Westfälische Hochschule

- Information Redundancy -

Christoph Hoopmann
Michael Krane

Agenda

I. Kodierungsgrundlagen

II. Parity Codes & Checksums

III. M-of-N & Berger Codes

IV. Cyclic Codes

V. Arithmetic Codes

VI. RAID Levels

VII. Data replication

VIII. Primary Backup Approach

IX. Algorithm-based Fault Tolerance

Einführung

Übertragung/Speichern von Daten ist fehleranfällig

Redundanz nötig um Fehler zuerkennen

1. Teil: Grundlagen von Kodierungen

2. Teil: Anwendung am Beispiel von RAID

3. Teil: Daten Replikation in verteilten Systemen

Westfälische Hochschule

I
Kodierungsgrundlagen

Kodierung

d-bit Datenwort wird zu c-bit Codewort kodiert $c > d$

Mehr Codewörter als Datenwörter ==> nicht gültige Codewörter

Dekodierungs-Versuch von nicht gültigem führt zu Fehler

Einteilung der Methoden anhand folgender Parameter

- Anzahl erkennbarer/korrigierbarer Fehler
- Overhead von Speicher und Laufzeit

Anzahl erkennbarer Fehler

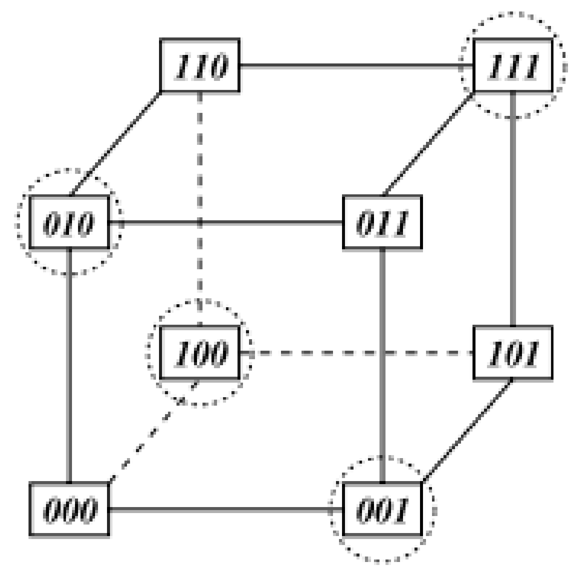
Hamming-Distanz : Anzahl Unterschiede in zwei Code-Wörtern

k Fehler erkennen : H_{dist} >= k + 1

k Fehler beheben : H_{dist} >= 2k +1

{001,010,100,111} => H_{dist} = 2

{000,111} => H_{dist} = 3



Overhead

Speicher:

1 bit(Parität) <= Overhead <=d bits (M-of-2M)

Laufzeit:

- Separierbar vs. Nicht separierbar

  $d_3d_2d_1d_0p_2p_1p_0$ nach d_0 abschneiden

  $c_7c_6c_5c_4c_3c_2c_1c_0$ Umwandeln mittels HW/SW

Westfälische
Hochschule

II
Parity Codes & Checksums

Parität Codes

Extra Bit am Ende, sodass Anzahl 1-bits (un)gerade

Abhängig davon ob Alles-0 oder Alles-1 Fehler wahrscheinlicher

0001 => 00011

Kodierung/Error-Check mittels XOR-Verknüpfung

$a0 \oplus a1 \oplus a2 \oplus a3 = P$

$a0 \oplus a1 \oplus a2 \oplus a3 \oplus P = error_{sig}$ [0 = kein Error]

separierbar, Overhead 1 bit, $H_{dist} = 2$

Varianten Parität-Codes

Fehler-Korrektur mittels überlappender Parität

0	0	0	1	1	1	**1**
1	0	1	0	1	1	**0**
1	1	0	0	0	0	**0**
0	0	0	1	1	1	**1**
1	1	1	1	1	1	**0**
1	**0**	**0**	**1**	**0**	**0**	**0**

m * n Datenwort in Matrix, m + n + 1 Paritätbits

- Alle Bitfehler erkennbar, 1 Bitfehler korrigierbar
- Großer Overhead

Alle Paritätbits nötig?

d + r mögliche Fehler-Positionen + kein Fehler

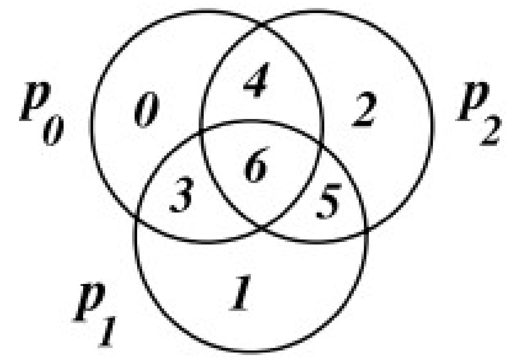
2^r >= d + r + 1

Varianten Parität-Codes (2)

(7,4) SEC (single-error correcting) code

d = 4 Bits $a_3a_2a_1a_0$ ==> 2^3 >= 4 + 3 + 1 => 3 Bits

c = 7 Bits $a_3a_2a_1a_0p_2p_1p_0$



Falsche Paritätbits weisen auf Fehler

- $p_2p_1p_0 \oplus p'_2p'_1p'_0$

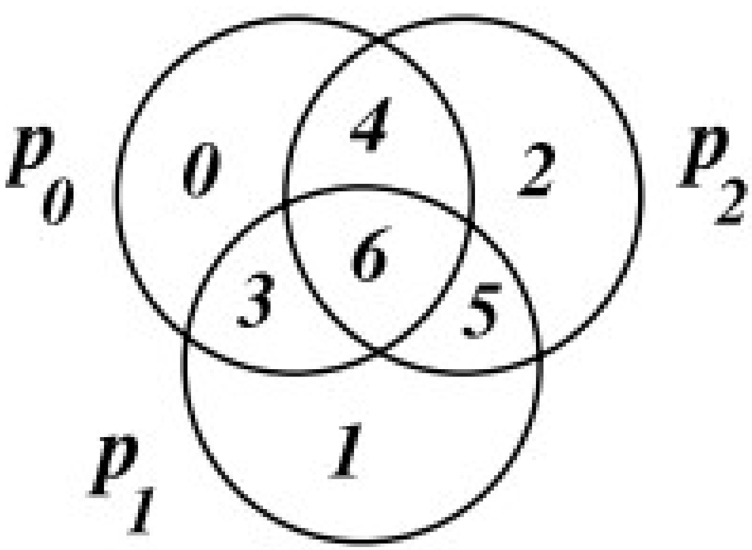
(8,4) SEC/DED (double-error detecting)

(7,4) SEC wird um normales Paritätsbit erweitert

- c_3 == 1 => Fehlercode korrekt

Varianten Parität-Codes (3)

Berechnung Fehler-Code in einem Schritt



$$
\begin{array}{c}
\begin{array}{ccccccc} a_3 & a_2 & a_1 & a_0 & p_2 & p_1 & p_0 \end{array} \\
\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}
\end{array}
\begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} s_2 s_1 s_0 \end{bmatrix}
$$

Checksum

Fehler-Erkennung in Datenübertragung in Netzwerken

Summe der Daten mod n wird verglichen

d-Bit Wörter: n = 2^d oder n = 2^(2d)

0000	0000	0000	
0101	0101	0101	
1111	1111	1111	00000101
0010	0010	0010	11110010
$\boxed{0110}$	$\boxed{00010110}$	$\boxed{0111}$	$\boxed{11110111}$
(a) Single-precision	(b) Double-precision	(c) Residue	(d) Honeywell

Residue: LSB += $carry_{MSB}$

separierbar, Overhead d oder 2d bits

Westfälische
Hochschule

III
M-of-N & Berger Code

M-of-N-Kodierung

Unidirektional (0 → 1 Fehler oder 1 → 0 Fehler)

N-Bit Codewörter mit M gesetzten Bits

Digit	Codeword
0	00011
1	00101
2	00110
3	01001
4	01010
5	01100
6	10001
7	10010
8	10100
9	11000

1 Bitfehler ändert M um +1 oder -1

nicht separierbar, Overhead (N-d) bit

M-of-N-Kodierung (2) - M-of-2M

d Bits werden angehängt, sodass M Bits gesetzt

111001 => 111001000011 (6-of-12)

separierbar, Overhead d Bits

Berger Code

Anzahl gesetzte Bits als Komplement angehängt

11101 => 11101011

separierbar, Overhead ceil($\log_2$(d+1)) Bits

Westfälische Hochschule

IV
Cyclic Codes

Cyclic Codes

Code-Wörter sind „geshifted“

- $a_3a_2a_1a_0$ Codewort, dann $a_0a_3a_2a_1$ auch

Kodierung: d-Bit Datenwort * Konstante mod 2

Dekodierung: c-Bit Codewort / Konstante

- Rest == 0 => Kein Fehler / Rest != 0 => Fehler
- Erkennt einzelne Bit-Fehler, und (c – d – 1) lange Fehler-Blöcke

nicht separierbar, Overhead (c - d) Bits

Cyclic Code (2) - Konstante

Konstante für (c,d) Cyclic Code

- (c - d +1) Bit Zahl, als Polynom (c - d) Grad
- Faktor von $x^c + 1$

(15,11) Code $G(x) = x^4 + x^3 + 1$, d=100 0110 0101

Kodierung: G(x) als Zahl = 11001

```
    10001100101
×         11001
---------------
    10001100101
   00000000000
  00000000000
 10001100101
10001100101
---------------
110000100011101
```

c=110 0001 0001 1101

Cyclic Code (3) – Fehler Erkennung

|Fehlerblock| <= 3, ansonsten nicht

110000**011**011101 : 11001 = 10001110011

```
11001
-----
    10011
    11001
    -----
     10100
     11001
     -----
      11011
      11001
      -----
          10110
          11001
          -----
           11111
           11001
           -----
           00110
```

1100001**11**01**0**101 : 11001 = 10001101101

```
11001
-----
    10111
    11001
    -----
     11100
     11001
     -----
       10110
       11001
       -----
        11111
        11001
        -----
          11001
          11001
          -----
          00000
```

Westfälische Hochschule

V
Arithmetic Codes

Arithmetic Codes – AN Codierung

Erkennen von 1 Bit Fehler in der ALU

Operanten sind codiert, Ergebnis behält Codierung

AN – Codierung

- Multiplikation und Addition : X' = A * X; Y' = A * Y
- Fehler-Erkennung: X' und Y' Vielfache von A =>
  Ergebnis auch Vielfaches von A
- Fehler nicht erkennbar wenn Anzahl mod A == 0

nicht separierbar, Overhead ceil($\log_2$(A)) Bits

Arithmetic Codes (2) - Residue Code

Operanten werden mit $C(X) = c_n \ldots c_0$ erweitert

$a_3a_2a_1a_0 \Longrightarrow a_3a_2a_1a_0C(X)$

- $C(X) = X \bmod A = |X|_A$
- Addition => $|X + Y| = ||X|_A + |Y|_A|_A$
- Multiplikation => $|X * Y| = ||X|_A * |Y|_A|_A$
- Division: $X - R = Q * Y \Longrightarrow ||X|_A - |R|_A|_A = ||Q|_A * |Y|_A|_A$

Extra ALU Operation nötig

separierbar, Overhead $\mathrm{ceil}(\log_2(A))$ Bits

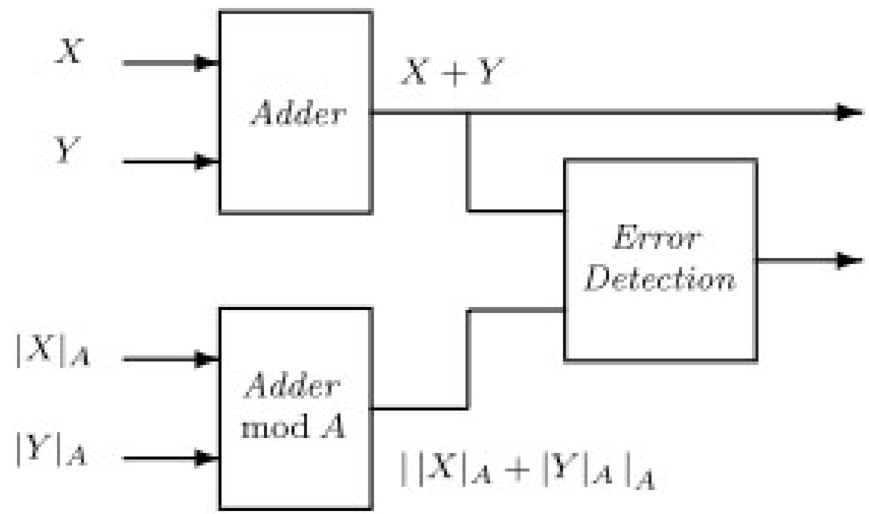
Arithmetic Codes (3) - Residue Code

Fehler-Erkennung benötigt extra ALU-Operation

Ergebnisse Vergleichen:

== ==> Ergebniss korrekt

!= ==> Ergebnis inkorrekt



Westfälische Hochschule

Arithmetic Codes (4) - Residue Code Bsp

$A = 3, X = 7, Y = 5 ==> |X|_3 = 1$ und $|Y|_3 = 2$

Addition: $|7 + 5|_3 = 0 = ||X|_3 + |Y|_3|_3 = |1 + 2|_3$

Mult: $|7 * 5|_3 = 2 = ||X|_3 * |Y|_3|_3 = |1 * 2|_3$

$Q = X/Y = 1$, Rest $= 2$

Divison: $||X|_3 - |Rest|_3|_3 = |1 - 2|_3 = 2$

$||Y|_3 * |Q|_3|_3 = |5 * 1|_3 = 2$

Westfälische Hochschule

VI
RAID levels

V RAID levels

Redundant **A**rray of **I**ndependent **D**isks

Ausfallsicherheit

IO Performance

Kosteneffizienz

Hot-Swap

Speichergröße

Westfälische Hochschule

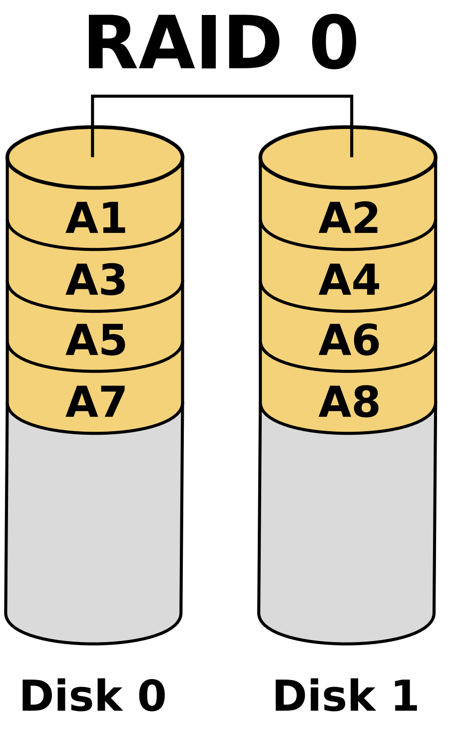
V RAID levels

RAID-0	**Stripe, Granularität**
RAID-1	**Mirror**
RAID-2	Bit-Stripe, Hamming Code
RAID-3	Byte-Stripe, zentr. Parität
RAID-4	Block-Stripe, zentr. Parität
RAID-5	**Bl. Stripe, vert. Parität**
RAID-6	Bl. Stripe, dp. vert. Parität

Striping n disks, Granularität

Keine Redundanz

#Disks * R-rate, 1 W-rate

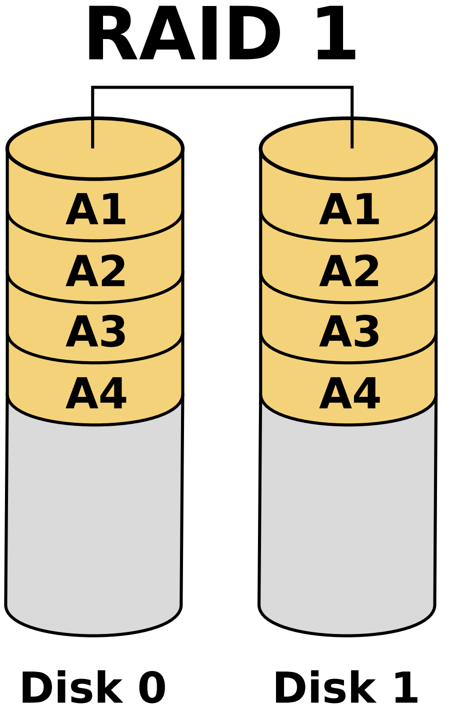


- ➕ Seq. R/W
- ➕ Random R/W
- ➖ Disk failure

Mirror n disks

Vollständige Redundanz

May #Disks * R-rate, 1 W-rate



- ➕ Seq. R/W
- ➕ Random R/W
- ➖ Kostspielig, Speichergr.

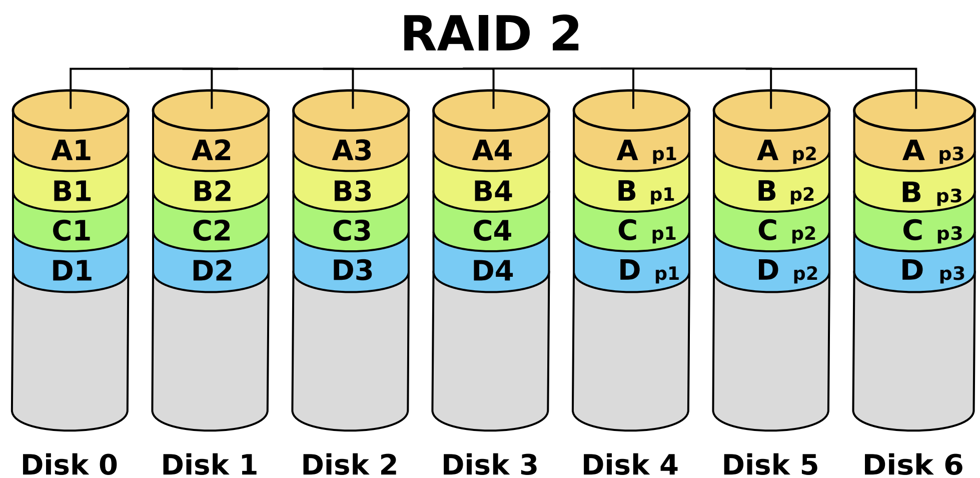
RAID-2

Bit-level striping

Hamming-Code

$\lceil \log_2 d + 1 \rceil =$ 4 data bit → 3 bit Parity

- Durchsatz
- Speichergr.
- Kostspielig

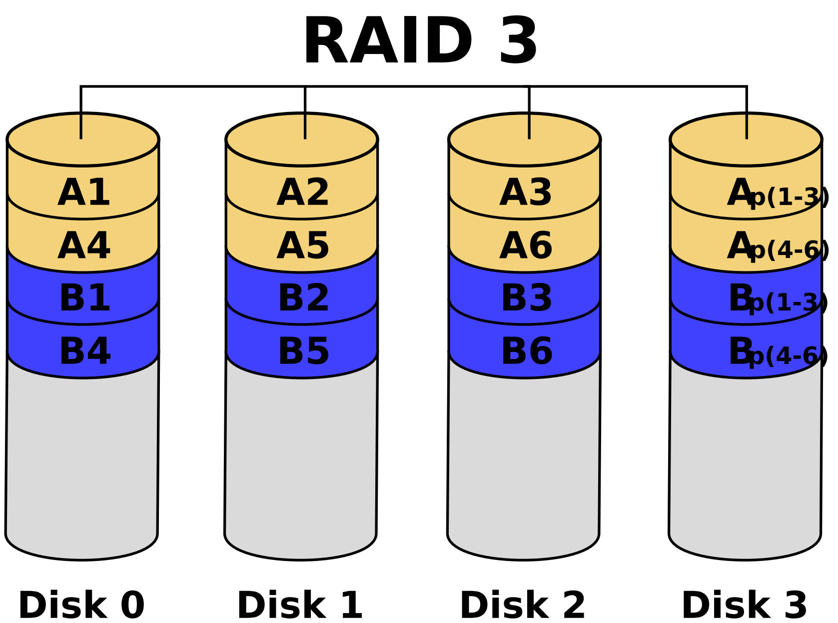


RAID-3

Byte-level striping

XOR-Parität separat

- ➕ nur 1x P-disk
- ➕ Seq. R/W
- ➖ Random W
- ➖ W-IO P-disk

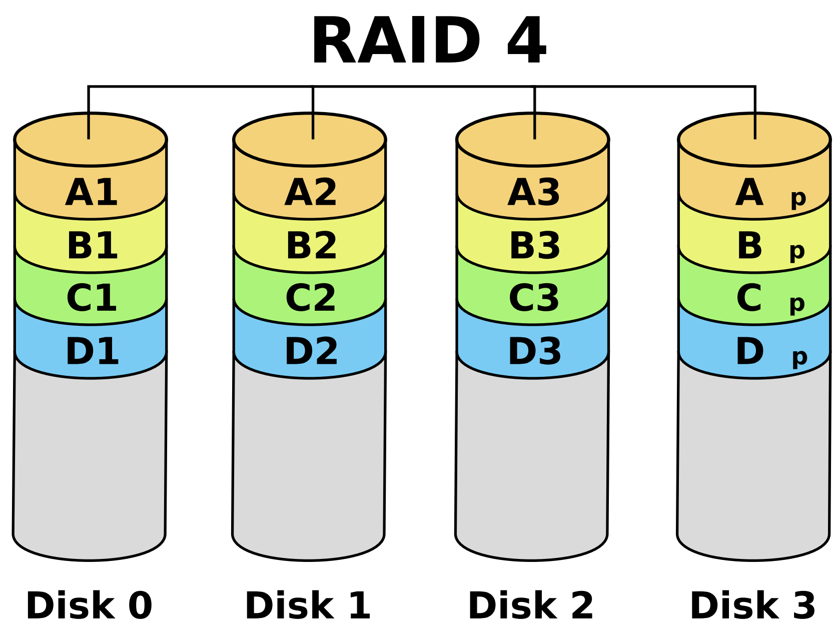


RAID-4

Block-level striping

XOR-Parität separat

- Seq. R/W
- Random W
- W-IO P-disk

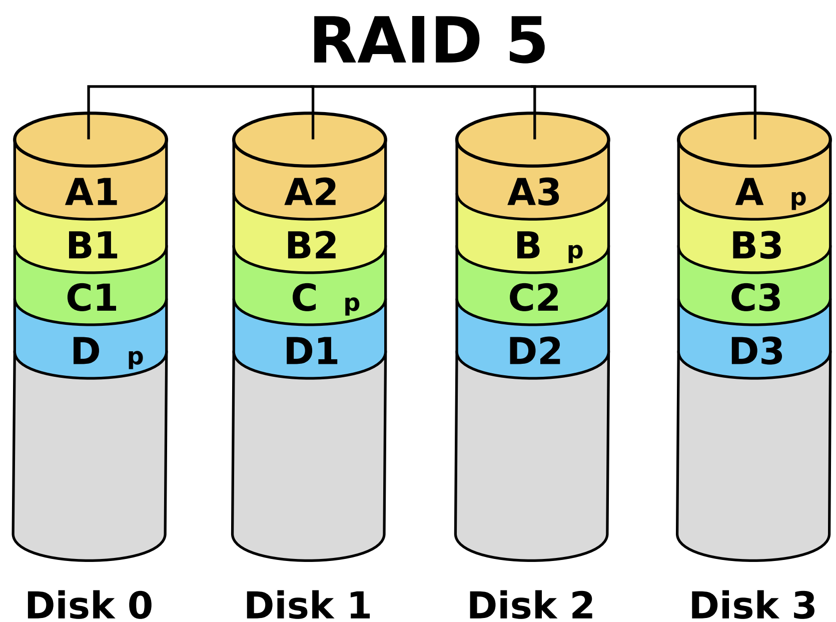


Block-level striping

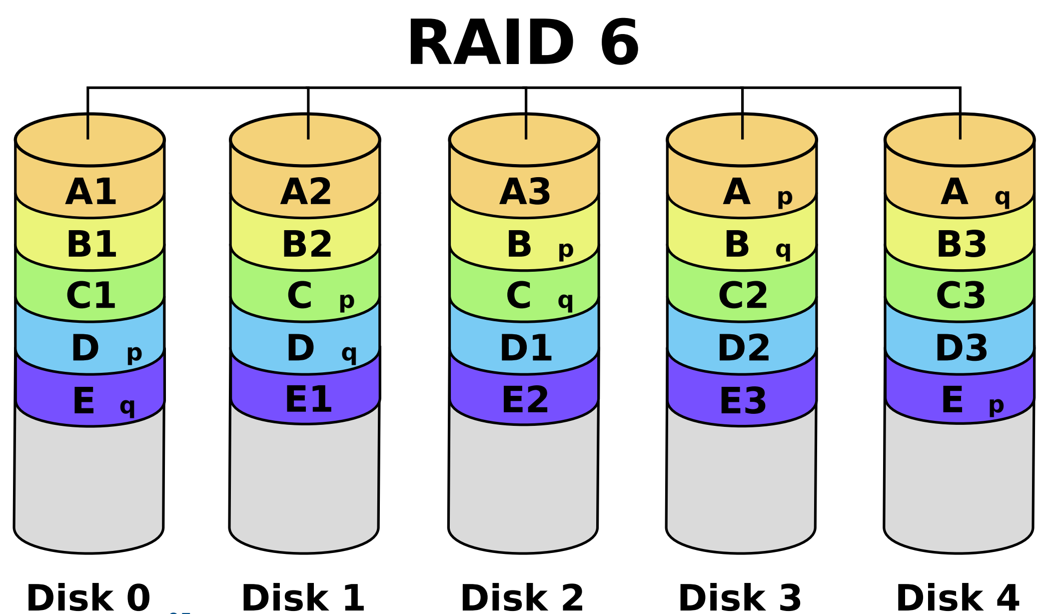
XOR-Parität verteilt

Kapazität: (n - 1) * kl. Disk

- ➕ 1 disk failure
- ➕ Online rebuild
- ➕ Seq. R/W
- ➖ Random W



- Block-level striping
- XOR-Parität 2x verteilt
- Kapazität: (n - 2) * kl. Disk
- \+ 2 disk failures
- \+ Online rebuild
- \+ Seq. R/W
- − Random W

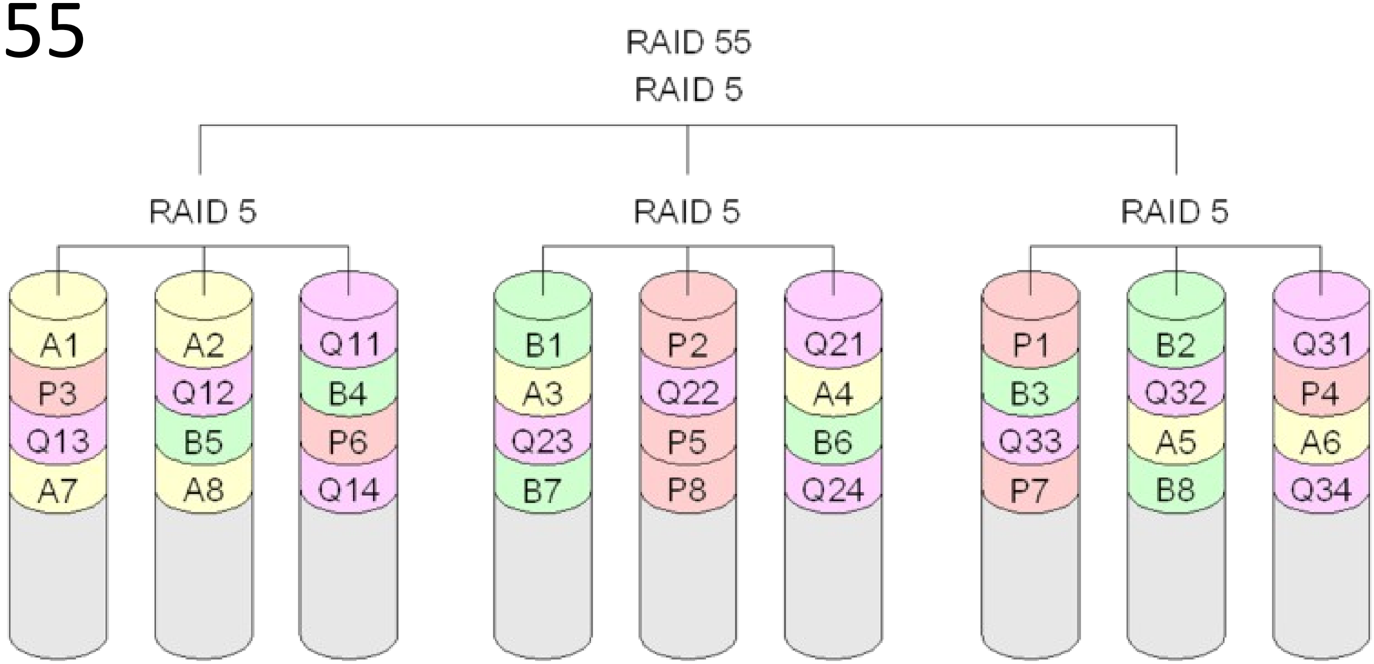


RAID-10 → Striped Mirror

RAID-51 → Mirrored RAID-5

RAID-55

..



RAID-0

$$n = 2$$

$$MTBF = 50000\,h$$

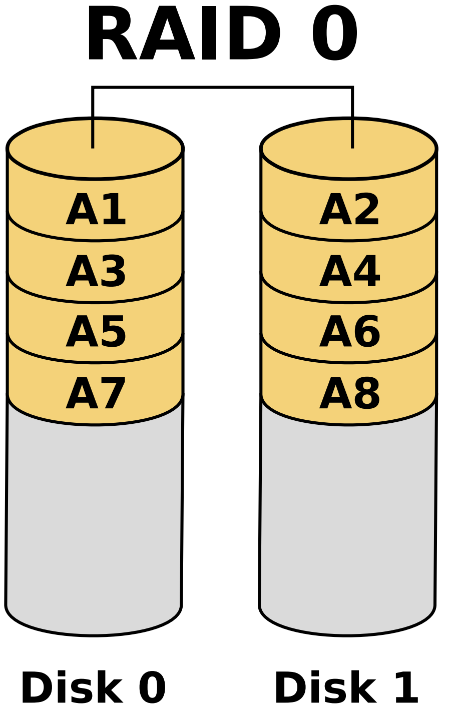
$$MTTDL = MTBF / n$$

$$MTTDL \approx 2{,}85\ \text{years}$$

$$DL_t = t / MTTDL * 100$$

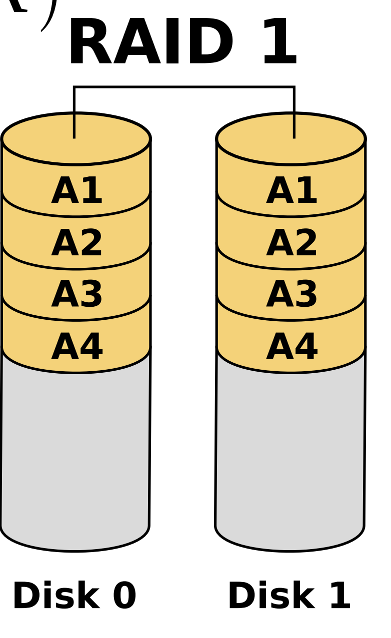
$$DL_2 \approx 70{,}18\,\%$$

$$DL_{10} \geq 100\,\%$$

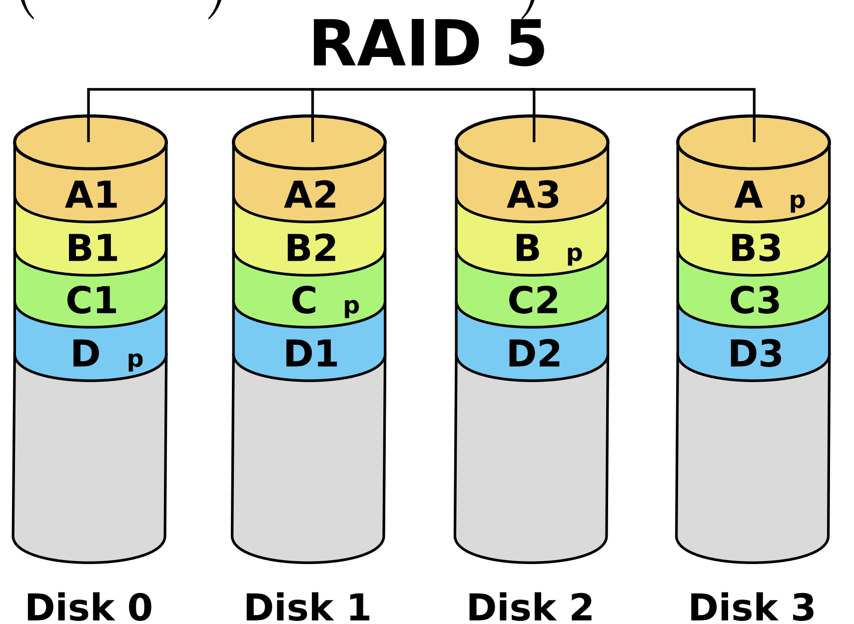


RAID-1

$$n=2$$
$$MTTR=24\,h$$
$$MTBF=50000\,h$$
$$MTTDL=\left(MTBF^2\right)/\left(n*\left(n-1\right)*MTTR\right)$$
$$MTTDL\approx 5945{,}59\text{ years}$$
$$DL_t=t\,/\,MTTDL*100$$
$$DL_2\approx 0{,}037\,\%$$
$$DL_{10}\approx 0{,}168\,\%$$



$$n = 5$$
$$MTTR = 24\,h$$
$$MTBF = 50000\,h$$
$$MTTDL = (MTBF^2)/(n*(n-1)*MTTR)$$
$$MTTDL \approx 594{,}56 \text{ years}$$
$$DL_t = t/MTTDL*100$$
$$DL_2 \approx 0{,}336\,\%$$
$$DL_{10} \approx 1{,}682\,\%$$

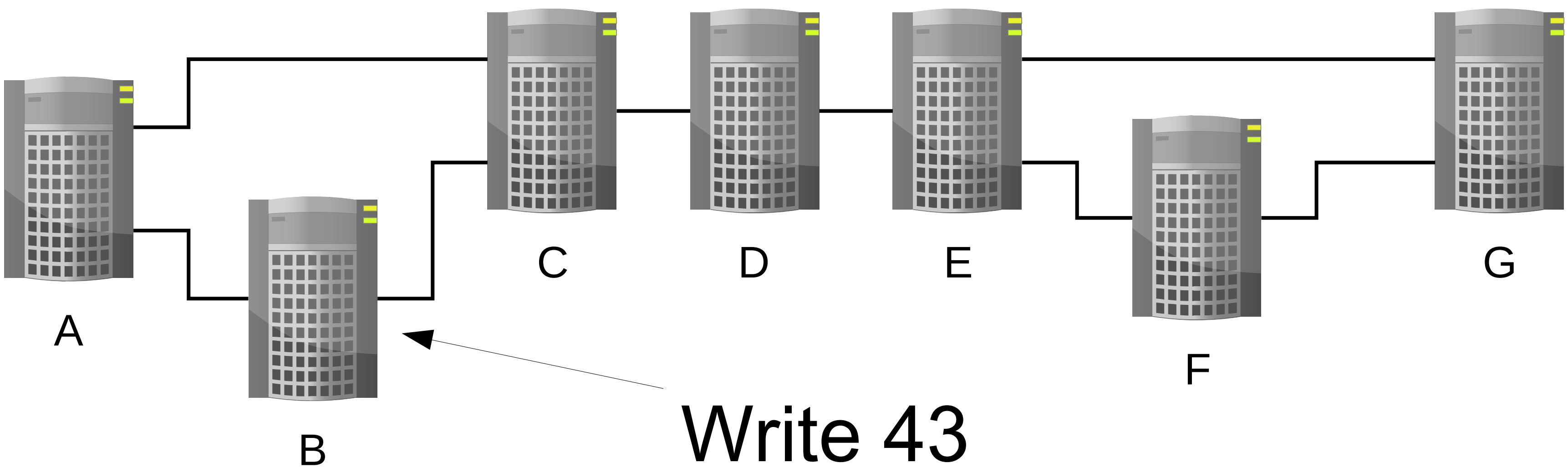


Westfälische Hochschule

VII
Data replication

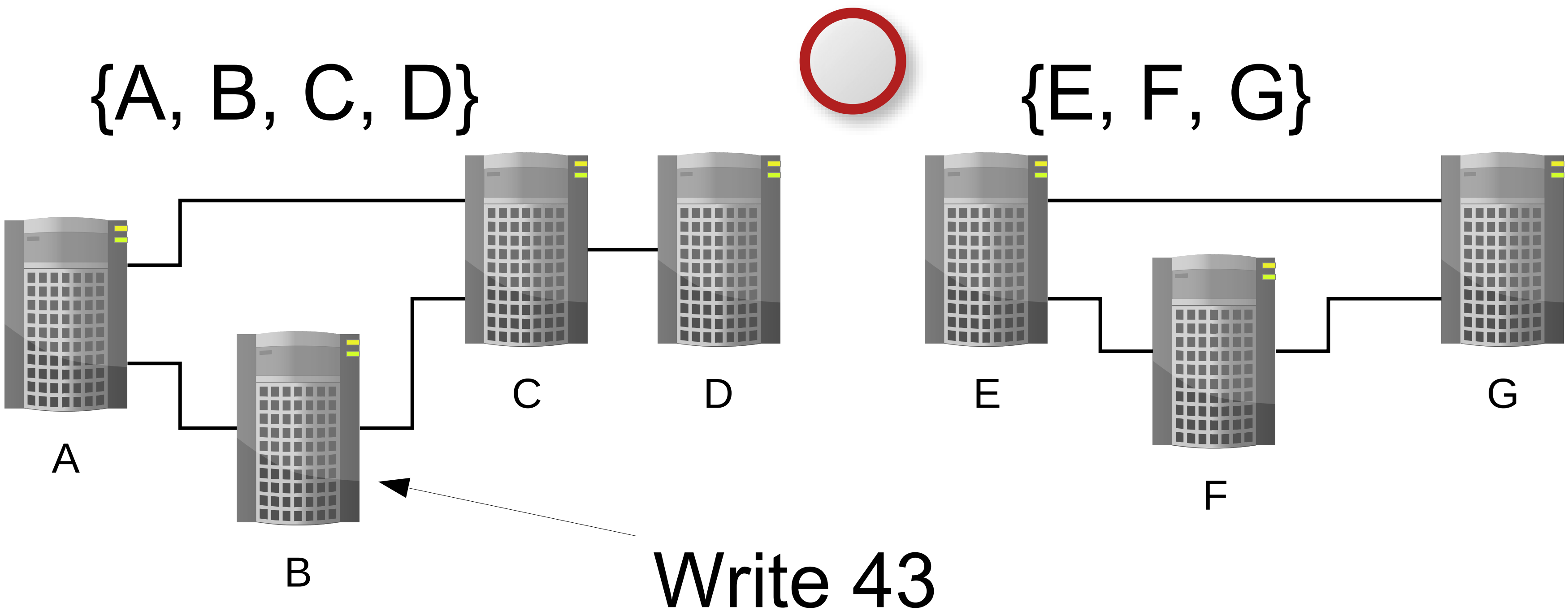
VI Data replication Quorum Consensus

{A, B, C, D, E, F, G}



A	B	C	D	E	F	G
42	42	42	42	42	42	42

VI Data replication Quorum Consensus



A	B	C	D	E	F	G
42	42	42	42	42	42	42

Quorum Consensus → r,
w

$$w > v/2$$
$$r + w > v$$

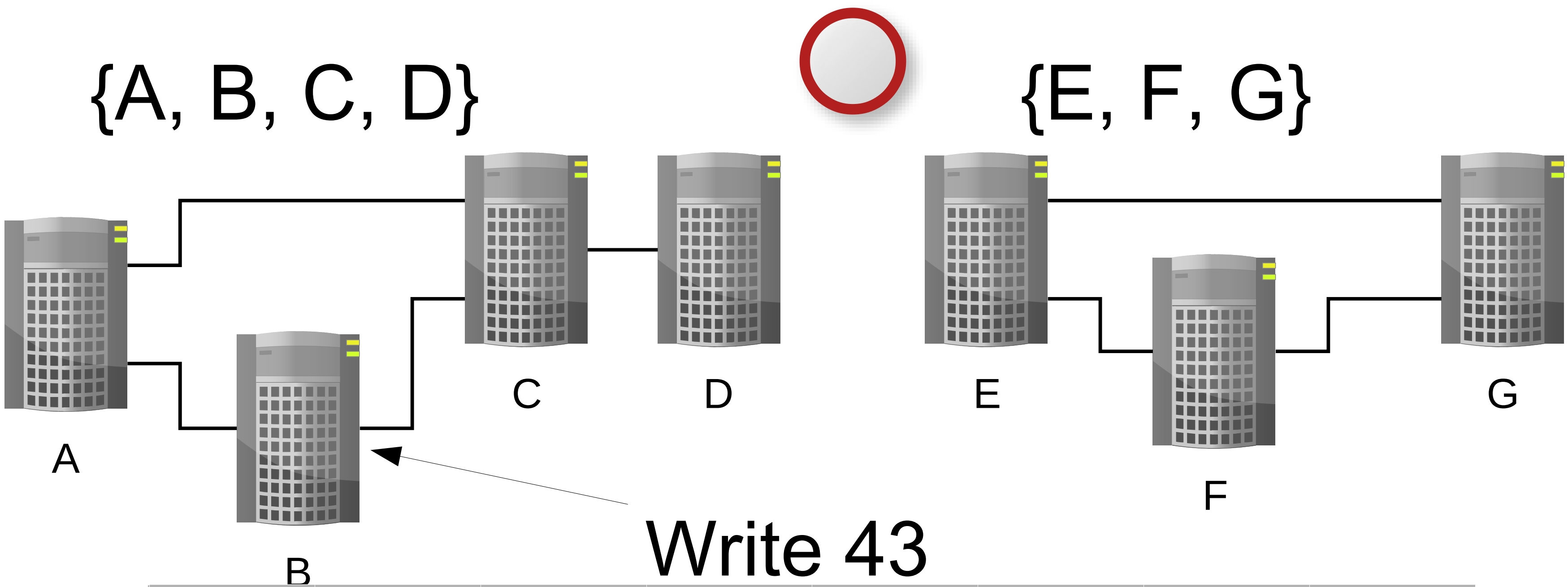
V = 7 #Nodes

w → 4

r → 4

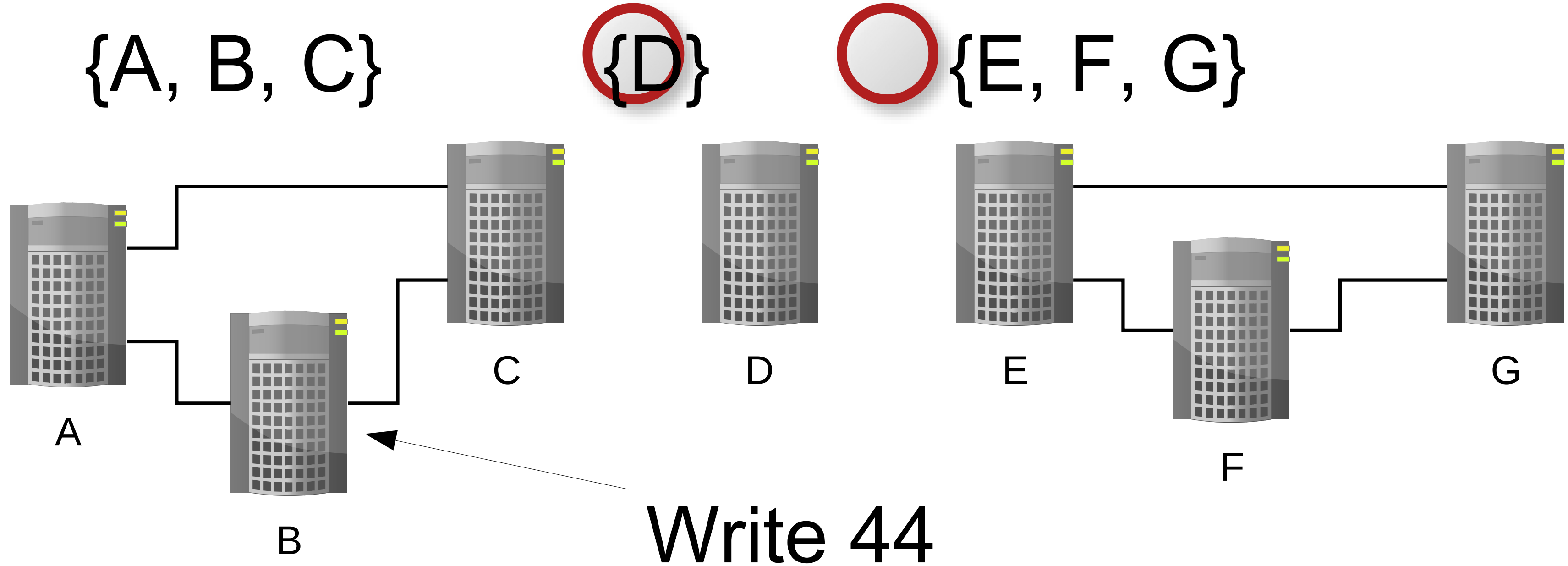
A: 1
B: 1
C: 1
D: 1
E: 1
F: 1
G: 1

VI Data replication Quorum Consensus



	A	B	C	D	E	F	G
x	42	42	42	42	42	42	42
v	7	7	7	7	7	7	7

Quorum Consensus



	A	B	C	D	E	F	G
x	43	43	43	43	42	42	42
v	4	4	4	4	7	7	7

VI Data replication Majority Consensus

Problem:

- r-Quorum, große Cluster

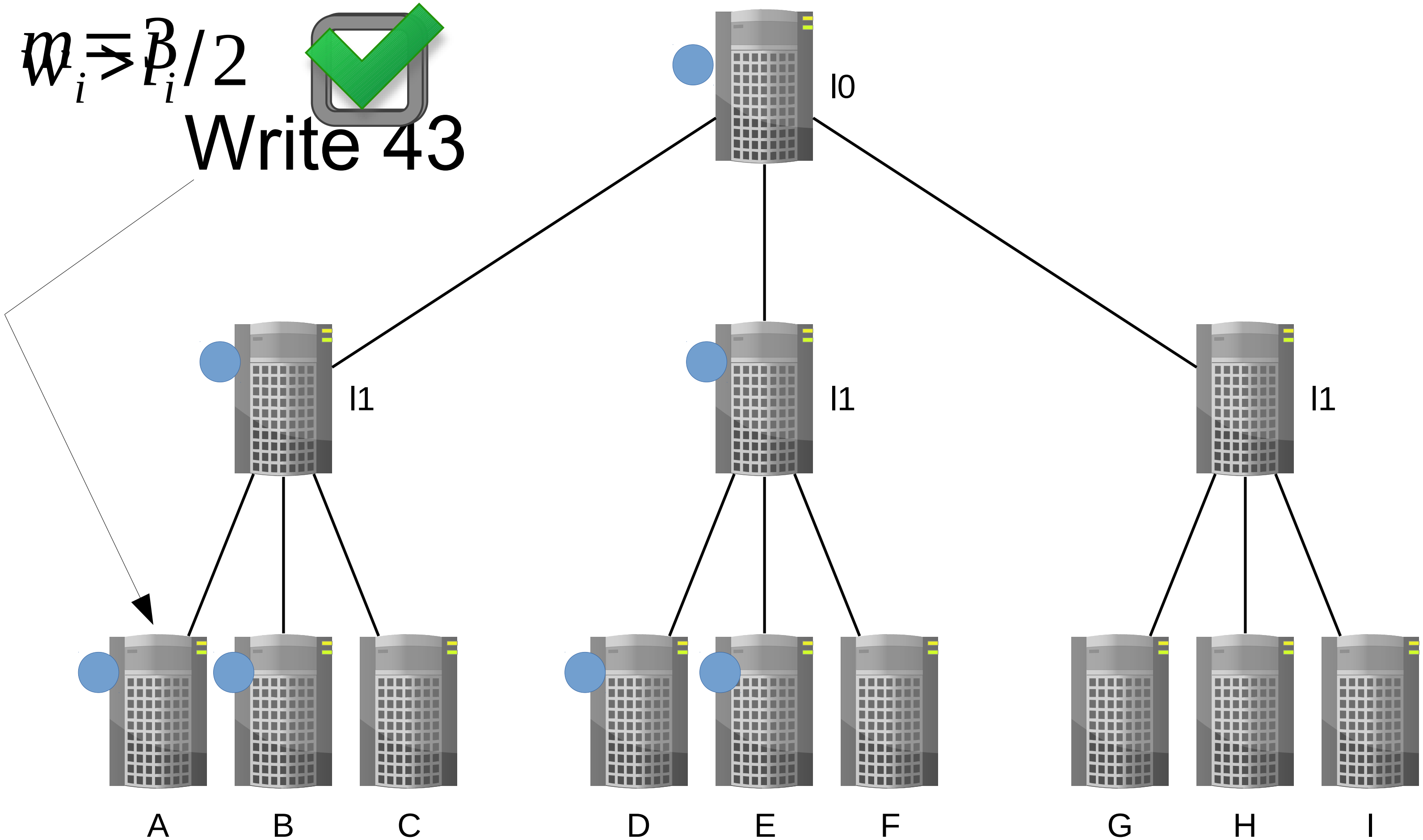
Lösung:

- Majority Consensus

VI Data replication   Majority Consensus

$m = 3$

$W_i > L_i/2$



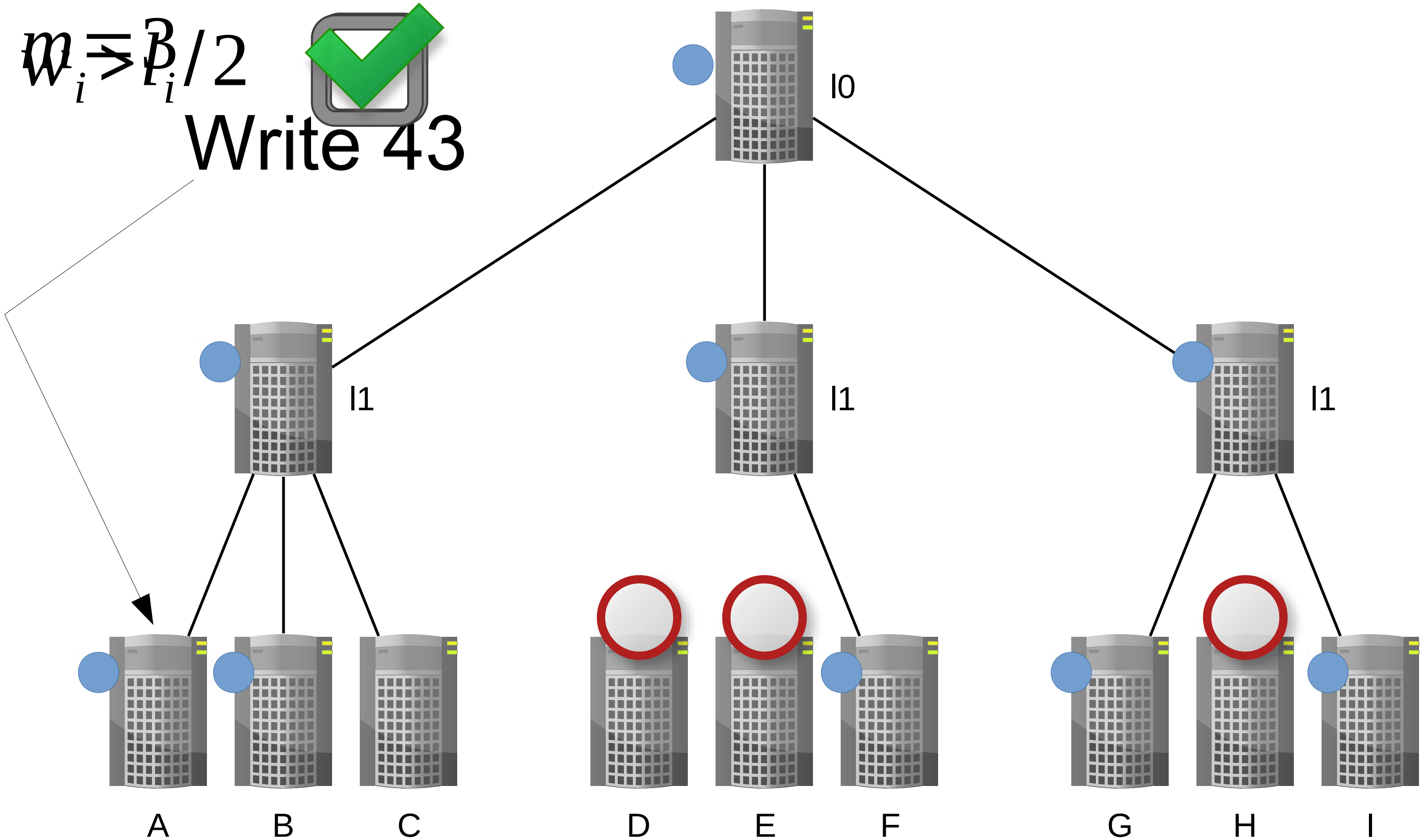
Westfälische Hochschule

VI Data replication   Majority Consensus

$m = 3$

$W_i > L_i/2$

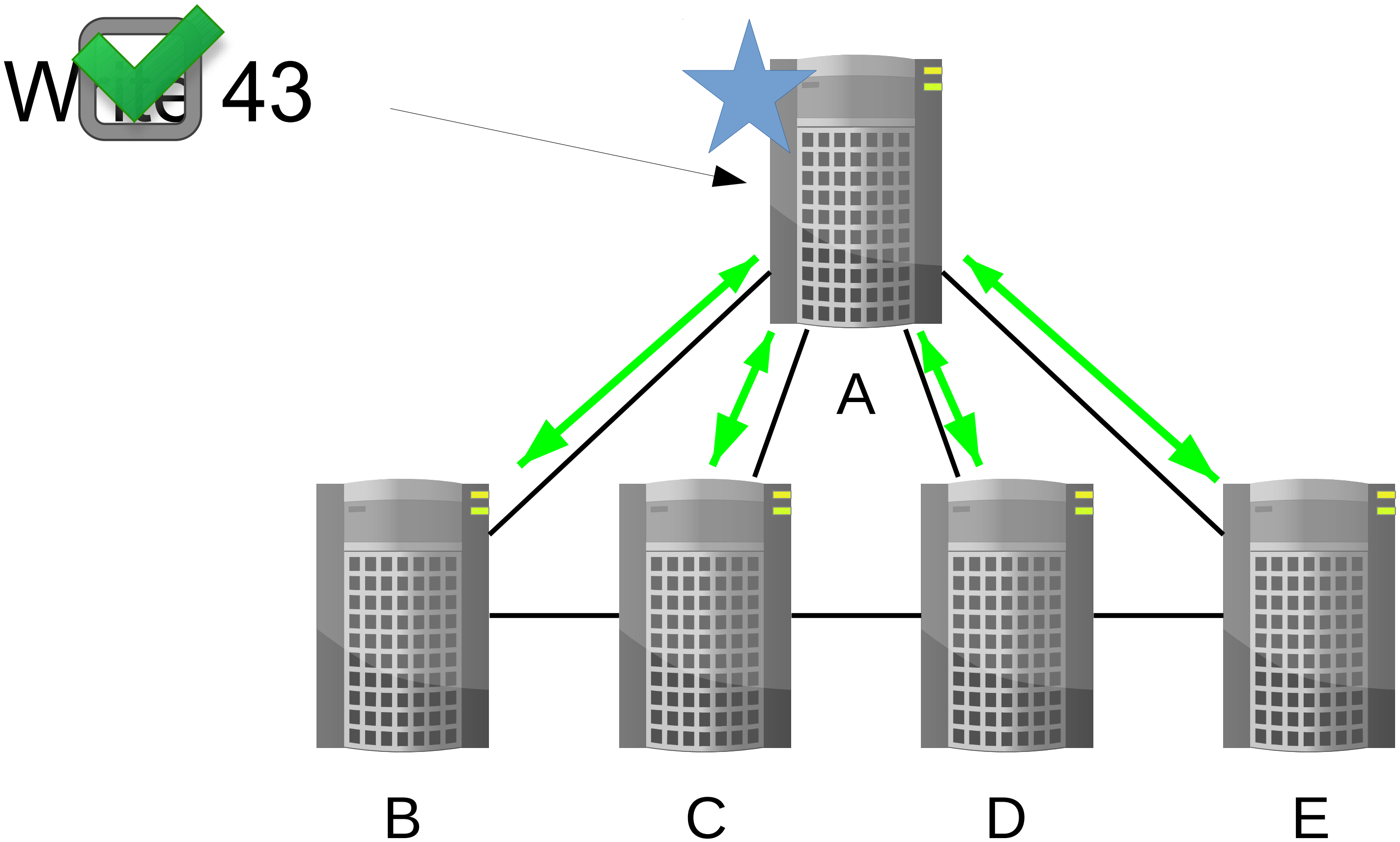
Write 43



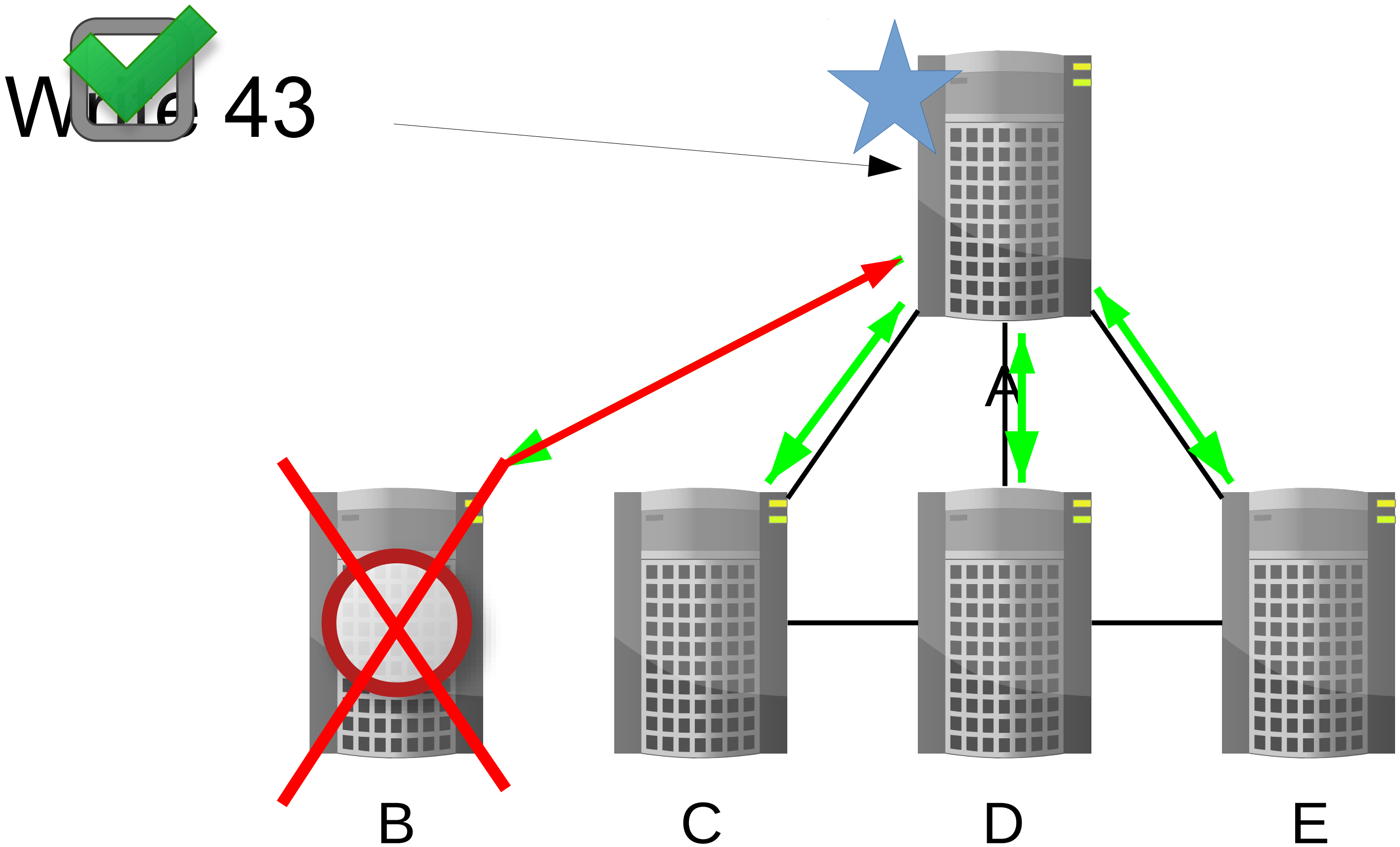
Westfälische
Hochschule

VIII
Primary Backup Approach

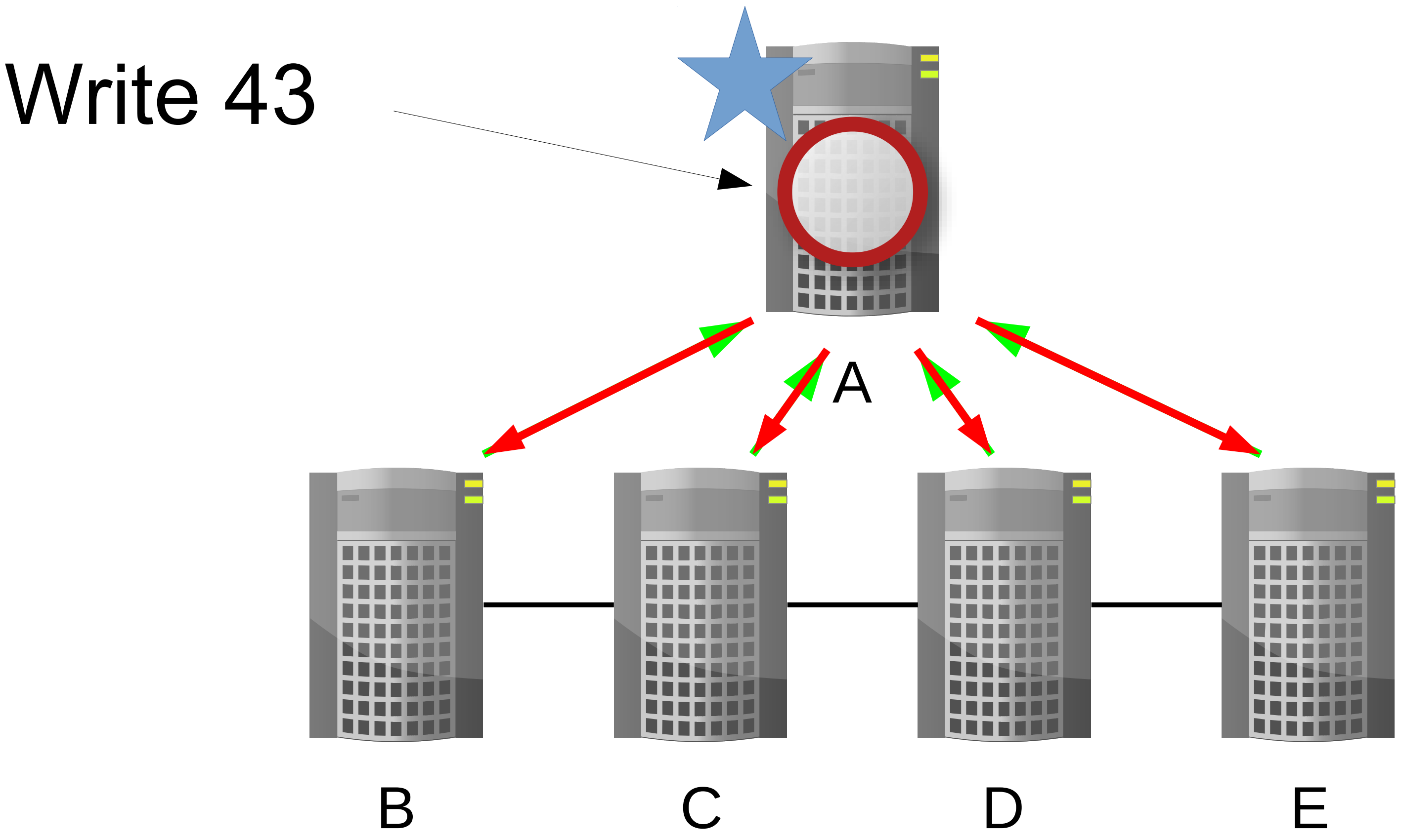
VI Primary Backup Approach



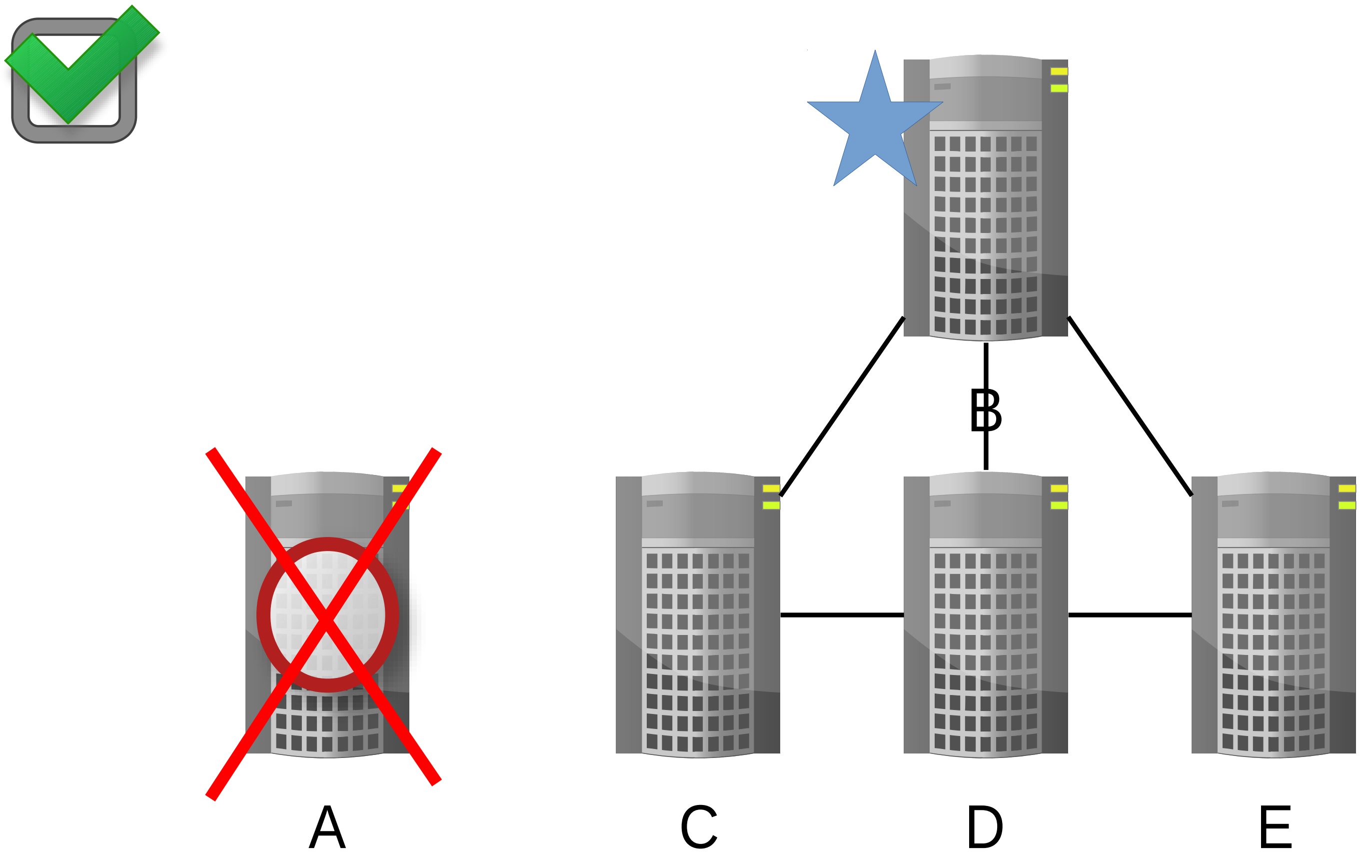
VI Primary Backup Approach



VI Primary Backup Approach



VI Primary Backup Approach



Westfälische Hochschule

IX
Algorithm-based fault tolerance

VII ABFT Matrix Arithmetic

1	0	1	0
0	1	0	1
0	1	0	0
1	0	0	1

A

$A * B \neq$

0	1	1	2
2	0	0	1
1	0	0	0
1	0	0	2

0	1
0	0
1	1
0	1

B

Matrix Arithmetic

1	0	1	0
0	1	0	1
0	1	0	0
1	0	0	1
3	2	1	2

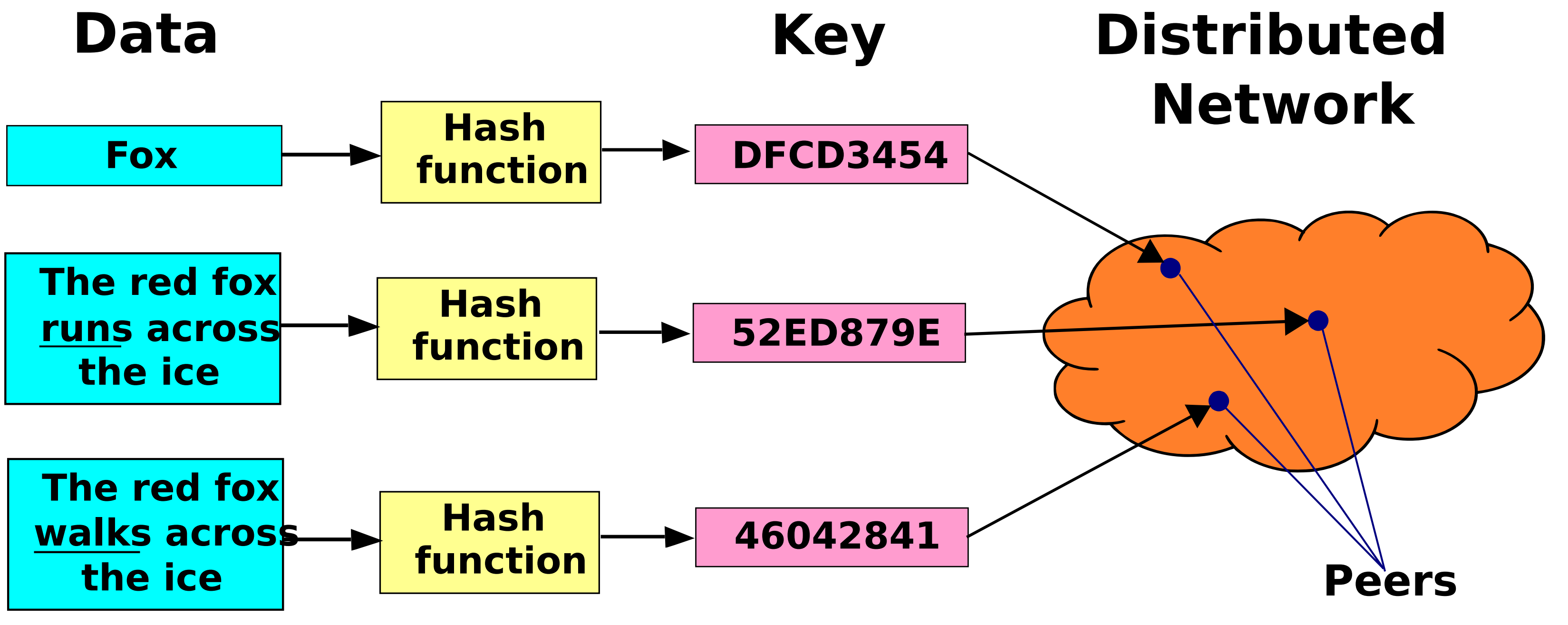
$A * B \neq$

A

0	0	0	1	1

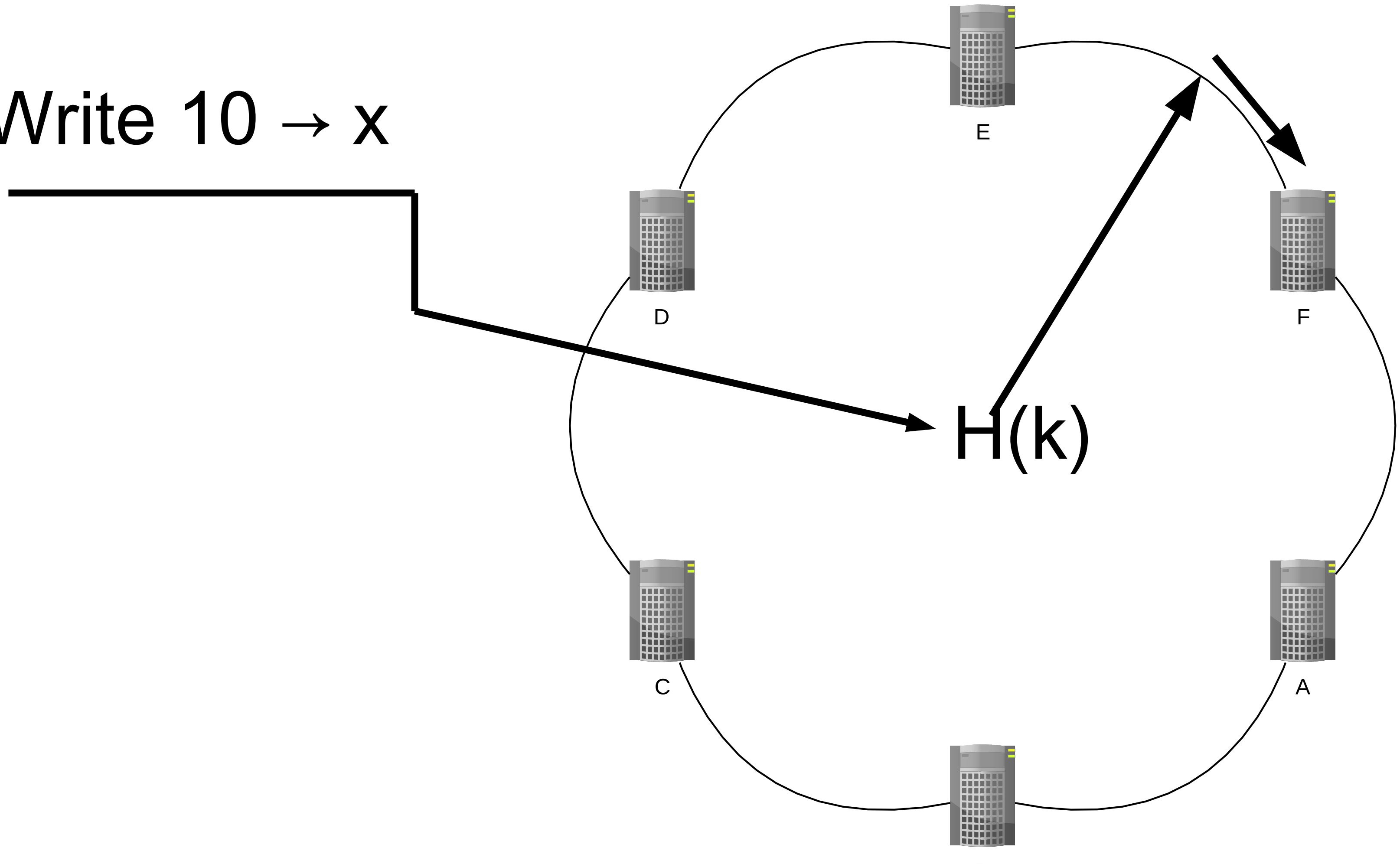
0	1	1	2	4	1
2	0	0	1	4	3
1	0	0	0	1	2
1	0	0	2	3	
4	1	1	6	12	

Distributed Hash Table Problem

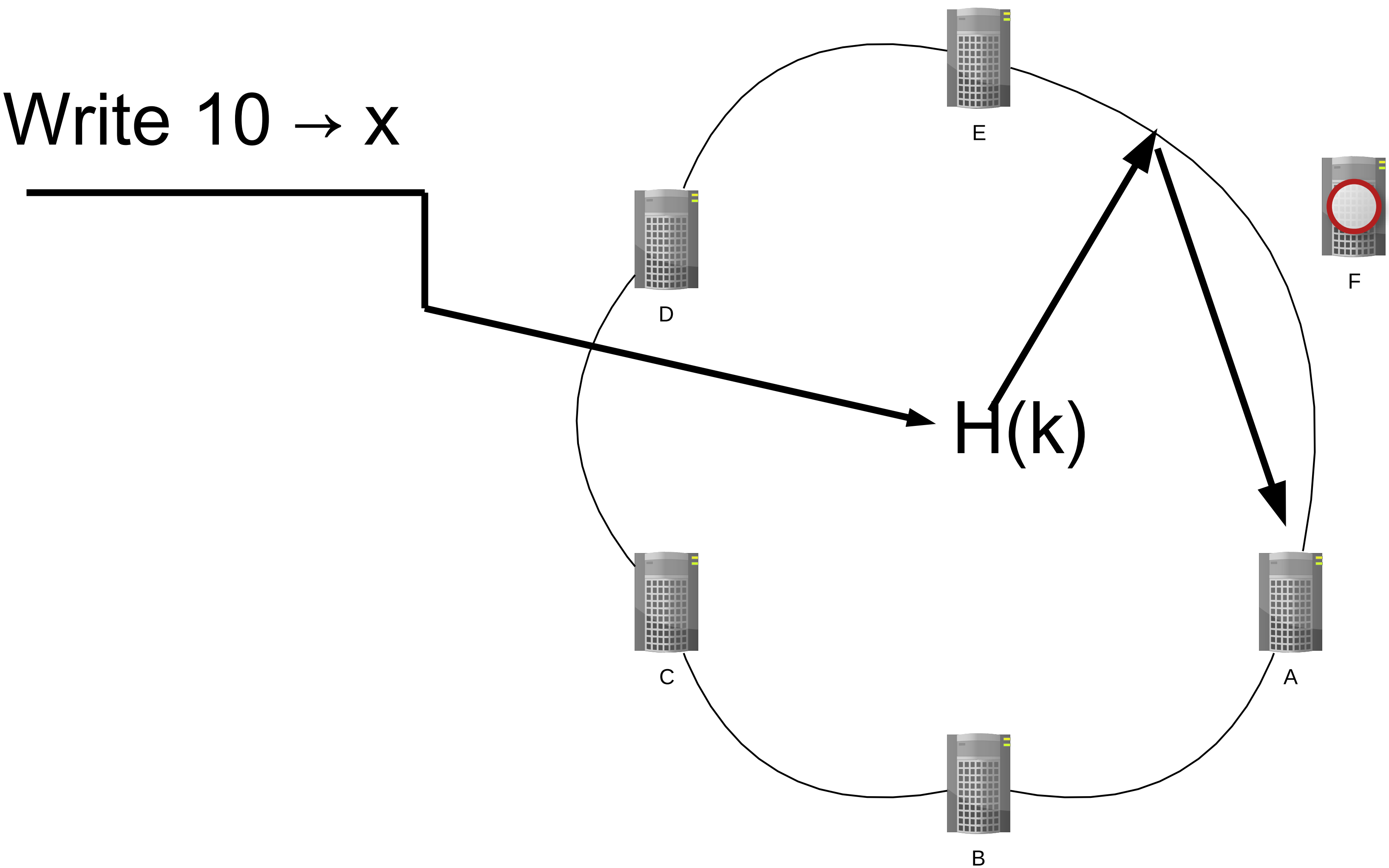


Consistent Hashing

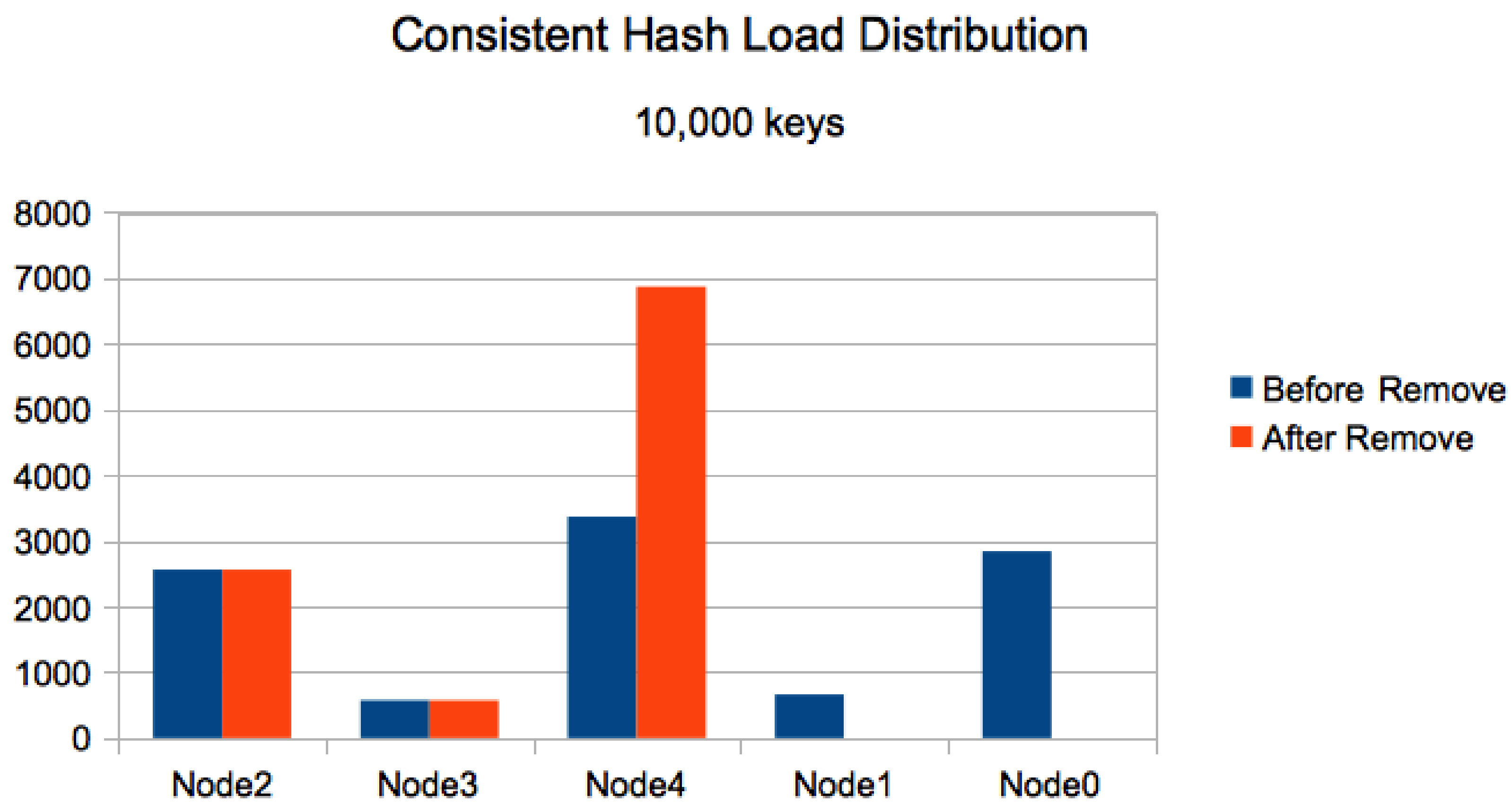
Write 10 → x



Consistent Hashing

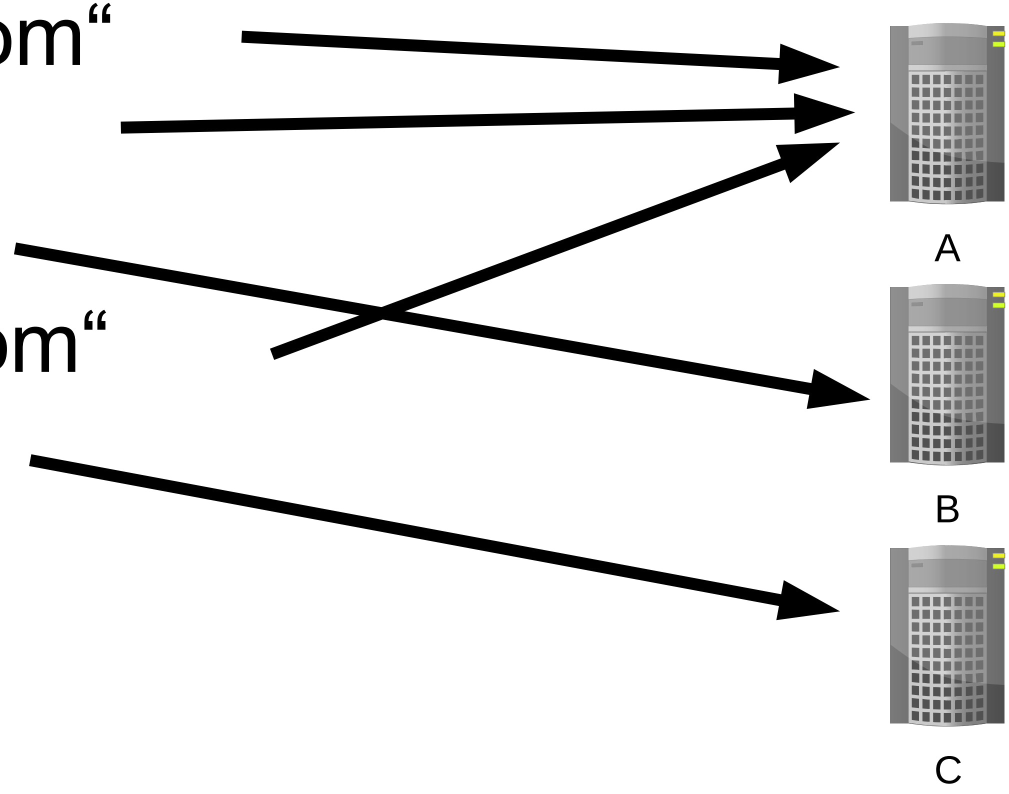


Consistent Hashing



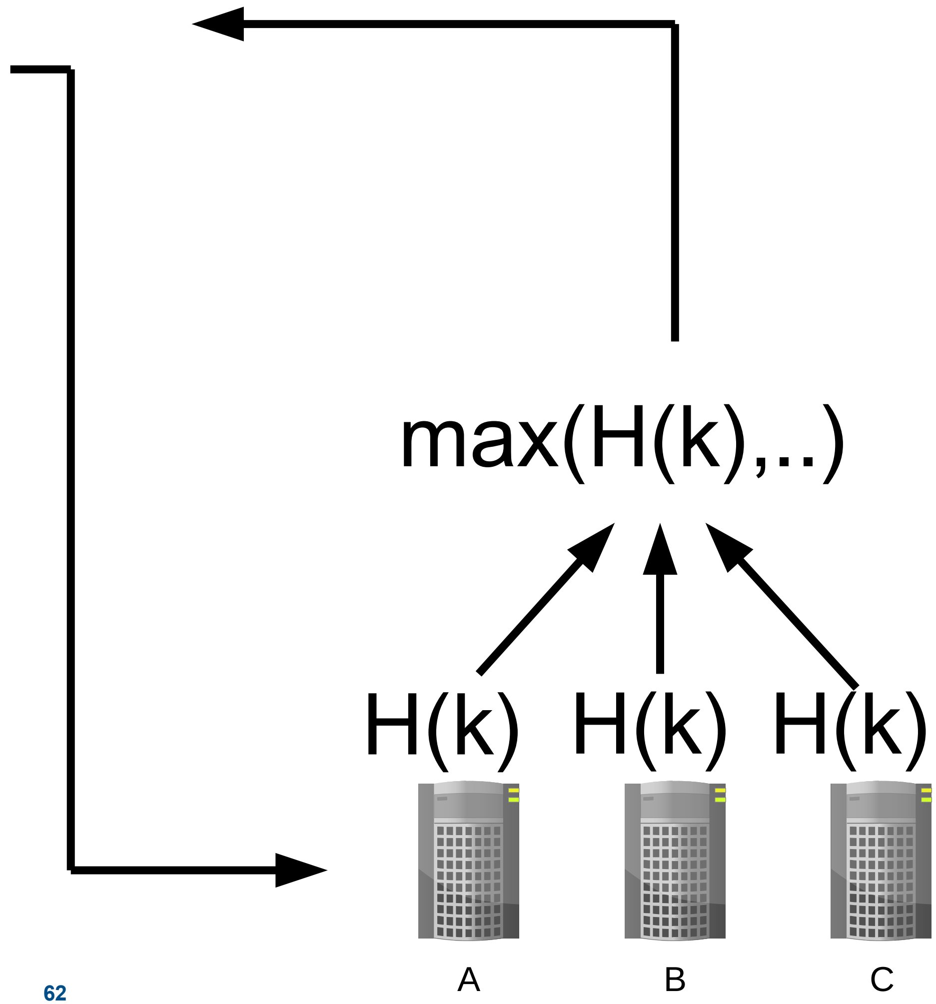
Rendezvous Hashing (HRW)

„www.google.com“
„www.ebay.de“
„heise.de“
„www.paypal.com“
„blog.fefe.de“

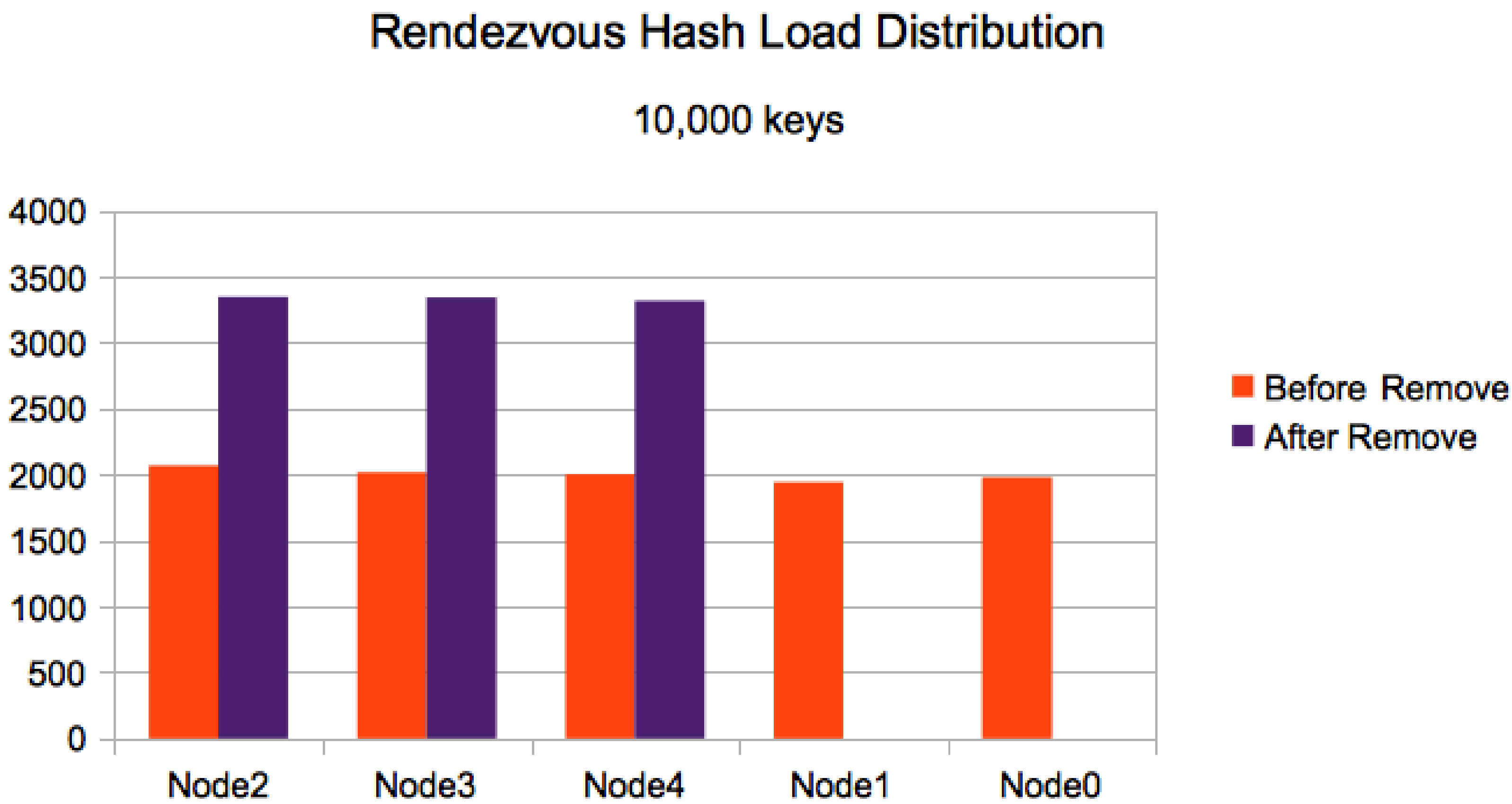


Rendezvous Hashing (HRW)

„www.google.com“
„www.ebay.de“
„heise.de“
„www.paypal.com“
„blog.fefe.de“



VII ABFT Rendezvous Hashing (HRW)



Westfälische Hochschule

Vielen Dank

Noch Fragen?