Petral / Frangipani

interface / abstraction:

foreach VIRTUAL DISK: blocks (0...N-1)
read/write (offset) (delete)

[All servers cooperating to provide abstraction]
Liveness: heartbeats + majority consensus

Global state:
  [membership]
  [global mapping into]
  \( \rightarrow \) paxos

\( \rightarrow \) common pattern
  \{ inner ring: paxos \}
  \{ outside: not \}

\( V \rightarrow P \) translation

e.g. \( \langle v \text{ disk, offset} \rangle \)
\( \updownarrow \text{ key} \)
\( \langle \text{server, disk, local offset} \rangle \)

Data Access
\( \Rightarrow \) mirroring, striping

Recovery
$V \rightarrow P$ (virtual → physical)

3 levels of mapping:

- **V Dir**: global, replicated
- **G Map**: not much data (slowly changes)
- **P Map**

Local/Server: lot of translation info (quickly changes)

Section 2.1:

- **V Dir**: $V$ disk ID $\rightarrow$ (Global Map ID, epoch #)
- **G Map**: GMap ID $\rightarrow$ server(s)
  
  servers, redundancy scheme (e.g., chained declustering)

- **P Map** (per server): offset ($V$ disk) $\rightarrow$ local disk, local offset
  
  $= \rightarrow$ (sparse) page table
  
  (64KB)
Backup: snapshots

(GMap ID, epoch) immutable

implies are not "consistent" (from app perspective)

Incremental Reconfiguration:

1) (local

Server node goal: add disk

Actions:
1) writes -> easy
2) reads -> hard offline/background rebalance

All local: "easy" (no Paxos) consensus
2) server addition

GMap \((S_1, S_2, \text{striping})\)

```
\begin{array}{ccc}
S_1 & S_2 & S_3 \\
\downarrow & \downarrow & \uparrow \\
0 & 1 & 2 \\
23 & 34 & 5 \\
\times 6 & \times 7 & 8 \\
\ldots & \ldots & \ldots \\
\end{array}
```

Process: reconfig

- create new gmap
- change voir \(\rightarrow\) gmap
- move all data
- continue normal operation

"it depends"
"Basic" algorithm:

\[ gmapnew \leftarrow gmap-old \]

writes: new

reads: use new map but, it fails, consult old map

(by: moving data \( \text{old map} \rightarrow \text{new} \))

@ some rate

Problem? too slow

Solution: relocate parts of map @ time

\[ \text{v Disk:} \]

regions: old, new, fenced

use use in flight

\( \text{old new} \) (new, old?)

use "basic"

Perf problem? Hot spots
Locality in W/L (workload)

Solve? = guessing that

Data Access / Recovery

1) Redundancy scheme
   => Striping
   => Chained declustering '89

Mirror: failure

0 0 1 1
2 2 3 3
4 1 5 5
6 6 7 7

load problem balance
Load: 0 1 2 3 4 5 6

better:

dynamic load balancing:
clients track outstanding # of reqs / disk

Petal: replicas

approach: logging: busy
Primary / Backup
writes

(=) writes: slower
logging

next optimization:
(=) lazy in
marking "not busy"
Petal: virtual disk

target for FS's, DBMS's

Limits: scale reconfig = expensive