Lecture 2a: RAID

Outline

1. Motivation
2. Background
3. Proposed RAID
4. Proposed Rebuild Algorithm and Analysis
5. Mapping Strategy
6. Result
7. Log-Structured File System
8. Summary

The I/O Bottleneck

- processor speeds approx. double each year.
- memory chip capacities quadruple each 2-3 years
- disk technology moves more slowly - disk storage capacity doubles each 3 years.
- <u>BUT</u> seek times halve approx. every 10 years.

I/O architecture is becoming a major issue

- Database applications
- Scientific applications (supercomputers)
- General applications

Metrics

- Storage capacity
- Bandwidth
- Access rate
- Mean time to failure
  (data unavailability)
- Performance with failed disks

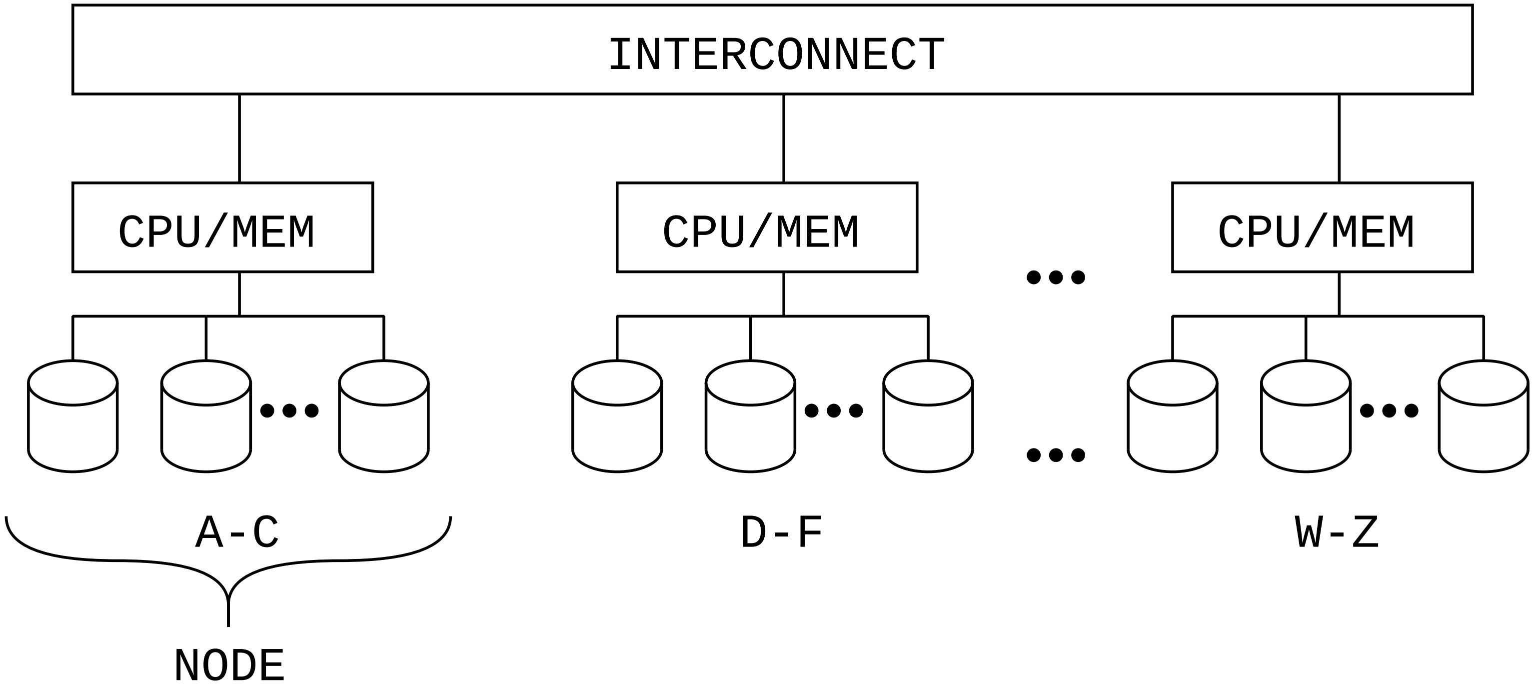
Relieving the I/O Bottleneck

Approaches

1. hardware, e.g. disk arrays, NVRAM
2. data placement, organization, e.g. log-structured file systems
3. query processing algorithms, e.g. “stream processing”

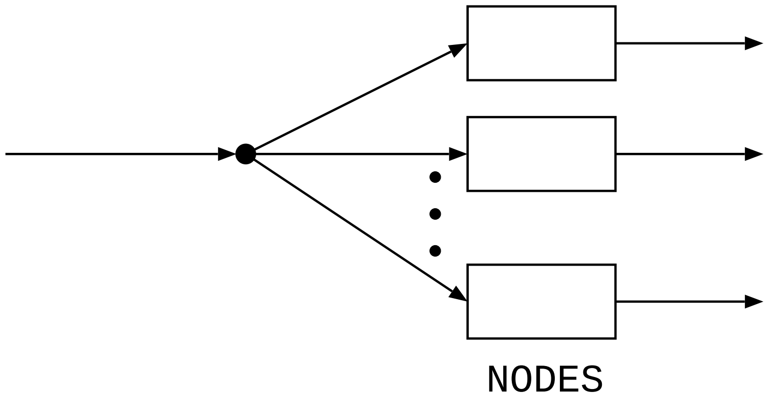
Parallel Execution of Database Queries

Declustering of Data for Parallelism

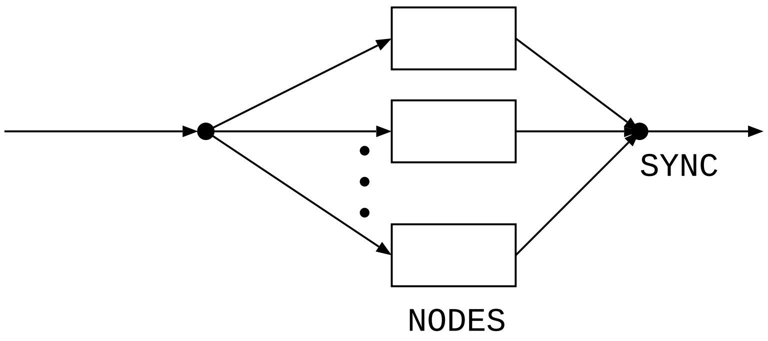


Two Extremes in Workload

- online transaction processing

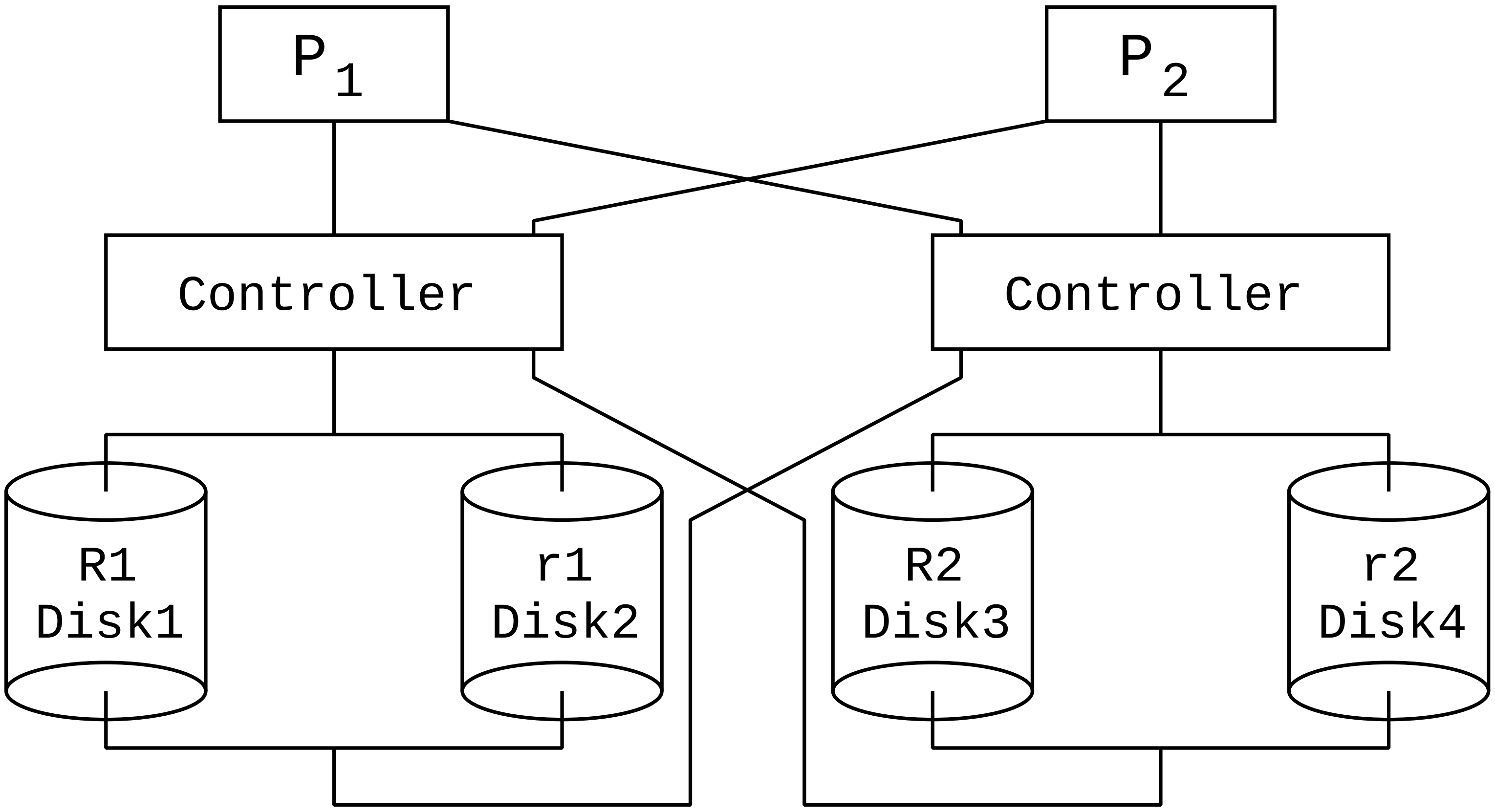


- complex queries



Full Replication Schemes

1. Mirrored Disks (Tandem)



Mirrored Disks

$$\left.\begin{array}{lcl} \lambda_r + \lambda_w & \longrightarrow & \bigcirc \rightarrow \\ \lambda_r + \lambda_w & \longrightarrow & \bigcirc \rightarrow \end{array}\right\} \text{mirrored pair}$$

With one failed:

$$\begin{array}{lcl} 2\lambda_r + \lambda_w & \longrightarrow & \bigcirc \rightarrow \\ & & \bigotimes \end{array}$$

Fraction increase in load $= r$

Interleaved Declustering

	cluster 0				cluster 1			
Node	0	1	2	3	4	5	6	7
Primary copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup copy		r0.0	r0.1	r0.2		r4.0	r4.1	r4.2
	r1.2		r1.0	r1.1	r5.2		r5.0	r5.1
	r2.1	r2.2		r2.0	r6.1	r6.2		r6.0
	r3.0	r3.1	r3.2		r7.0	r7.1	r7.2	

$$\left.\begin{array}{ll} \lambda_r + \lambda_w & \longrightarrow \bigcirc \rightarrow \\ \lambda_r + \lambda_w & \longrightarrow \bigcirc \rightarrow \\ & \cdot \\ & \cdot \\ & \cdot \\ \lambda_r + \lambda_w & \longrightarrow \bigcirc \rightarrow \end{array}\right\} \text{cluster}$$

C =cluster size

With one failed disk:

- load on each surviving disk $= \lambda_r + \lambda_w + \frac{1}{C-1}\lambda_r$
- fraction increased load $= \frac{r}{C-1}$

Chained Declustering

Node	0	1	2	3	4	5	6	7
Primary copy	R0	R1	R2	R3	R4	R5	R6	R7
Backup copy	r0	r1	r2	r3	r4	r5	r6	r7

$\lambda_r + \lambda_w \longrightarrow \bigcirc \rightarrow$	$\lambda_r + \lambda_w + \frac{\lambda_r}{N-1} \longrightarrow \bigcirc \rightarrow$
.	.
.	.
.	.
$\lambda_r + \lambda_w \longrightarrow \bigcirc \rightarrow$	$\lambda_r + \lambda_w + \frac{\lambda_r}{N-1} \longrightarrow \bigcirc \rightarrow$
$\lambda_r + \lambda_w \longrightarrow \bigcirc \rightarrow$	$\otimes$

Node	0	1	2	3	4	5	6	7
Primary copy	1R0	-	$\frac{1}{7}$R2	$\frac{2}{7}$R3	$\frac{3}{7}$R4	$\frac{4}{7}$R5	$\frac{5}{7}$R6	$\frac{6}{7}$R7
Backup copy	$\frac{1}{7}$r7	-	1r1	$\frac{6}{7}$r2	$\frac{5}{7}$r3	$\frac{4}{7}$r4	$\frac{3}{7}$r5	$\frac{2}{7}$r6

RAID N+1

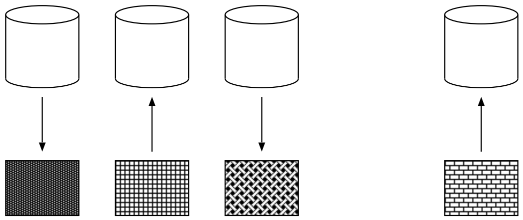
Example: $N + 1 = 5$

	disk-1	disk-2	disk-3	disk-4	disk-5
block 0	$D_{0,1}$	$D_{0,2}$	$D_{0,3}$	$D_{0,4}$	C_0
block 1	$D_{1,1}$	$D_{1,2}$	$D_{1,3}$	C_1	$D_{1,4}$
block 2			C_2		
block 3		C_3			
block 4	C_4				
block 5					C_5
	...	...	...	...	...

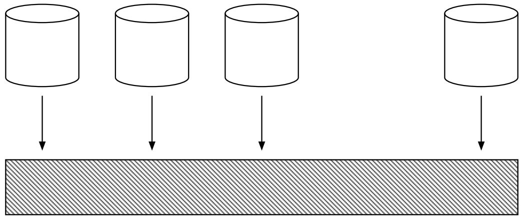
$$C_i = \oplus_{j=1}^{4} D_{ij}$$

$$D_{ik} = \oplus_{j \neq k} D_{ij} \oplus C_i$$

RAID (cont')



(a) independent reads and writes simultaneously



(b) for large transfer; can amortize latency over large transfer per disk; scientific, supercomputer applications.

We concentrate on (a).

RAID: N+1

$$\left.\begin{array}{c} \lambda_r + \lambda_w \longrightarrow \bigcirc \rightarrow \\ \lambda_r + \lambda_w \longrightarrow \bigcirc \rightarrow \\ \cdot \\ \cdot \\ \cdot \\ \lambda_r + \lambda_w \longrightarrow \bigcirc \rightarrow \end{array}\right\} \text{“cluster” = group size = G}$$

With one failed disk:

- load on each surviving disk
  $= \lambda_r + \lambda_w$(normal) $+$
  λ_r(each read to failed disk becomes a read to each surviving disk)
- fraction increased load $= r$

New Scheme: RAID N+1/C

- objective: less degradation under failure
- allow groups to be spread over larger number of disks, C
- max. of $\begin{pmatrix} C \\ G \end{pmatrix} * \text{G}$ distinct group arrangements.

Example: G=4, C=5

	Disk-1	Disk-2	Disk-3	Disk-4	Disk-5
0	A	A	A	A	-
1	B	B	B	-	B
2	C	C	-	C	C
3	D	-	D	D	D
4	-	E	E	E	E

idea: decouple group size from cluster size

Example: G=5 C=10

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
A	A	A	A	P					
A	A	A	P	A					
A	A	P	A	A					
A	P	A	A	A					
P	A	A	A	A					
		B			B	B	B	P	
		B			B	B	P	B	
		B			B	P	B	B	
		B			P	B	B	B	
		P			B	B	B	B	
			C	C		C		C	P
			C	C		C		P	C
			C	C		P		C	C
			C	P		C		C	C
			P	C		C		C	C
.	.	.							
.	.	.							
.	.	.							

Mean Time to Data Unavailability

- Data Unavailable if two disks down in a "cluster".
- RAID N+1/C has fewer, bigger clusters.
- more combinations of failed disks that cause data unavailability

Max Increased Load with Single Failed Disk in a Cluster

- $r\frac{G-1}{C-1}$
- use $C > G$ to reduce impact of failure

Cluster size vs. Mean time to data unavailability

Online spares are assumed
Automatic rebuild of failed disk.

Two opposing effects:

1. larger cluster size $\rightarrow$ higher rate of second failure in a cluster
2. larger cluster size $\rightarrow$ faster rebuild of failed disk contents

Also, assume there is some system specified minimum load per disk: $\lambda_0 = \lambda_r + \lambda_w$

- a high λ_0 can impact the rebuild rate,
- a larger cluster size can accommodate a larger λ_0.

Analysis of RAID N+1/C

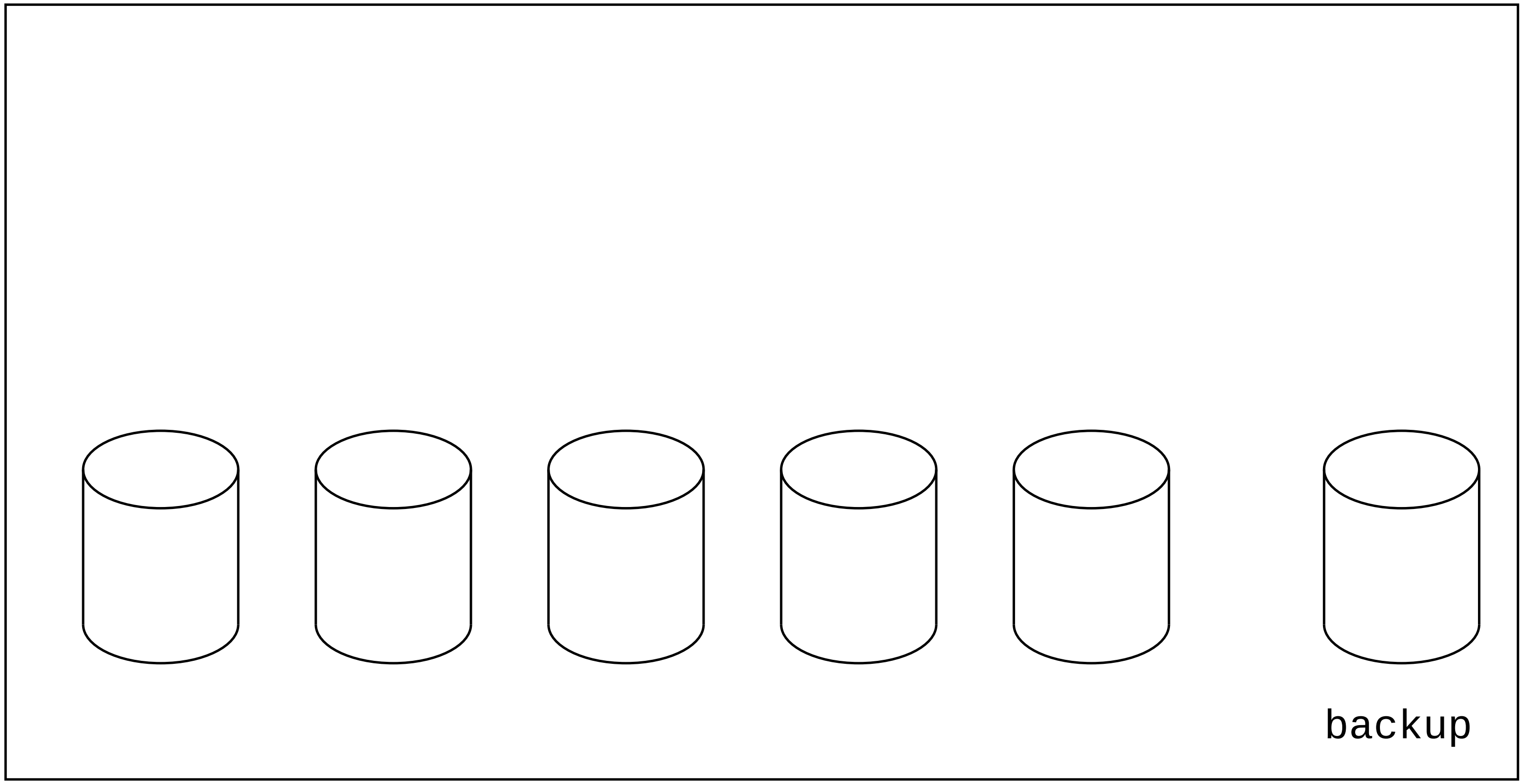
Assumptions:

- disks are system bottleneck
- uniform data access pattern
- 'fluid flow' analysis
- all writes to failed disk are written to standby disk

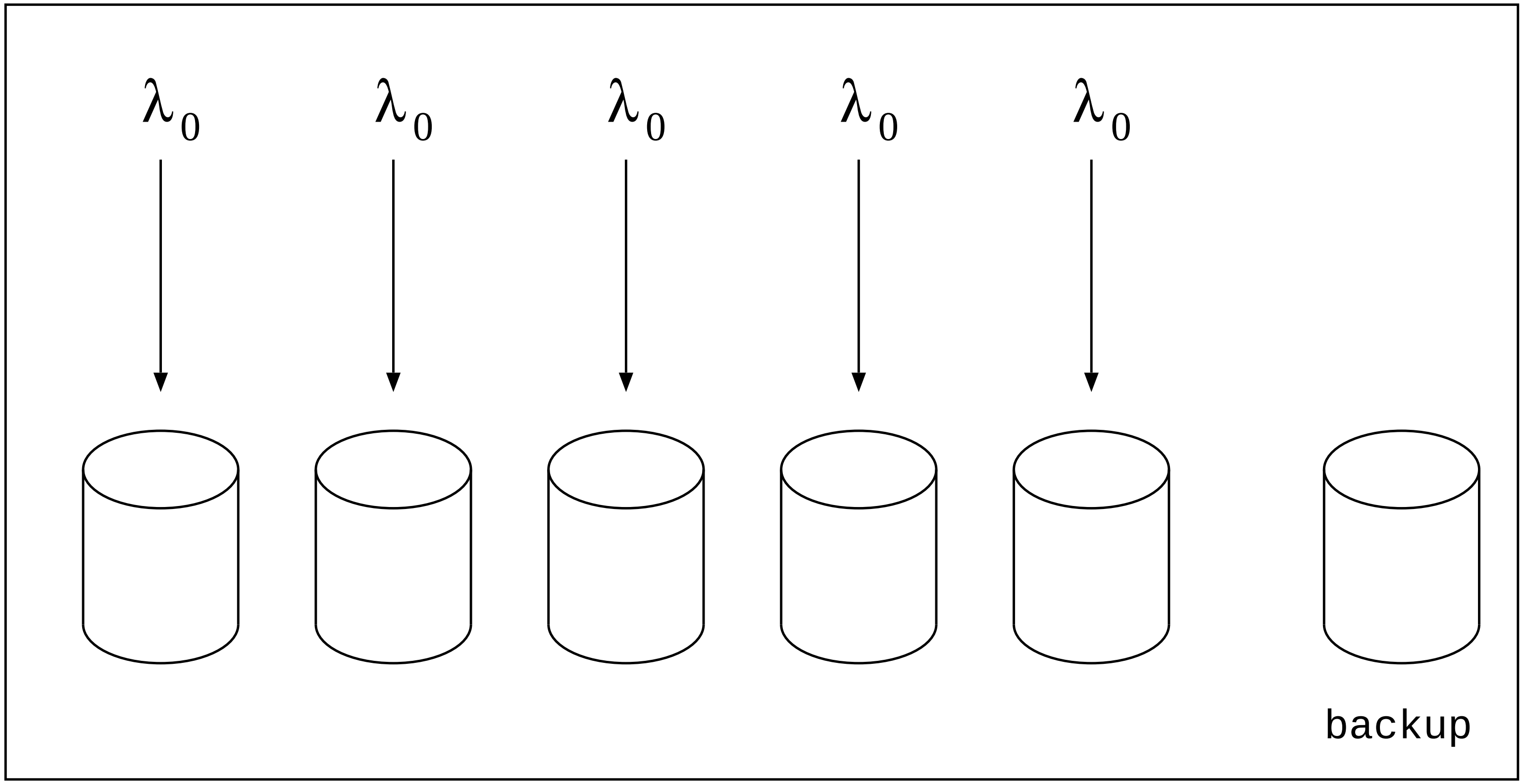
Summary of Parameters

- G determines storage overhead and the number of reads required to reconstruct a data block from a failed disk.
- C determines the mean time to data unavailability
- $r\frac{G-1}{C-1}$ determines increased load on surviving disks when one disk fails.

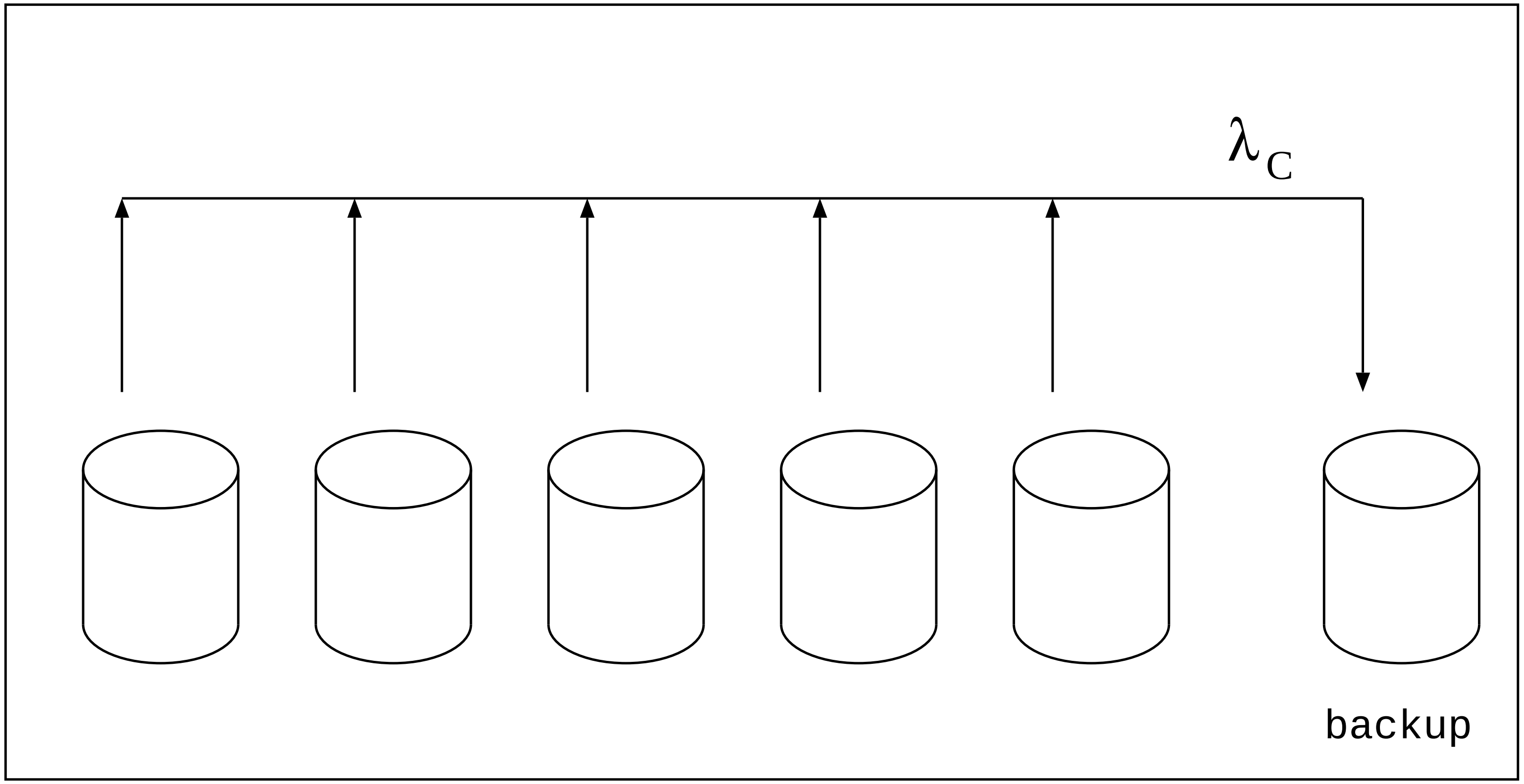
REBUILDING FAILED DISK



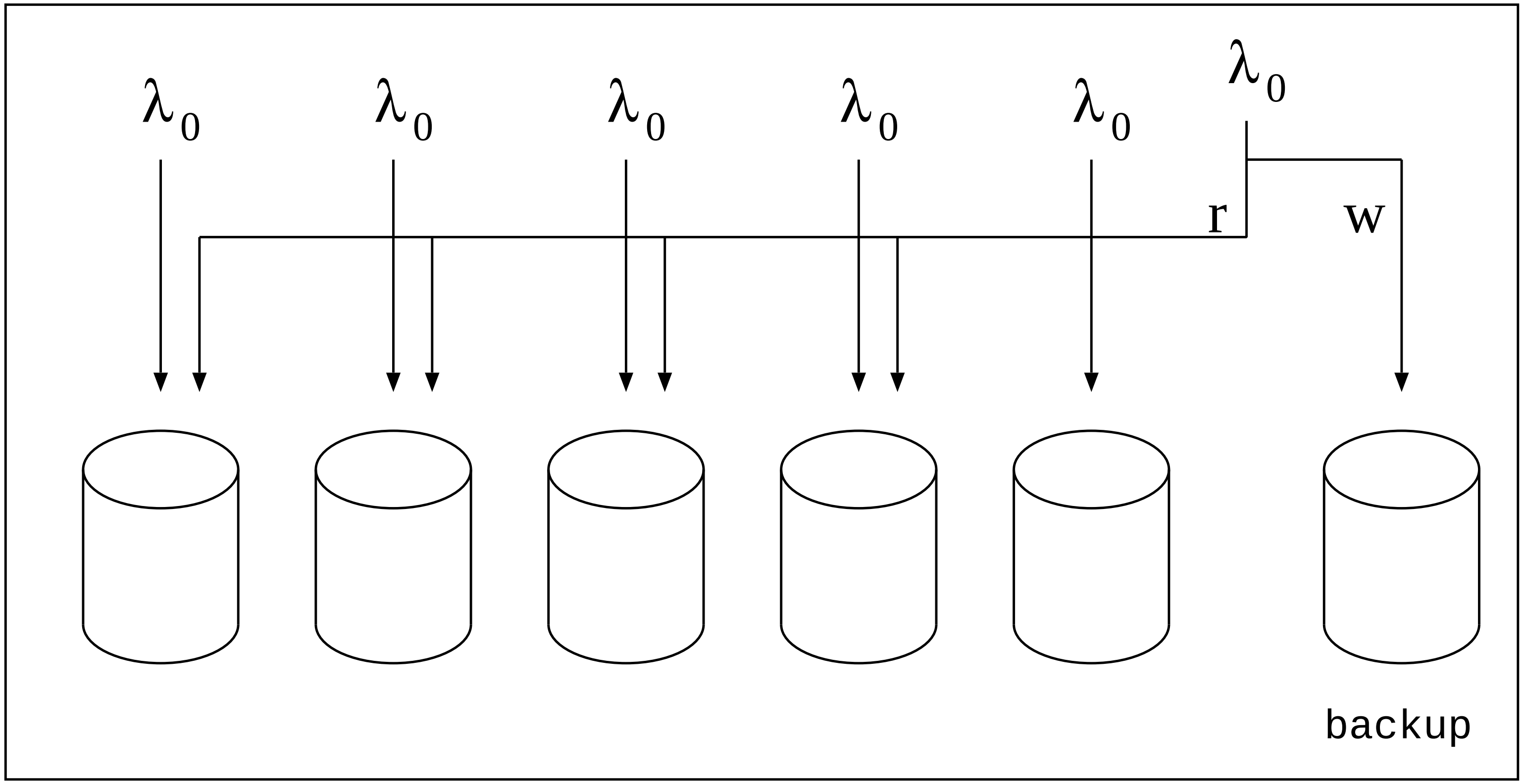
REBUILDING FAILED DISK



REBUILDING FAILED DISK



REBUILDING FAILED DISK



Baseline Case

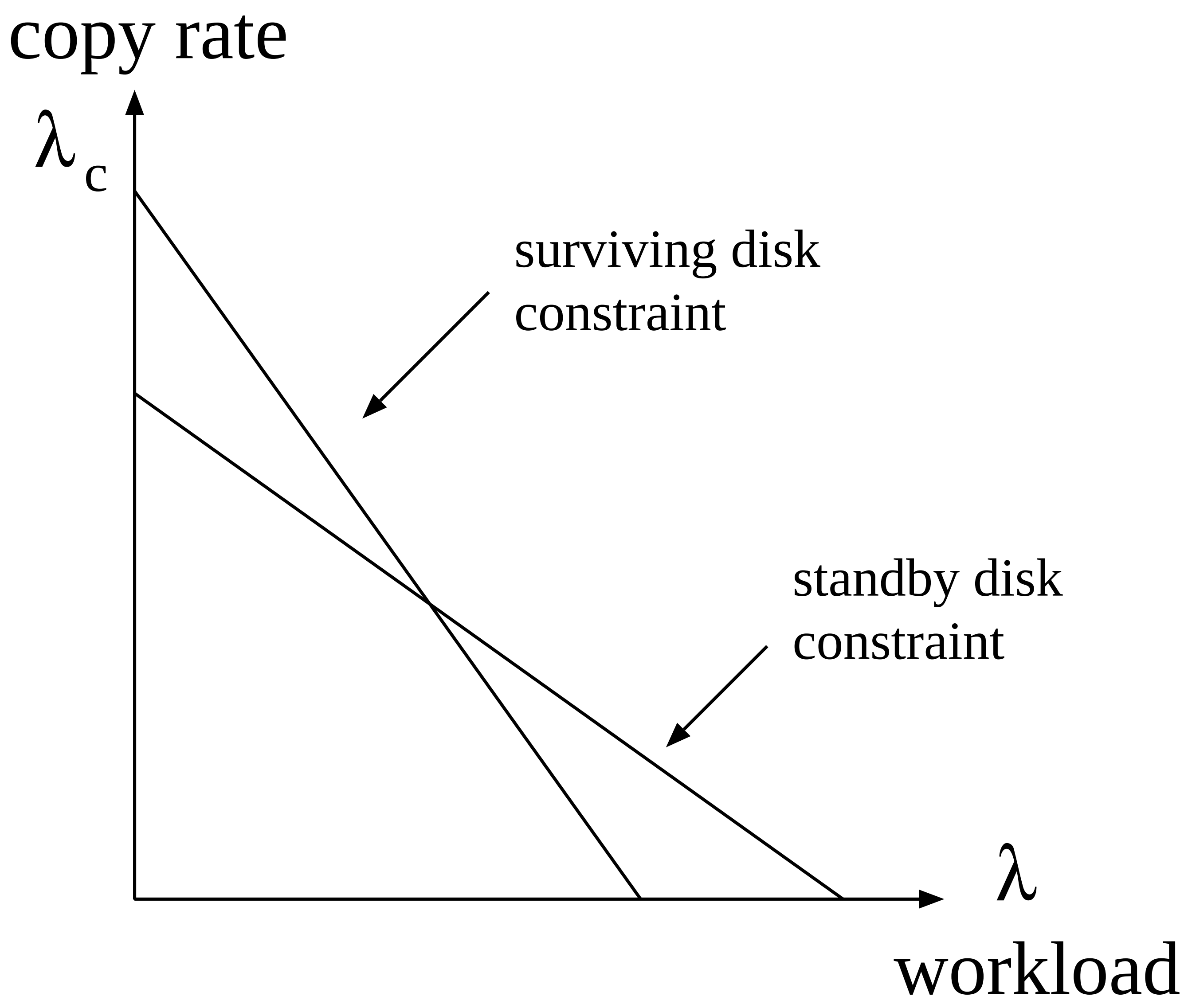
- all writes to standby disk
- all reads from surviving disks

Constraints:

1. surviving disks: $\mu \geq \lambda_0(t) + \alpha\lambda_0(t) + \alpha\lambda_c(t)$
2. standby disk $\mu \geq \lambda_c(t) + \omega\lambda_0(t)$

where:

$$\begin{aligned} \mu &= \textit{max I/O rate per disk} \\ \alpha &= \frac{G-1}{C-1} \end{aligned}$$


copy rate
λ_c
surviving disk constraint
standby disk constraint
λ
workload

Three rebuild strategies studied

- Baseline
- Dynamic rerouting of reads – some reads to failed disk can be routed to standby.
- Dynamic rerouting + 'piggy-backing' – when the normal workload requires a block from the failed disk to be reconstructed, 'capture' the block and copy it

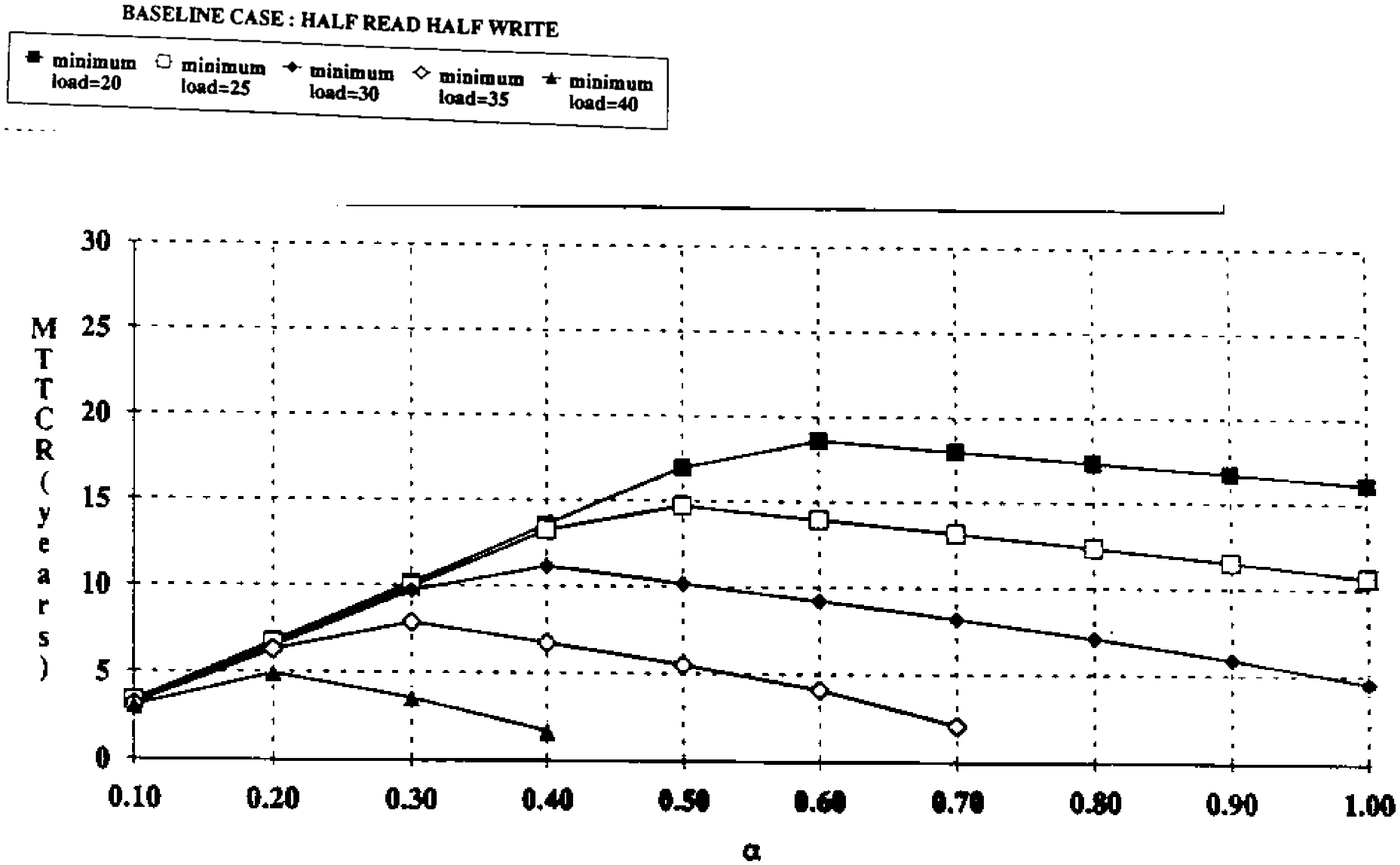
$$\mu \geq \lambda(t) + \alpha r \lambda(t)[1 - b(t)f(t)] + \alpha \lambda_c(t) \quad \text{(C1)}$$

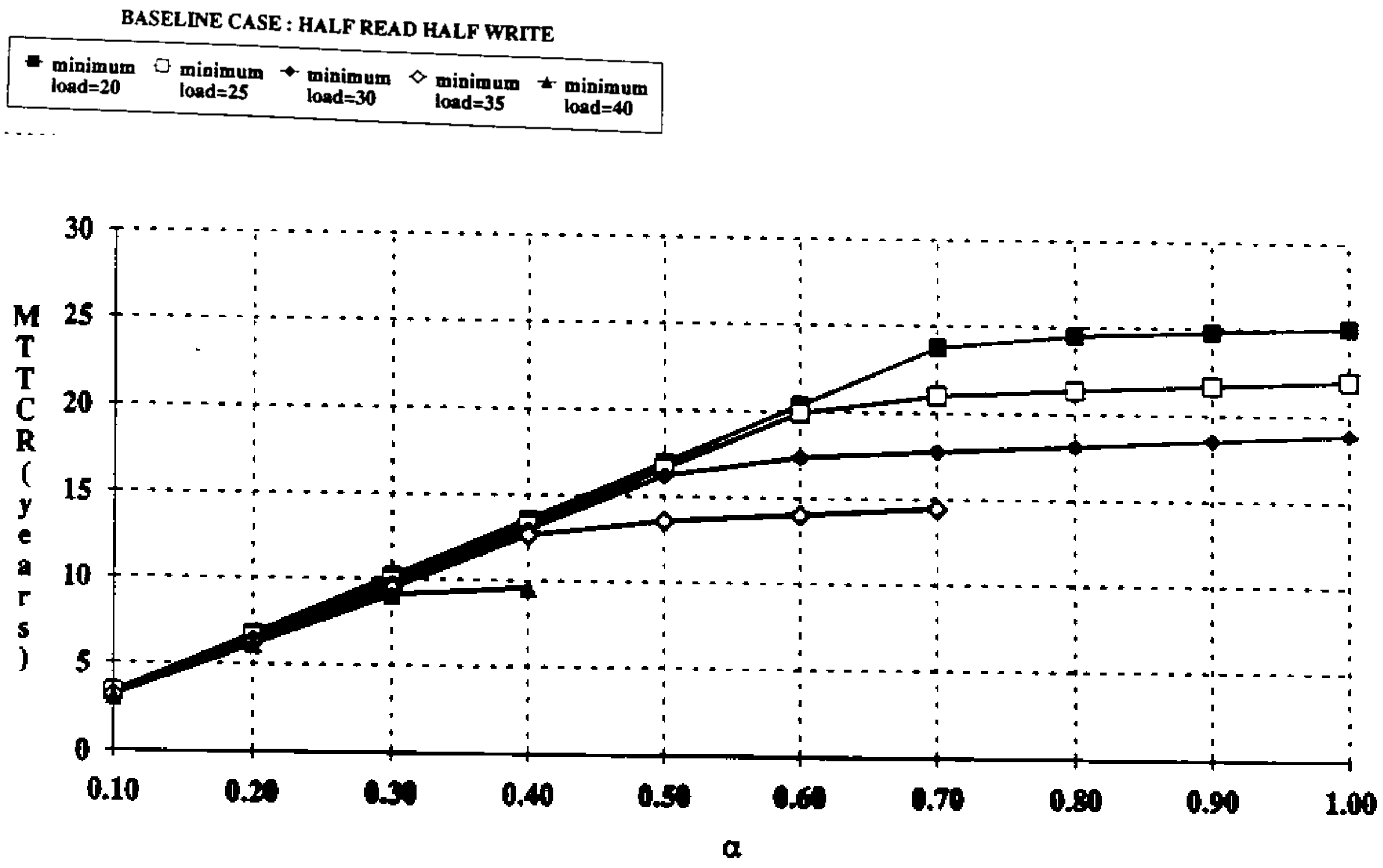
$$\mu \geq \lambda_c(t) + w\lambda(t) + r\lambda(t)b(t)f(t) \quad \text{(C2)}$$

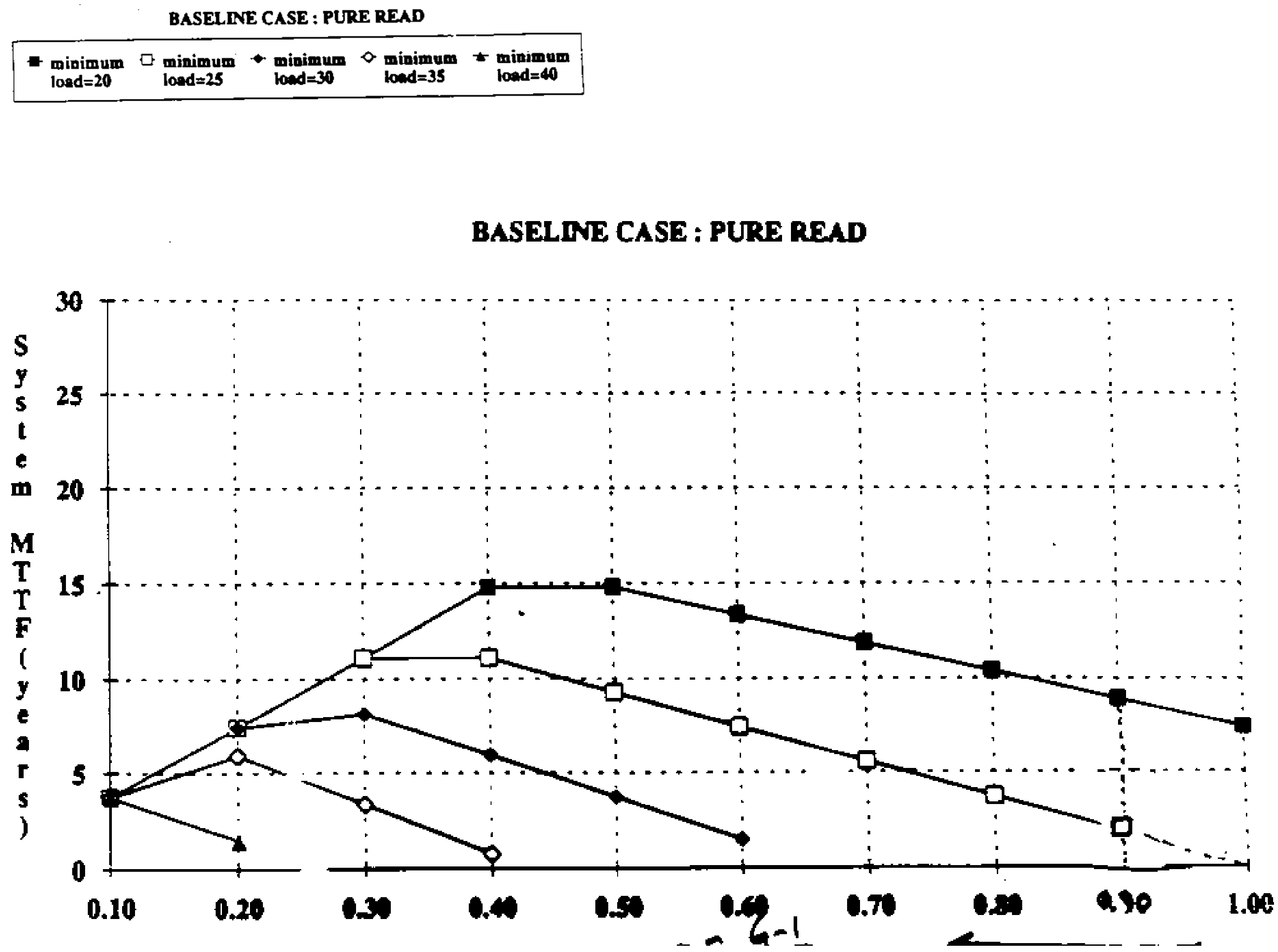
$$f(t) = \frac{1}{I} \int_0^t \{\lambda_c(\tau) + \omega\lambda(\tau)[1 - f(\tau)] + r\lambda(\tau)[1 - f(\tau)]\} d\tau$$

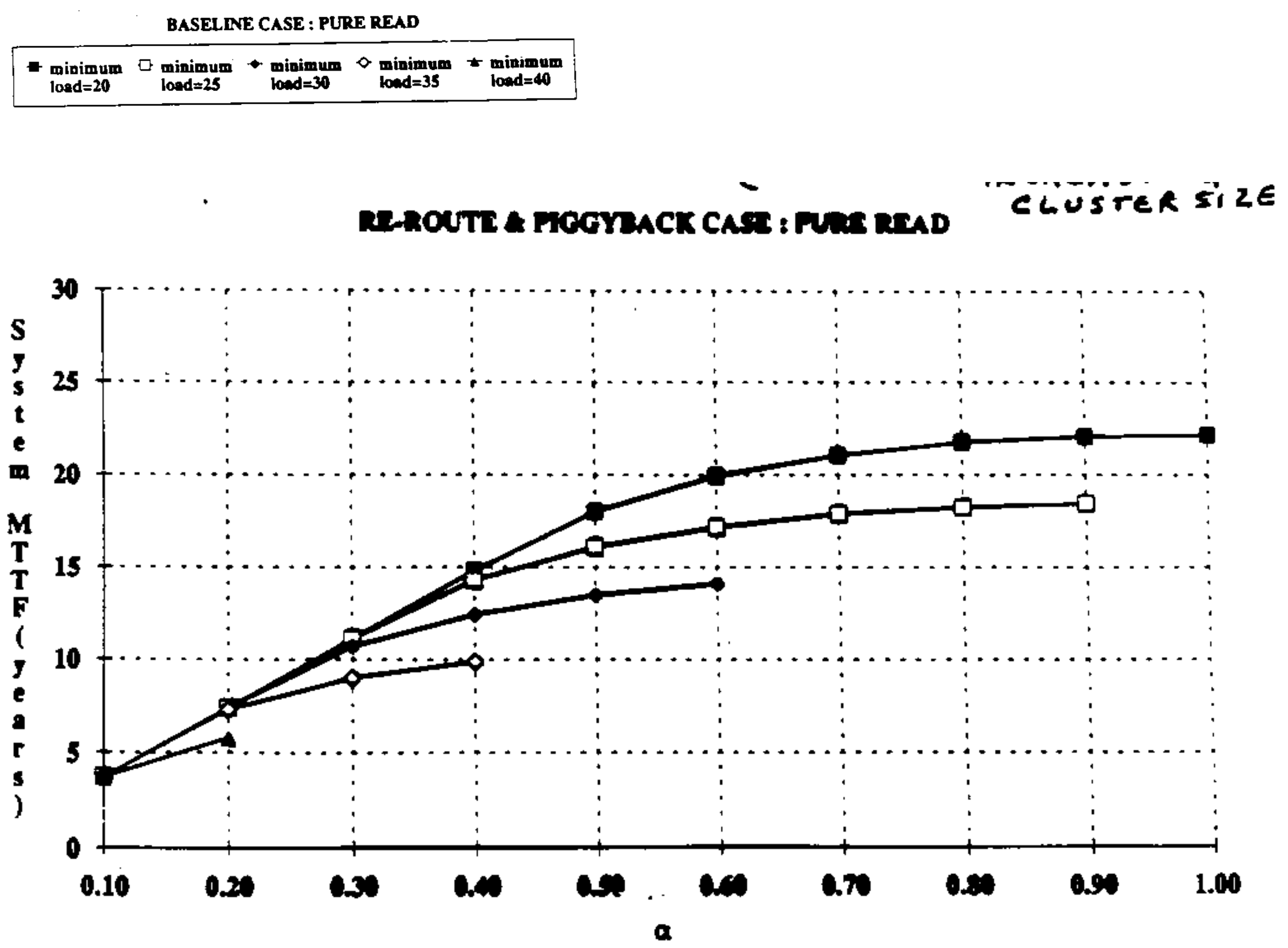
where:

$$\alpha = \frac{G-1}{C-1}$$


BASELINE CASE : HALF READ HALF WRITE
minimum load=20
minimum load=25
minimum load=30
minimum load=35
minimum load=40
MTTCR (years)
30
25
20
15
10
5
0
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
α


BASELINE CASE : HALF READ HALF WRITE
minimum load=20
minimum load=25
minimum load=30
minimum load=35
minimum load=40
MTTCR (years)
30
25
20
15
10
5
0
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
α


BASELINE CASE : PURE READ
minimum load=20
minimum load=25
minimum load=30
minimum load=35
minimum load=40
BASELINE CASE : PURE READ
System MTTF (years)
30
25
20
15
10
5
0
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00


BASELINE CASE : PURE READ
minimum load=20
minimum load=25
minimum load=30
minimum load=35
minimum load=40
CLUSTER SIZE
RE-ROUTE & PIGGYBACK CASE : PURE READ
System MTTF (years)
30
25
20
15
10
5
0
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
α

Mapping Buddies

If the Cluster size (C) is large compare to the group size (G), then it is not a trivial matter to map the buddies for each parity group.

Approaches:

- mapping algorithm
- table lookup

Balanced Incomplete Block Design

Grouping of v kinds of object into b blocks s.t.:

1. each block contain k distinct object.
2. each object occurs in r different blocks.
3. any pair of different object occurs together in λ blocks.

$$
\begin{aligned}
vr &= bk \\
r(k-1) &= \lambda(v-1)
\end{aligned}
$$

Solution

Assume we have C disks, What we want:

- each disk has M domains.
- every two domains of a parity group are in two different disks.
- every pair of disks has exactly L parity groups in common.

Mapping

$$(b, v, r, k, \lambda)$$
$$Object \longleftrightarrow Disk$$
$$block \longleftrightarrow ParityGroup$$

We set:

1. $C = v$
2. number of parity group $= b$.
3. size of each parity group $G = k$.
4. number of domain per disk $M = r$.
5. number of parity groups common to any pair of disks $L = \lambda$.

$$(b = 15, v = 10, r = 6, k = 4, \lambda = 2)$$

Blocks

A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q
1	1	1	1	3	2	2	1	2	3	4	1	2	3	4
2	3	4	2	4	4	3	5	6	5	5	8	5	6	7
5	6	7	3	5	6	7	6	7	7	6	9	9	8	8
8	9	10	4	8	9	10	7	8	9	10	10	10	10	9

Disks

D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}
A	A	B	C	A	B	C	A	B	C
B	D	D	D	E	F	G	E	F	G
C	F	E	E	H	H	H	J	K	L
D	G	G	F	K	J	J	M	M	M
H	J	K	L	L	L	K	P	N	N
M	N	P	Q	N	P	Q	Q	Q	P

Variable Parity Group Size

Combining two or more BIBD's with the same parameter v.

Given $(b_1, v, r_1, k_1, \lambda_1)$ and $(b_2, v, r_2, k_2, \lambda_2)$

We set:

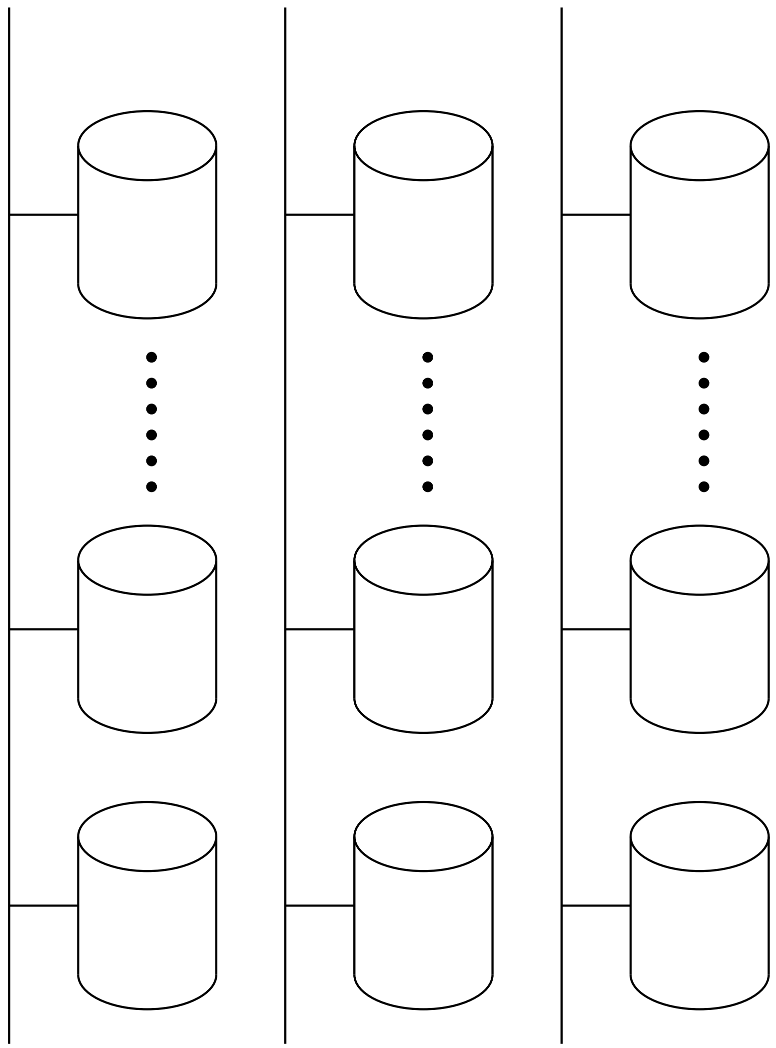
1. $C = v$.
2. number of parity group $= b_1 + b_2$.
3. size of each parity group k_1 or k_2.
4. number of domain per disk $M = r_1 + r_2$
5. number of parity groups common to any pair of disks $L = \lambda_1 + \lambda_2$.

$$(b_1 = 7, v = 7, r_1 = 3, k_1 = 3, \lambda_1 = 1)$$
$$(b_2 = 7, v = 7, r_2 = 6, k_2 = 6, \lambda_2 = 5)$$

Disks

D_1	D_2	D_3	D_4	D_5	D_6	D_7	$PGsize$
A	A	A					3
B			B	B			3
	C		C		C		3
		D		D	D		3
E					E	E	3
	F			F		F	3
		G	G			G	3
H	H	H	H	H	H		6
J	J	J	J	J		J	6
K	K	K	K		K	K	6
L	L	L		L	L	L	6
M	M		M	M	M	M	6
N		N	N	N	N	N	6
	P	P	P	P	P	P	6

Orthogonal Array



1. each disk is partitioned into M domains
2. any two domains of a parity group are located in two different disks;
3. every pair of disks in two different strings have exactly L parity groups in common;
4. every pair of disks in the same string do not have any parity group in common

$(b = 24, v = 4, r = 18, k = 3, \lambda = 12)$ $v = String$

D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
A		A		A			
B			B			B	
C					C		C
	D			D		D	
	E		E		E		
	F	F					F
		G			G	G	
			H	H			
I		I		I			
J		J		J			
K			K			K	
L			L			L	
M					M		M
N					N		N
	O			O		O	
	P			P		P	
	Q		Q		Q		
	R		R		R		
	S	S					S
	T	T					T
		U			U	U	
		V			V	V	
			W	W			W
			X	X			X

Conclusion

1. Clustered RAID is better.
2. Disk rebuilding algorithm.
3. Parity Group mapping.
4. Orthogonal Array.
5. Log-structured Array.