P2P Storage Systems: How Much Locality Can They Take?

[a.k.a Sprederies]

Olivier Dalle
Frédéric Giroire
Julian Monteiro
Stéphane Pérennes

Le Boreon
Mar. 13, 2009

Motivation

- Sensitive data
  - Commercial corporate data: client databases, ...
  - Personal data: photo of your favorite goldfish, ...
- Frequent disk failures
- Fire, flooding, earthquake, martian invasion, bugs in software ...
- Tradition Approaches → High Cost
  - Structure: robust dedicated servers + IT group.
  - Several data centers in different areas.

→ **reliable storage**: replicate data and spread copies among different peers.

Motivation

- Sensitive data
  - Commercial corporate data: client databases, ...
  - Personal data: photo of your favorite goldfish, ...
- Frequent disk failures
- Fire, flooding, earthquake, martian invasion, bugs in software ...
- Tradition Approaches → High Cost
  - Structure: robust dedicated servers + IT group.
  - Several data centers in different areas.

→ **reliable storage**: replicate data and spread copies among different peers.

Motivation

- Sensitive data
  - Commercial corporate data: client databases, ...
  - Personal data: photo of your favorite goldfish, ...
- Frequent disk failures
- Fire, flooding, earthquake, martian invasion, bugs in software ...
- Tradition Approaches → High Cost
  - Structure: robust dedicated servers + IT group.
  - Several data centers in different areas.

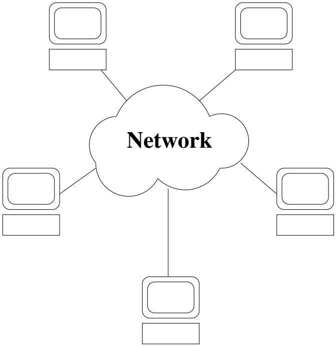
→ **reliable storage**: replicate data and spread copies among different peers.

Motivation

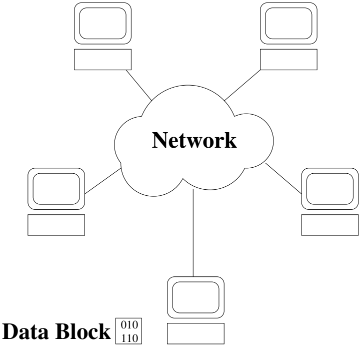
- Sensitive data
  - Commercial corporate data: client databases, ...
  - Personal data: photo of your favorite goldfish, ...
- Frequent disk failures
- Fire, flooding, earthquake, martian invasion, bugs in software ...
- Tradition Approaches → High Cost
  - Structure: robust dedicated servers + IT group.
  - Several data centers in different areas.

→ **reliable storage**: replicate data and spread copies among different peers.

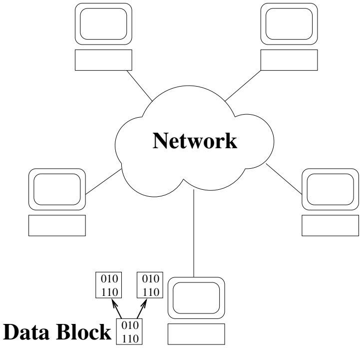
P2P Storage System: How does that works?



P2P Storage System: How does that works?



P2P Storage System: How does that works?



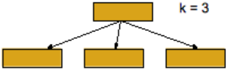
Introduction of redundancy

P2P Storage System: How does that works?

2 methods for redundancy:

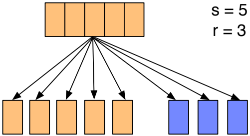
Replication

- Data duplicated k times
- Tolerance: $k-1$ faults
- Usable storage: $1/k$

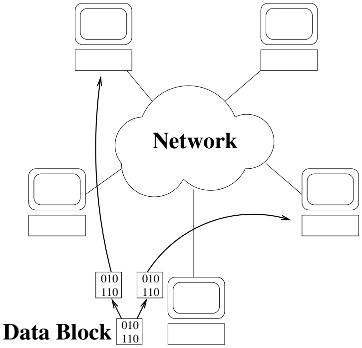


Error Correcting Codes (e.g. Reed Solomon)

- s initial fragments + r redundancy fragments
- Tolerance: r faults
- Usable storage: $s/(s+r)$

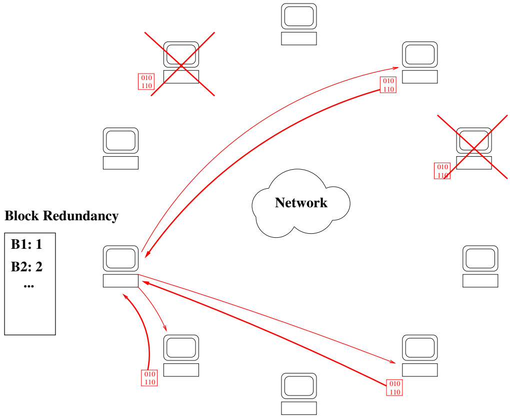


P2P Storage System: How does that works?



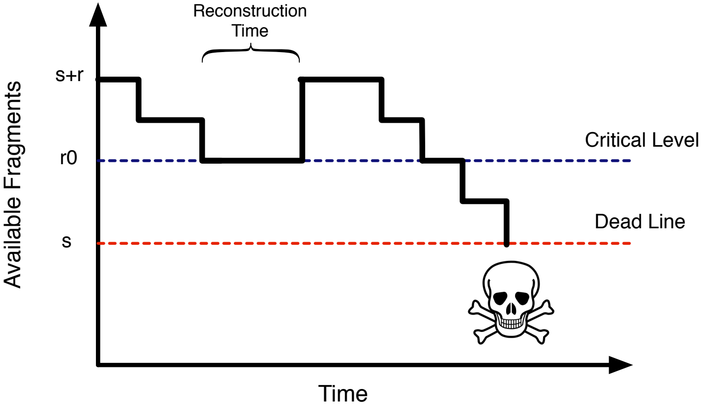
Replicas sent to different peers

Reconstruction



Fragments downloaded by the node in charge of reconstruction

Redundancy



Lifetime of one data block

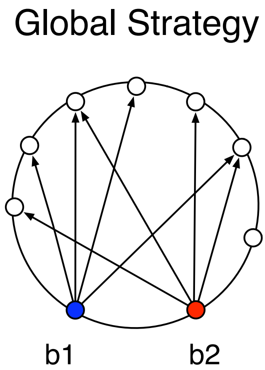
Questions that arise

- "When to reconstruct?"
  → Reconstruction policies and best parameters.
- "How much data will be lost?"
  → Probability to lose data.
- "How much bandwidth used by reconstruction?"
  → Resources usage.
- "Where to place replicas?"
  → placement policies and its impact in performances

Questions that arise

- "When to reconstruct?"
  → Reconstruction policies and best parameters.
- "How much data will be lost?"
  → Probability to lose data.
- "How much bandwidth used by reconstruction?"
  → Resources usage.
- *** "Where to place replicas?" ***
  → placement policies and its impact in performances

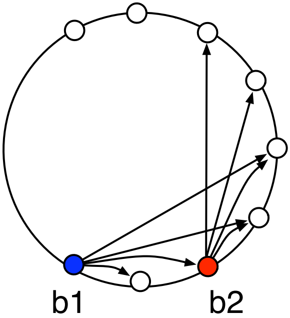
Where to place replicas?



- Global strategy: fragments are distributed to $s + r$ peers chosen at random among all peers
- Previous work: correlation between data-block failures
  single disk crash: tens of thousand of pieces lost
  $\rightarrow$ Markov Chain Models and Fluid Models

Where to place replicas?

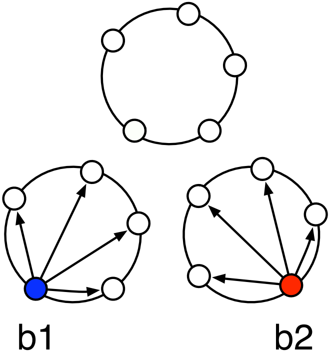
Chain Strategy



- The Chain strategy: Fragment are stored on the $s + r$ closest logical neighbors (used by many DHTs).
(also named “local” here)

Where to place replicas?

Buddy Strategy



- The buddy strategy: Many small subsystems of size $s + r$. Fragments are stored on peers of the same group.

Which placement strategy is better?

- Probability to lose data
- Mean Time To Data Loss (MTTDL)
- Bandwidth used by reconstruction

It depends on the resource constraints

Which placement strategy is better?

- Probability to lose data
- Mean Time To Data Loss (MTTDL)
- Bandwidth used by reconstruction

It depends on the resource constraints

Which placement strategy is better?

Related Work

- Lien et al. (2005): Mean Time to Data Loss (MTTDL) metric.
  They show that MTTDL is better for the chain policy
  $\longrightarrow$ They do not talk about the *probability to lose a block*
- Chun et al. (2006): chain policy induces higher reconstruction times, thus, lower durability
  $\longrightarrow$ They do not address bandwith provisioning scenarios

.

Without resource constraints

- Global, Chain and Buddy consume the same amount of resources
- However, the variations are different

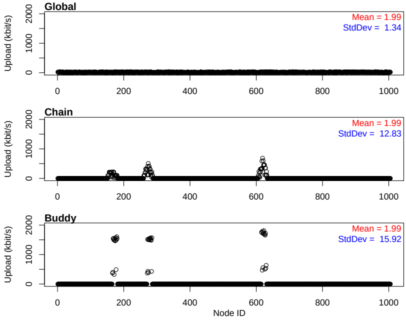


Figure: Variations of maintenance bandwidth usage across users

Without resource constraints

- Global, Chain and Buddy have the same prob. to lose data
- However, the variations are also different !

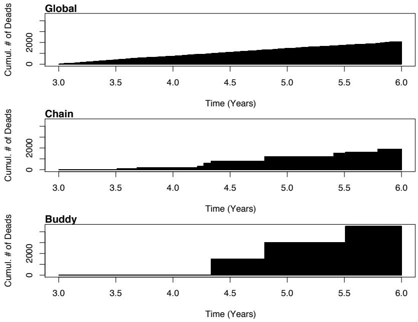


Figure: Cumulative number of dead blocks for three years.

- MTTDL (Mean Time To Data Loss) is higher in the Buddy.

With bandwidth constraints per peer

- Reconstruction time is very high in Chain and Buddy policies

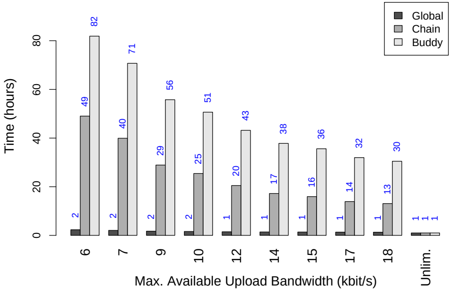


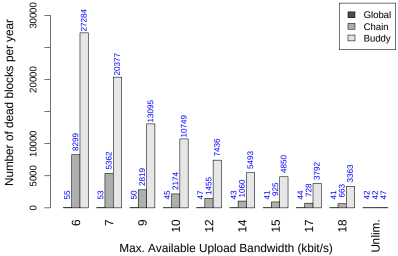
Figure: Average reconstruction times for different bandwidth limits

With bandwidth constraints per peer

- Exponential relation between the probability to die and the reconstruction time

$$\Pr[die|W = T] = \binom{s + r_0}{r_0}(1 - e^{-\beta T})^{r_0}(e^{-\beta T})^{s}.$$

The prob. for a peer to still be alive after a time T is $\exp(-\beta T)$, where β is the peer fault rate



Improvements to the Chain Strategy

- Choosing “external” peers to reconstruct blocks improve the prob. to lose data

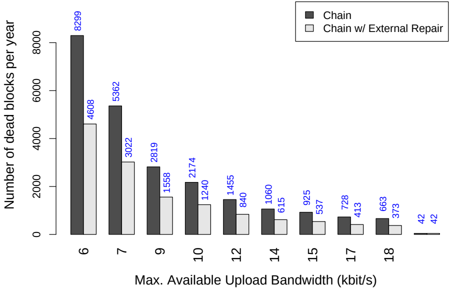


Figure: Number of block losses for different bandwidth limits.

Improvements to the Chain Strategy

- Bigger chain sizes could also improve the prob. to lose data.

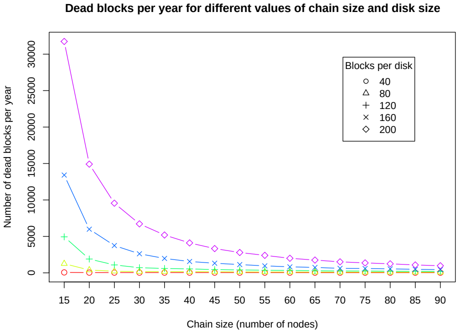


Figure: Study of the size of the block neighborhood. Number of block losses per year for different sizes and different number of fragments per disks.

Simulations

- Home made cycle-based simulator.
  → does not have fine granularity;
  → but it is easy to compare with the analytical models

- In each cycle of 1 hour:
  → induces disk failures
  → monitors critical blocks
  → finish reconstructions
  → reinjects dead blocks and dead peers in the system (to maintain stability)

- Simple queue based network layer
  → each peer has a upload/download capacity per cycle
  → a global FIFO order is imposed

Simulation parameters

- Number of peers, $N := 1005$
- Number of blocks, $B := 2 \cdot 10^5$
  (i.e. only 600MB of data per peer)
- $s := 9$, $r := 6$, $r_0 := 3$ (fragment size $400KB$)
- MTBF of disks: 90 days
- Time to perceive a disk failure: 12 hours
- Number of fragments, $F := 3 \cdot 10^6$
- Simulation time: 20 years
  (i.e. $4.3 \cdot 10^4$ cycles)

Obrigado!

Obrigado!