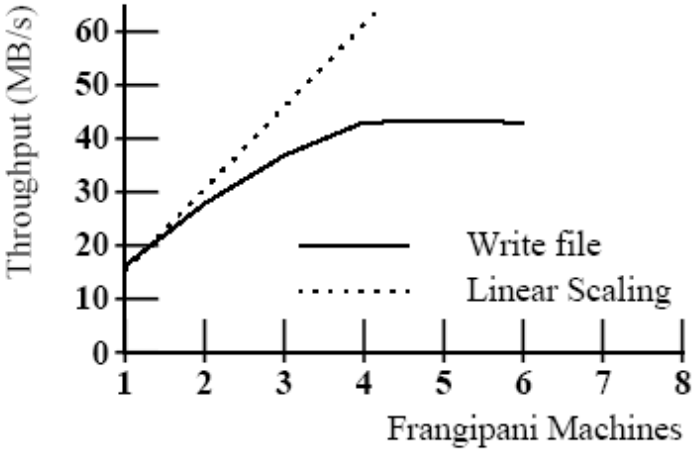
Test	Description	AdvFS		Frangipani	
		Raw	NVR	Raw	NVR
1	file and directory creation: creates 155 files and 62 directories.	0.92	0.80	3.11	2.37
2	file and directory removal: removes 155 files and 62 62 directories.	0.62	0.62	0.43	0.43
3	lookup across mount point: 500 getwd and stat calls.	0.56	0.56	0.43	0.40
4	setattr, getattr, and lookup: 1000 chmods and stats on 10 files.	0.42	0.40	1.33	0.68
5a	write: writes a 1048576 byte file 10 times.	2.20	2.16	2.59	1.63
5b	read: reads a 1048576 byte file 10 times.	0.54	0.45	1.81	1.83
6	readdir: reads 20500 directory entries, 200 files.	0.58	0.58	2.63	2.34
7	link and rename: 200 renames and links on 10 files.	0.47	0.44	0.60	0.50
8	symlink and readlink: 400 symlinks and readlinks on 10 files.	0.93	0.82	0.52	0.50
9	statfs: 1500 statfs calls.	0.53	0.49	0.23	0.22

Connectathon Benchmark with unmount operations.

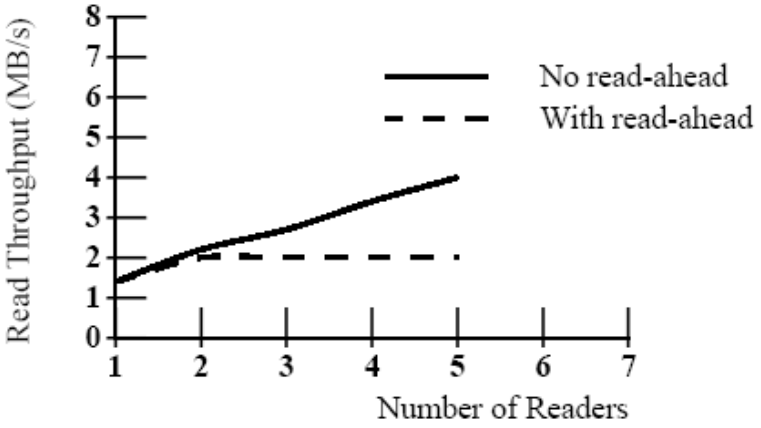
	Throughput (MB/s)		CPU Utilization	
	Frangipani	AdvFS	Frangipani	AdvFS
Write	15.3	13.3	42%	80%
Read	10.3	13.2	25%	50%

Frangipani Throughput and CPU Utilization.

Frangipani: Performance



Frangipani Scaling on Write.



Frangipani Reader/Writer Contention.

Discussion

- Layered design → Performance impact?
  - Although the design gains in simplicity, we loose control over the way to store the data in a 2 layered system.
  - Exists a performance impact but it might be worth it according to the requirements.
- One big virtual disk → Best approach?
  - Depends on the context, this case was a clustered environment.
- Inodes separated from data blocks → impact?
  - Sparse information, impact in the Reads
- Simplicity vs Performance
  - Very dependable on the needs.