EFFICIENT CLOUD STORAGE

ywu@cs.hku.hk

Large scale data centers need large volumes of storage...

gets cheaper
($ per byte)

more expensive
(percentage)

Existing techniques try to spin down disks when idle...

"Idle" => "standby"

Δt is the threshold

Power dissipation (4-disc values shown)	Avg (watts 25° C)
Spinup	—
Idle*	9.30
Idle* (with offline activity)	10.40
Operating (40% r/w, 40% seek, 20% inop.)	13.00
Seeking (random, 20% idle)	12.60
Standby	0.80
Sleep	0.80

10 X

[1] http://www.seagate.com/support/disc/manuals/sata/100402371a.pdf

$$\Delta t = \frac{E_{up}+E_{down}}{P_I}$$

E_{up}	spin up energy
E_{down}	spin down energy
P_I	idleness energy

is the optimal deterministic power management policy [2] **with a competitive ratio of 2**.

[2] S. Irani, G. Singh, S. K. Shukla, and R. K. Gupta. An overview of the competitive and adversarial approaches to designing dynamic power management strategies. *IEEE Trans. VLSI Syst.*, 13(12):1349–1361, 2005.

However...

- Energy overheads
- Thrashing latencies
- Application traces dependent
- Hardware failures
- ......

Then what?

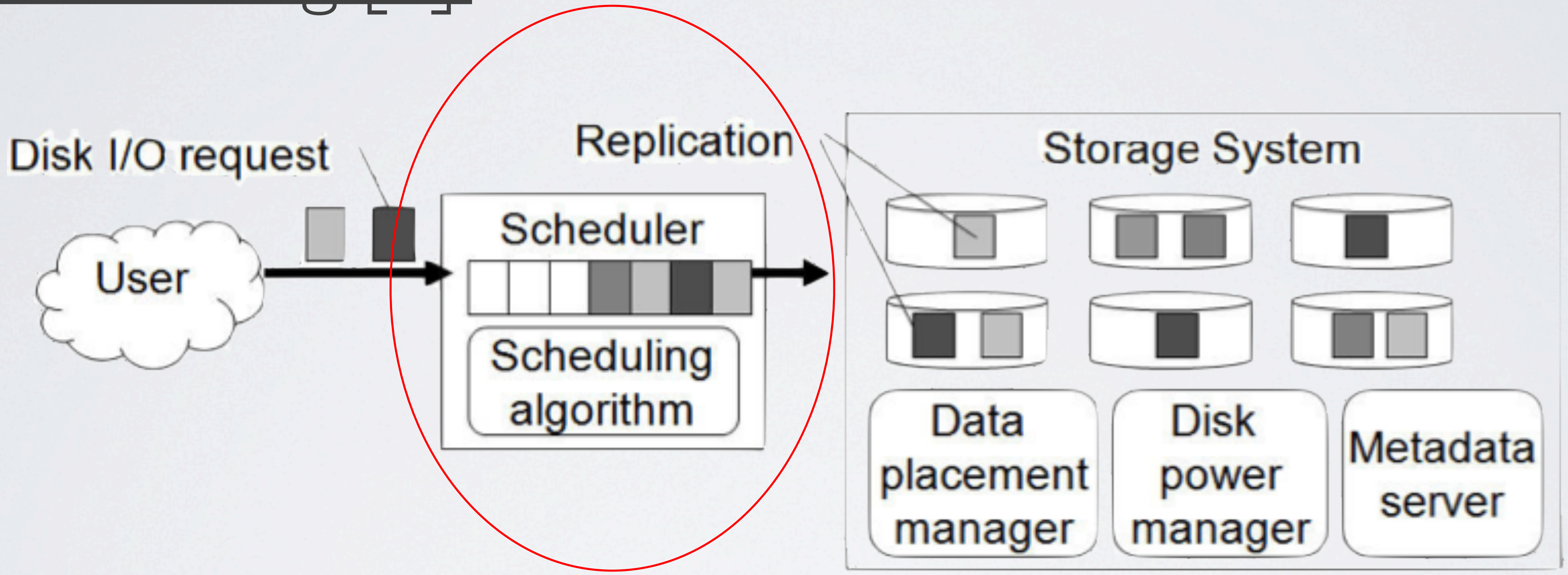
Traffic reshaping: redirect the workload to a small subset of the disks, allowing the other disks to enjoy long periods of inactivity.

Depends on data placement/migrations...

Performance Concerns!

From another perspective?

Scheduling [3]:



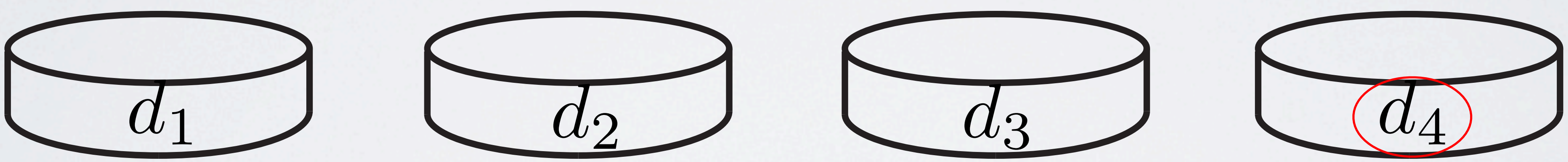
Energy aware!

[3] Jerry Chou, Jinoh Kim, Doron Rotem: Energy-Aware Scheduling in Disk Storage Systems. ICDCS 2011: 423-433

Let's see some examples...

Request

$r_1 : b_1$	$r_2 : b_2$	$r_3 : b_3$	$r_4 : b_4$	$r_5 : b_5$	$r_6 : b_6$



Disk

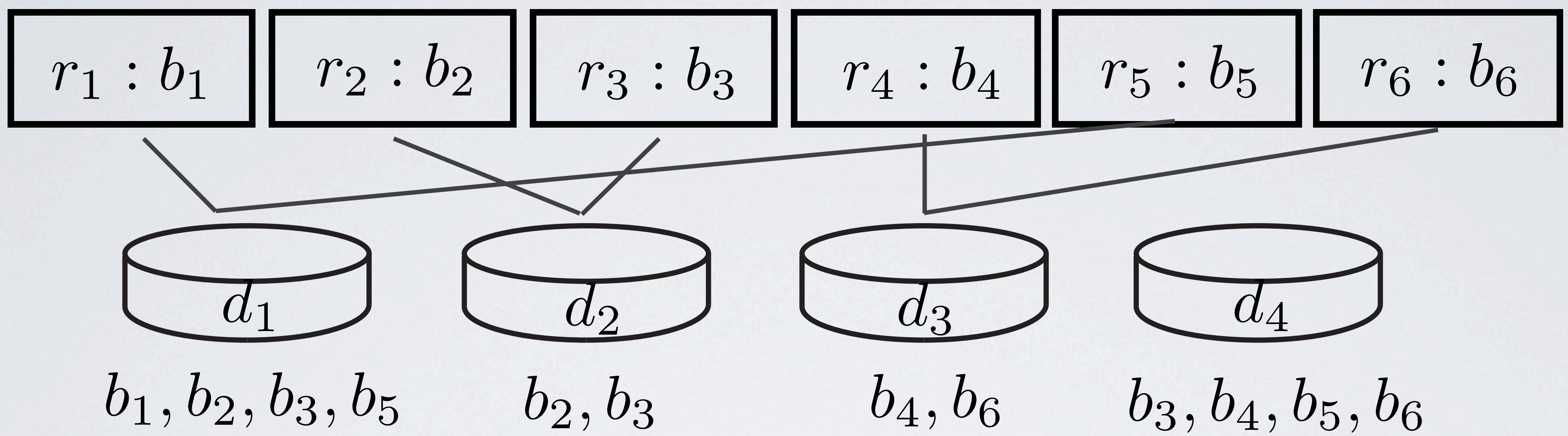
b_1, b_2, b_3, b_5	b_2, b_3	b_4, b_6	b_3, b_4, b_5, b_6

Block

$\Delta t = 5$ seconds

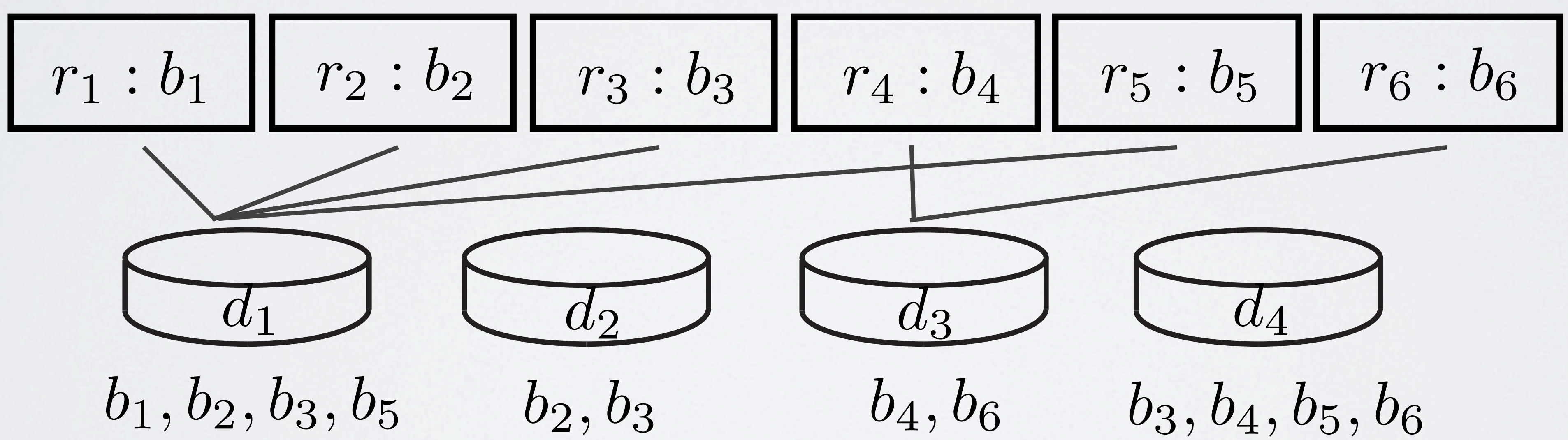
In Batch Mode:

A:



energy = 15

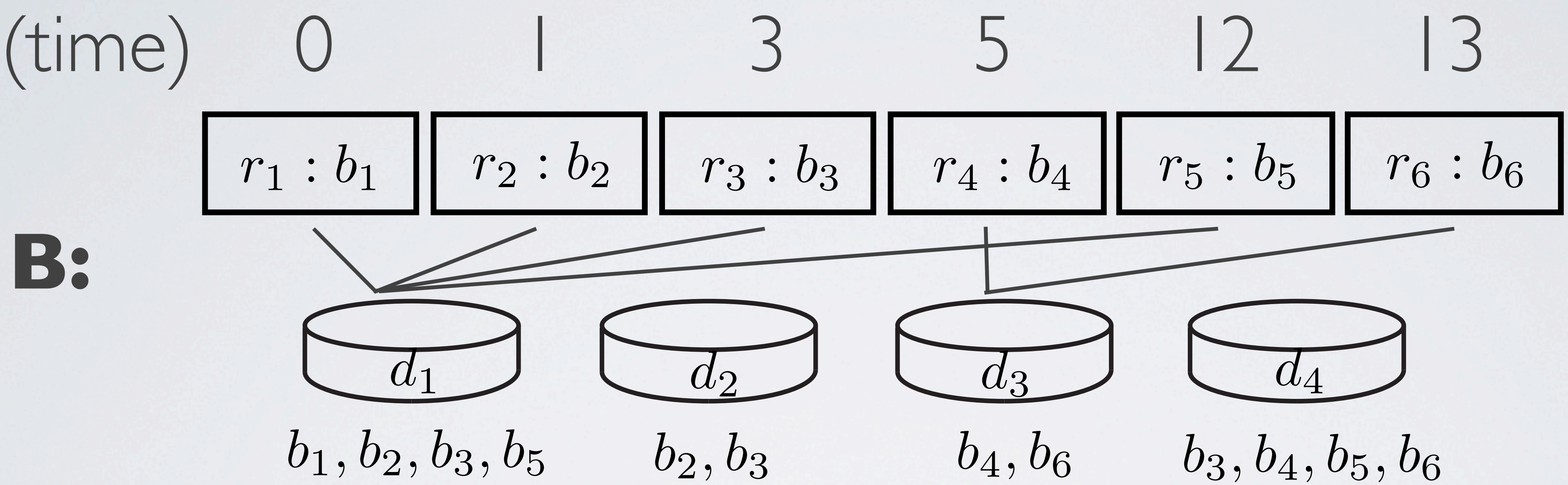
B:



energy = 10

B is optimal.

In Realtime Mode:



energy = 23, not optimal.

How to find the optimal scheduling?

An energy-aware scheduling problem (ES) is described by an input of **(R, D, L, P)**

- **R** is the request stream $r_1, r_2, ...$
- **D** is the disk arrays $d_1, d_2, \ldots$
- **L** is data placement state
- **P** is the system configuration

$$T_{up}, T_{down}, E_{up}, E_{down}, T_B, P_I$$

T_{up}	spin up latency
T_{down}	spin down latency
T_B	threshold
E_{up}	spin up energy
E_{down}	spin down energy
P_I	idleness energy

Let S_{ES} be feasible schedule set,

We are trying to find an optimal schedule

$$S^*_{ES} = min(S^x_{ES} | \forall S^x_{ES} \in S_{ES})$$

T_{up}	spin up latency
T_{down}	spin down latency
T_B	threshold
E_{up}	spin up energy
E_{down}	spin down energy
P_I	idleness energy

$$S_{ES}^{*} = min(S_{ES}^{x} | \forall S_{ES}^{x} \in S_{ES})$$

Let $X(\cdot)$ be a function of energy saving.

$X(S_{ES}^{x}, r_i)$
denotes the energy saving of request

r_i , which comes at t_i

Assume the next request r_j at the same disk comes at t_j

The energy consumed by request r_i is

$$(t_j - t_i) \times P_I \quad \text{if } 0 \leq t_j - t_i < T_B + T_{up} + T_{down}$$

$X(S_{ES}^{x}, r_i)$

$$= X(i, j, k) = E_{up} + E_{down} + T_B \times P_I - (t_j - t_i) \times P_I$$
$$= E_{up} + E_{down} + (T_B - t_j + t_i) \times P_I$$

if $0 \leq t_j - t_i < T_B + T_{up} + T_{down}$,

r_i is at disk k, and its successor request at disk k is r_j

$= 0$

otherwise

$$X(S_{ES}^{x}) = \sum_{\forall r_i \in R} X(S_{ES}^{x}, r_i)$$

The scheduling optimization problem:

Given a scheduling problem $\mathrm{ES}(R, D, L, P)$
and a set of energy saving $\mathcal{U}$:
$$\mathcal{U} = \{X(i,j,k) | \forall X(i,j,k) \in \mathrm{ES}\}$$
Find a subset of energy saving $\mathcal{S} \subseteq \mathcal{U}$
S.t $X(S^*_{ES}) = \sum_{\forall X(i,j,k) \in \mathcal{S}} X(i,j,k)$ is maximized
subject to *//scheduling constraints*
For each pair of $X(i,j,k) \in \mathcal{S}$ and $X(i',j',k') \in \mathcal{S}$:
$i \neq i', j \neq j'$
$k == k'$ if $\{i,j\} \cap \{i',j'\} \neq \emptyset$

How to understand the constraints?

Reducible to Weighted Independent Set problem

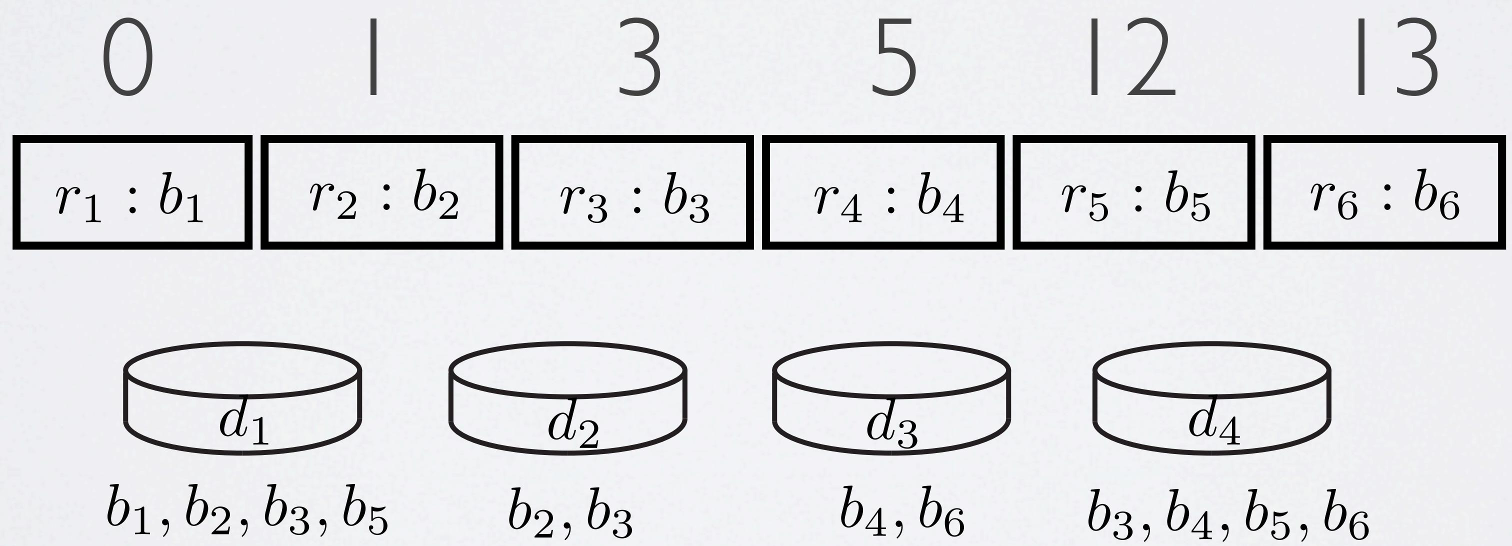
NP-Complete

Algorithm:

Phase 1: Construct a graph $G = (V, E)$

(1) Add a node to V for each $X(i,j,k) \in ES$ with non-zero value

$X(i,j,k) \in ES \Leftrightarrow$ d_k has the data for r_i, r_j



$\boxed{X(1,2,1) = 4}$ $\boxed{X(2,3,1) = 3}$ $\boxed{X(1,3,1) = 2}$

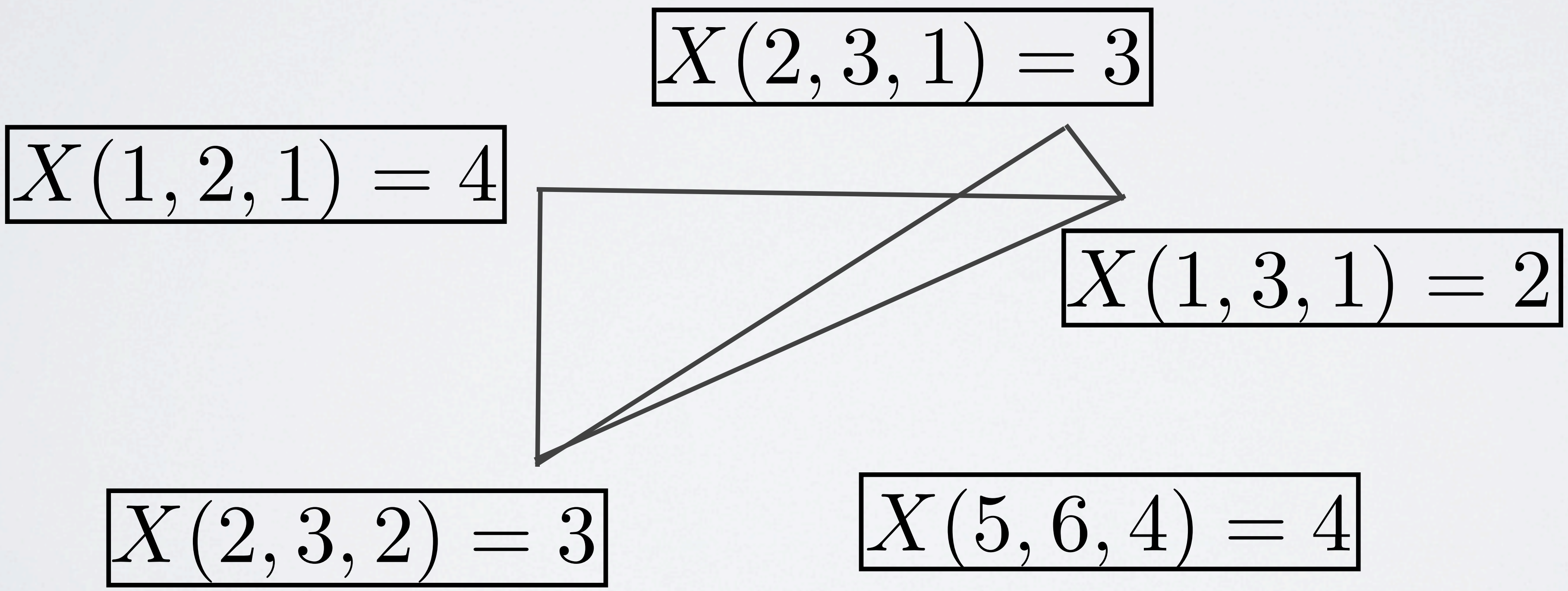
$\boxed{X(2,3,2) = 3}$ $\boxed{X(5,6,4) = 4}$

Given a scheduling problem $\mathrm{ES}(R, D, L, P)$
and a set of energy saving $\mathcal{U}$:
$\mathcal{U} = \{X(i,j,k) \mid \forall X(i,j,k) \in \mathrm{ES}\}$
Find a subset of energy saving $\mathcal{S} \subseteq \mathcal{U}$
S.t $X(S^*_{ES}) = \sum_{\forall X(i,j,k) \in \mathcal{S}} X(i,j,k)$ is maximized
subject to *//scheduling constraints*
For each pair of $X(i,j,k) \in \mathcal{S}$ and $X(i',j',k') \in \mathcal{S}$:
$i \neq i'$, $j \neq j'$
$k == k'$ if $\{i,j\} \cap \{i',j'\} \neq \emptyset$

Algorithm:

Phase 1: Construct a graph $G = (V, E)$

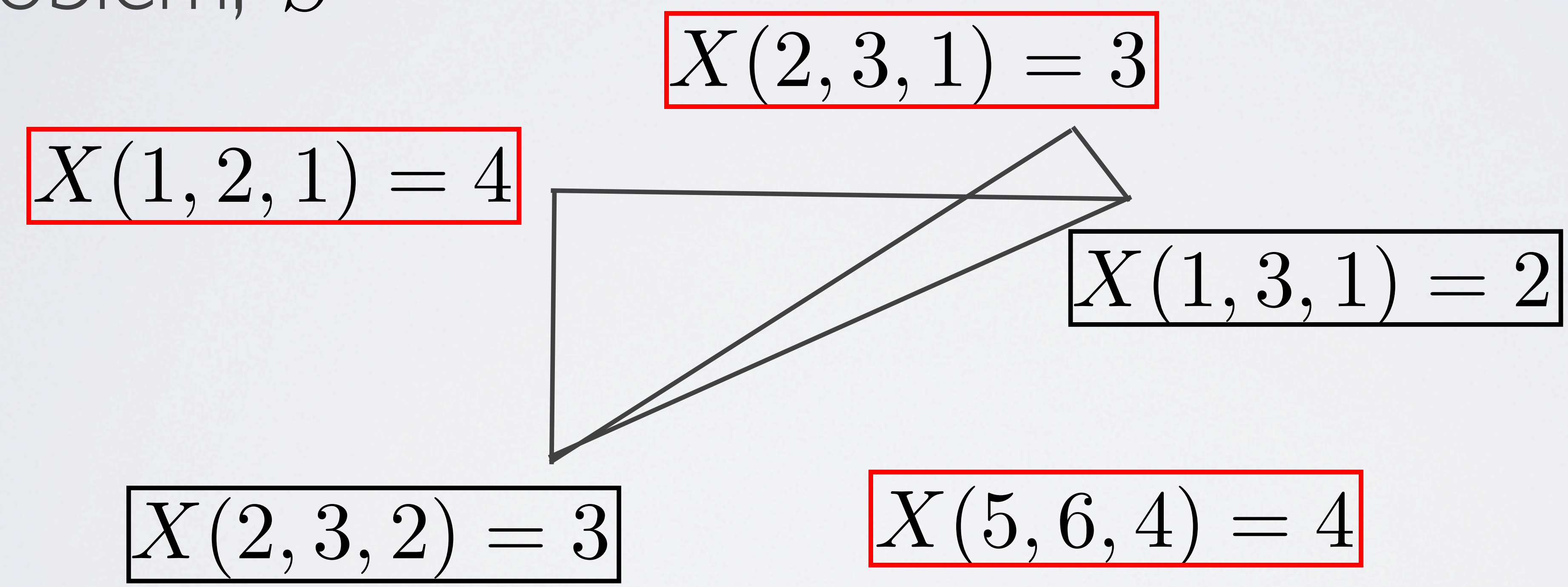
(2) Add edges between any pair of nodes that do not satisfy the constraints



Algorithm:

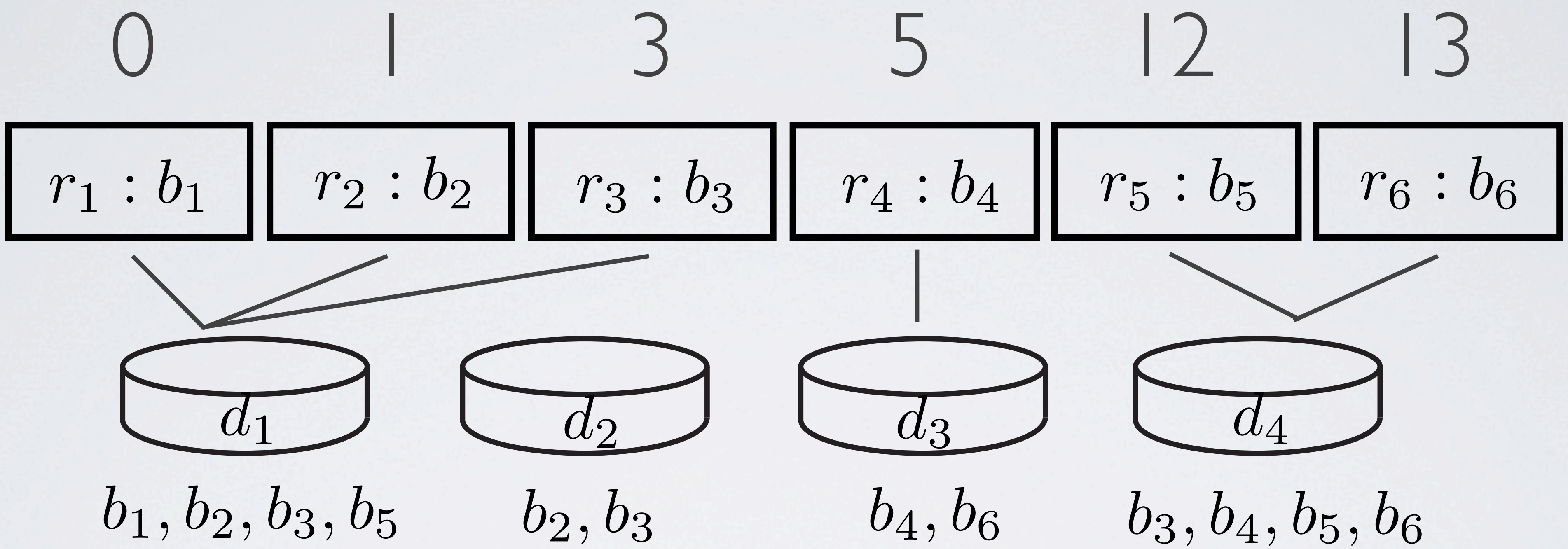
Phase 2: derive the schedule from $G = (V, E)$

(3) maximum weighted independent set problem, S



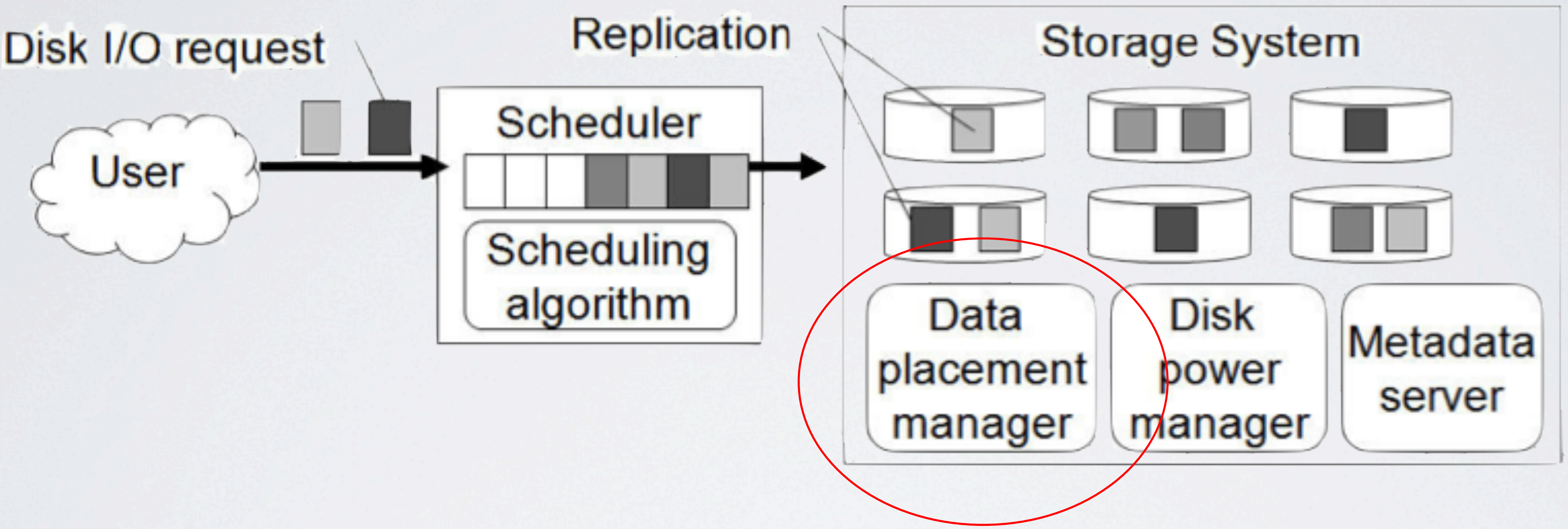
(4) schedule requests r_i, r_j to disk d_k if $X(i, j, k) \in S$

Algorithm:



$\boxed{X(1,2,1) = 4}$ $\boxed{X(2,3,1) = 3}$ $\boxed{X(5,6,4) = 4}$

Let's look back at...



How does Hadoop handle it?

When data set grows, it's necessary for a distributed storage file system.

- Network based
- Commodity hardware
- Tolerate to node failures

Hadoop's filesystem...

HDFS = (Hadoop Distributed File System)

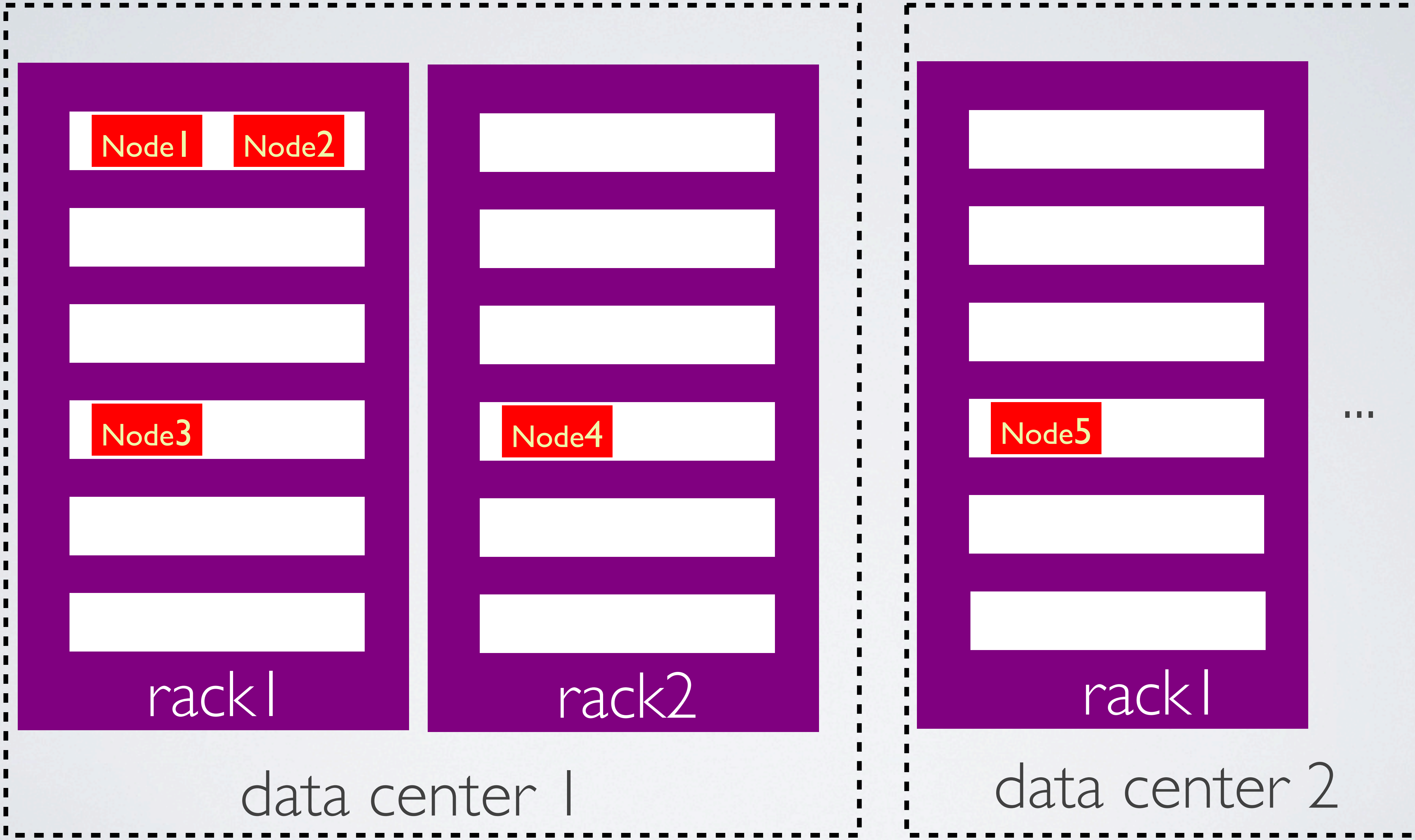
HDFS

In the context of high volume data processing, bandwidth is a scarce commodity.

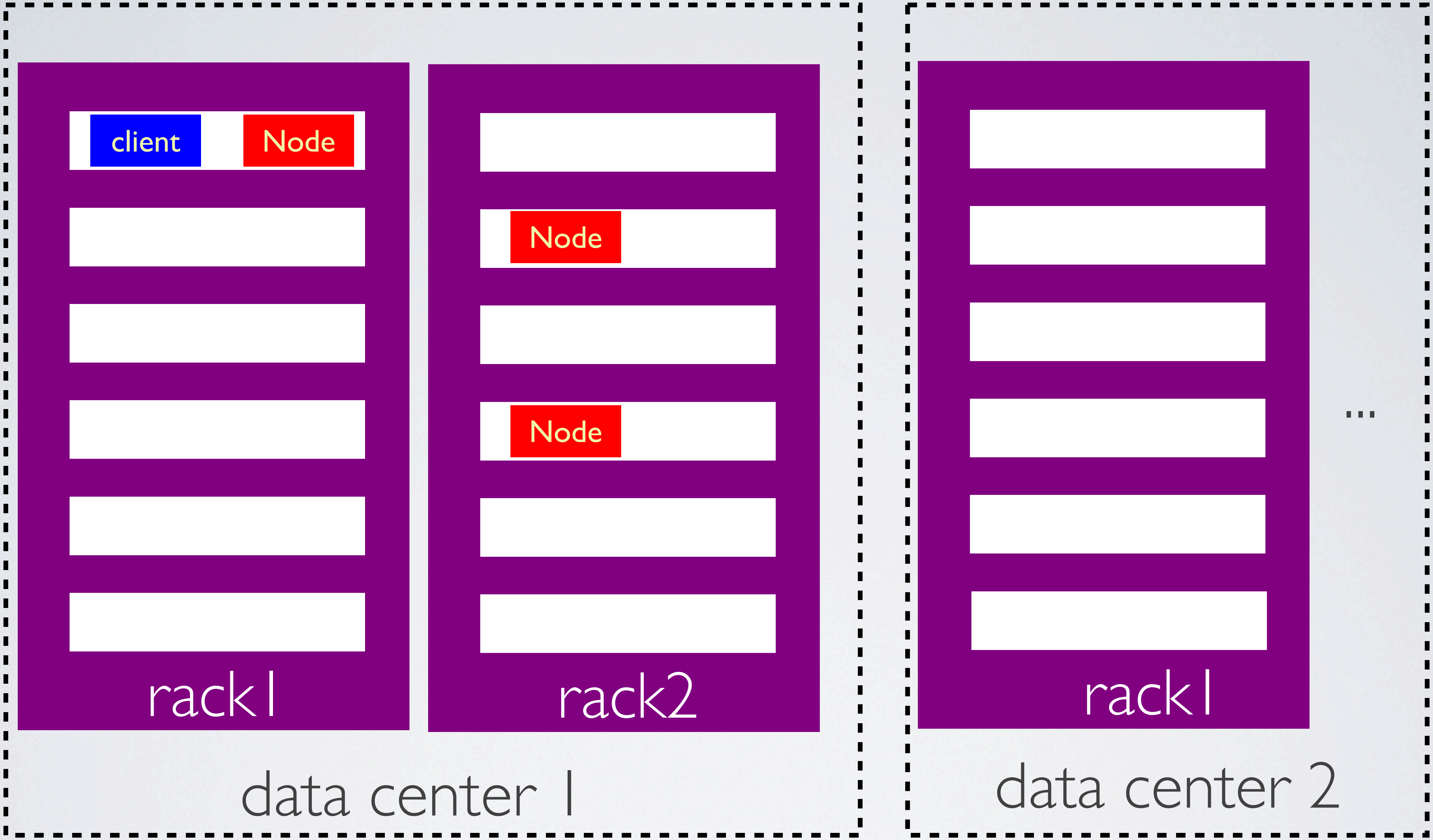
Things get worse, if we consider data replicas.

It makes more sense to use the bandwidth as the measure of distance?

HDFS



Bandwidth (1,2) > Bandwidth(1,3) > Bandwidth(1,4) > Bandwidth (1,5)



How to replicate?

It's tradeoff between reliability and bandwidth.

A lot of constraints in Hadoop for performance consideration...

But data redundancy is always the design goal in mind.

Lots of other alternative filesystem...

What about a generic data replication scheme?

Thanks!