# The what, The from, and The to: The Migration Games in Deduplicated Systems

Roei Kisous and Ariel Kolikant, Technion - Israel Institute of Technology;

Abhinav Duggal, DELL EMC; Sarai Sheinvald, ORT Braude College of Engineering;

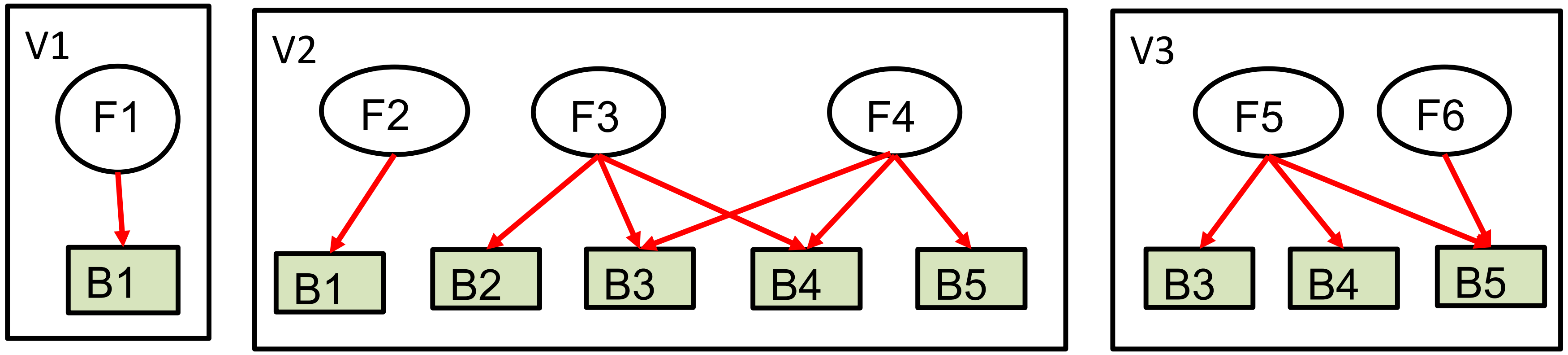
Gala Yadgar, Technion - Israel Institute of Technology

**FAST 22**

# Background

## Storage system with deduplication

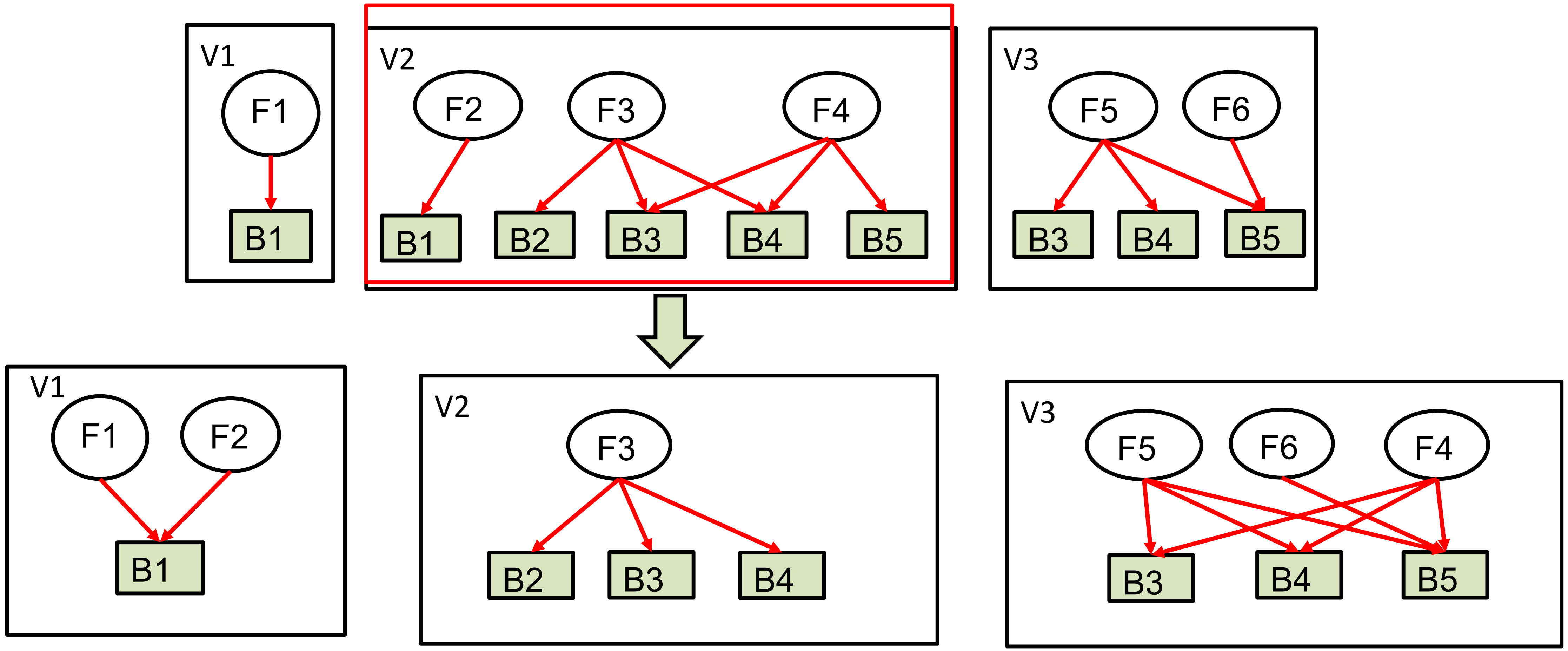
- Just store unique blocks inside the volume, duplicate blocks can exist between volumes.



- Advantage: spave saving
- Disadvange: complex metadata management (insert/delete)

# Background

- **Data migration in deduplication storage system**
  - Volumes reaching their capacity limitation or other bottlenecks forming in the system.
  - Migration can cause blocks to be deleted, replicated, moved



# Existing approach's disadvantage

- **Migration considerations?**
  - Minimize final system size
  - Minimize traffic overhead
  - Load balance

- Harnik et al. (FAST 19) presented a **greedy** iterative algorithm for reducing the total capacity of data in a system with multiple volumes.
  - In each iteration, one file is remapped to a new volume, and the process continues until the total capacity is reduced by a predetermined deletion goal.
- GoSeed (FAST 20) presented an **ILP** algorithm which migration goal is to **delete a portion of a volume's blocks** by remapping files to an empty target volume.

# Motivation

- **Formulate the general migration problem for deduplicated systems as an optimization problem and give the solutions.**
  - **Minimize the system's size.**
  - **Storage load balance**
  - **Network traffic required for the migration does not exceed its allocation.**

# Formal expression

- **S with a set of volumes V, let B ={b0,b1, . . .} be unique blocks. Let F = { f0, f1, . . .} be files in S. Let IS ⊆ B × F ×V be an inclusion relation, where (b, f ,v) $\in$ IS means that file f mapped to volume v contains block b which stored in this volume. For simplicity, use b $\in$ v to denote that (b, f ,v) $\in$ IS for some file f .**

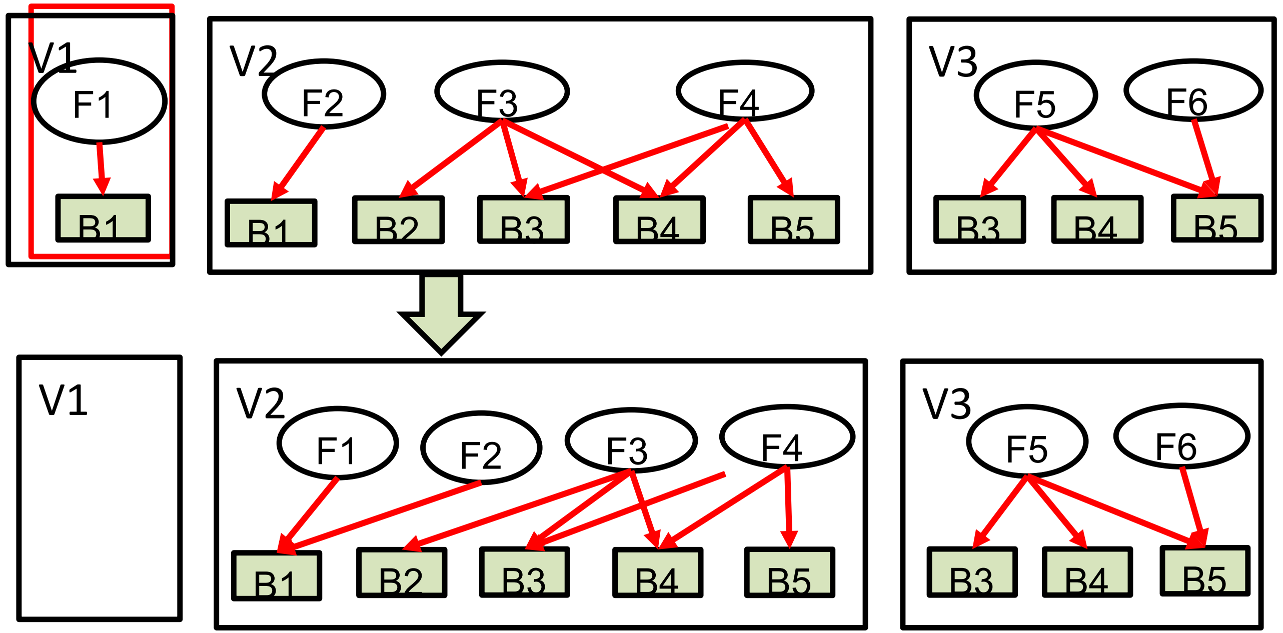
- **A volume size(v) = $\sum_{b \in \mathrm{V}} size(b)$. The whole system size(S) = size(V ) = $\sum_{v \in \mathrm{V}} size(v)$.**

- **The general migration problem is to find a set of files FM ⊆ F to migrate, the volume each file is migrated to, the blocks that can be deleted from each volume, and the blocks that should be copied to each volume. Applying the migration plan results in a new system, S'.**

- **The migration goal is to minimize the size of S'.**
- **Constraint:**
    - traffic constraint specifies Tmax: total size of blocks that are added to volumes they were not stored in is no larger than Tmax.
    - load balancing specifies a margin 0 ≤ μ < 1: the size of each volume in S' is within μ of the average volume size.

# Greedy

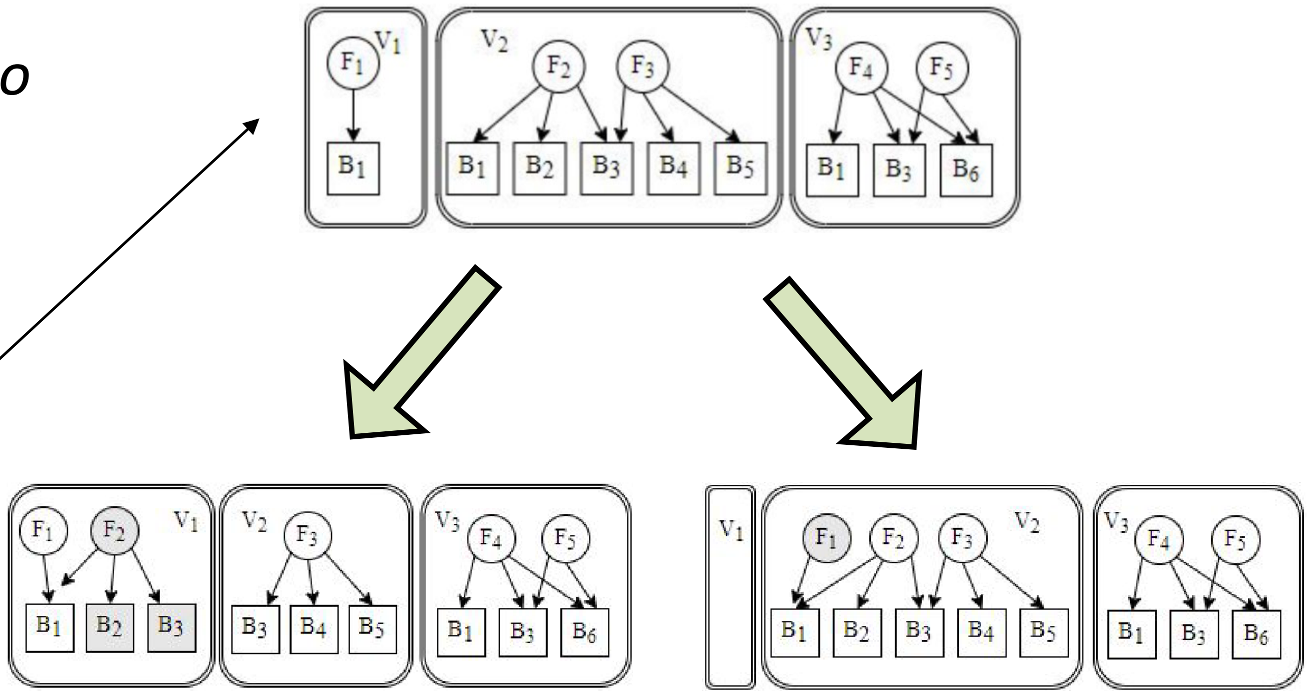
- Basic idea:
  - *Iterates over all the files in each volumes, and calculates the space-saving ratio from remapping a single file to each of the other volumes*
    - **Space-saving ratio**: the ratio between the total size of the blocks that would be replicated and the blocks that would be deleted from the file's original volume.
  - *In each iteration, file with the lowest ratio is remapped. The process halts when the total capacity is reduced by a predetermined deletion goal*



# Greedy

- **Traffic constraint :**
  - *The iterations stop when there is no file that can be remapped without exceeding the maximum allocated traffic.*

- **load-balancing constraint:**
  - **Challenge:** load balance conflicts with system size reduce.
  - Load balance may caught stop warly.

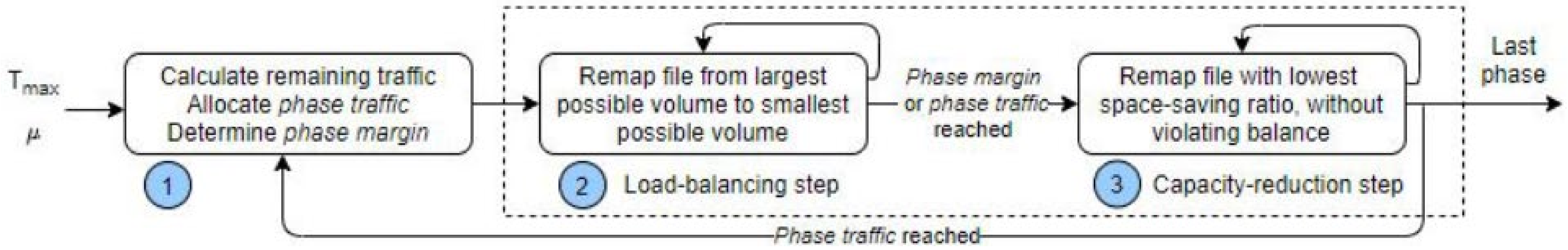


Alternative 1

Alternative 2

- **Solution:**
  - ***First is defining two iteration types: one whose goal is to balance system's load, and another whose goal is to reduce its size.***
  - ***The second is to relax load-balancing margin for early iterations and continuously tighten it until end of execution.***

# Greedy



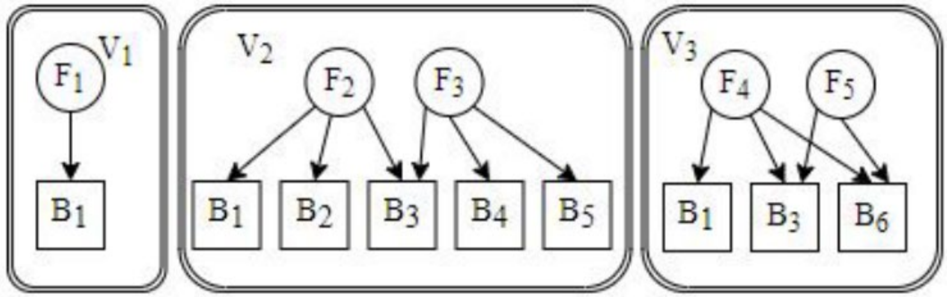
$T_{max}$
$\mu$
Calculate remaining traffic
Allocate *phase traffic*
Determine *phase margin*
1
Remap file from largest possible volume to smallest possible volume
2
Load-balancing step
*Phase margin* or *phase traffic* reached
Remap file with lowest space-saving ratio, without violating balance
3
Capacity-reduction step
Last phase
*Phase traffic reached*

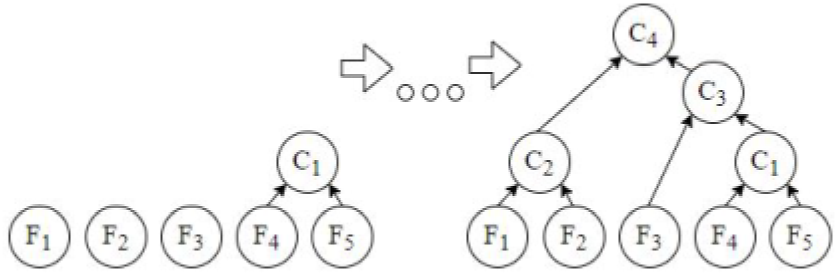
- ➢ Three phases:
  - ➢ Caculate each iteration’s traffic Tmax’ and u’, u’ is gradually shrink
  - ➢ Load-balancing and capacity-reduction alternate execution.

# Cluster

- Basic idea: (based on ***Hierarchical clustering***)
  - *Jabcco Distance metric (sample):*

    $$dist_j(A,B)=1 - |A\cap B|/|A\cup B|.$$
  - *Use distance metric to cluster.*





|  | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ |
|---|---|---|---|---|---|
| $F_1$ | - | - | - | - | - |
| $F_2$ | 2/3 | - | - | - | - |
| $F_3$ | 1 | 4/5 | - | - | - |
| $F_4$ | 2/3 | 2/3 | 4/5 | - | - |
| $F_5$ | 1 | 3/4 | 3/4 | 1/3 | - |

$\xrightarrow{F_4,F_5}$

|  | $F_1$ | $F_2$ | $F_3$ | $C_1$ |
|---|---|---|---|---|
| $F_1$ | - | - | - | - |
| $F_2$ | 2/3 | - | - | - |
| $F_3$ | 1 | 4/5 | - | - |
| $C_1$ | 1 | 3/4 | 4/5 | - |

$\xrightarrow{F1,F2}$

|  | $C_2$ | $F_3$ | $C_1$ |
|---|---|---|---|
| $C_2$ | - | - | - |
| $F_3$ | 1 | - | - |
| $C_1$ | 1 | 4/5 | - |

$\xrightarrow{F3,C_1}$

|  | $C_2$ | $C_3$ |
|---|---|---|
| $C_2$ | - | - |
| $C_3$ | 1 | - |

dist(A∪B,C)
=max{dist(A,C),dist(B,C)}

## Cluster

- Their approach is to create clusters of similar files and to assign each cluster to a volume, remapping files that were assigned to a volume different from original location.

- Three challenges:
  - ***Unpredictable traffic***
    - *The traffic required can only be calculated after a migration plan is generated. When clustering decisions are being made, their implications on overall traffic are unknown.*
  - ***Unpredictable system size***
    - *The load-balancing constraint is given by system's size after migration. However, this size is required to ensure created clusters are within the allowed sizes during clustering process.*
  - ***High sensitivity***
    - *The file similarity metric is based on a sample of the storage system's fingerprints, it is highly sensitive to the sampling degree and rule.*

## Cluster

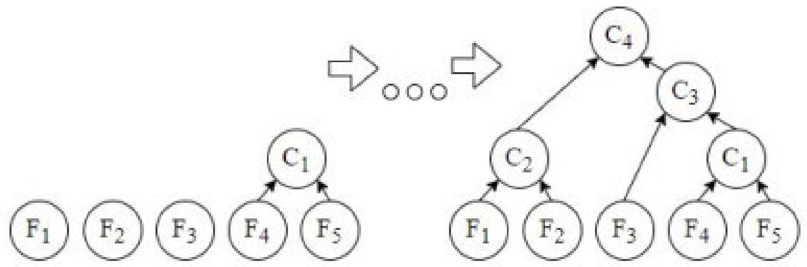
- address these challenges with heuristics:
  - ***Traffic weight ($W_T$ )***
    - *Volume distance of a cluster:*

      $dist_v$ *(C)*

      *presents portion of the system's volumes whose files are included in cluster. eg,* $dist_v\,(c1) = 1/3$
    - *New weighted distance:*
      - $dist_w$ *(A,B) =* $W_T$ *×* $dist_j$ *(A,B) + (1 –* $W_T$ *) ×* $dist_v$ *(A ∪ B)*
      - $W_T$ *: trade-off between same blocks and same volumes.*
      - *Increasing* $W_T$ *increases the amount of traffic allocated for the migration.*

# Cluster

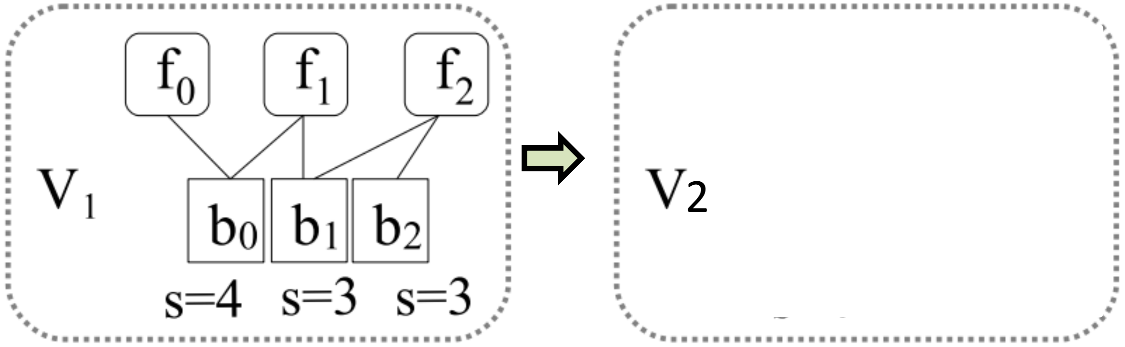
- address these challenges with heuristics:
  - ***Load-balancing considerations***
    - *By preventing merges that result in clusters exceeding the maximal volume size.*
    - *The max volume size = (initial system size + traffic size( generated from traffic weight) )/ |V|*
  - ***Sensitivity to sample***
    - *Choose a random pair⟨Ci,Cj⟩from the 10 (or less) smallest distance pairs rather than the smallest pair.* optimal but slowest
      - ***Example, choose form a pair from pairs DistW (Ci,Cj) ≤ minimum distance × (1 + gap), gap is a parameter***
  - ***Constructing the final migration plan***
    - *Execute 180 compositions of weighted traffic, gap and other parameters. Finally use the best migration plan.*



- Hierarchical clustering
- Use weighted traffic distance.
- Enforce load-balance by estimated max volume size.
- Introduce Multiple Selection Sampling to Reduce Sampling Errors

# Integer Linear Programming

- It is inspired by GoSeed [FAST 20]



(1) $0 \leq x_0, x_1, x_2, m_0, m_1, m_2, r_0, r_1, r_2 \leq 1$

(2) $m_0 \leq x_0,\ m_0 \leq x_1,\ m_1 \leq x_1,\ m_1 \leq x_2,\ m_2 \leq x_2$

(3) $x_0 \leq m_0 + r_0,\ x_1 \leq m_0 + r_0,\ x_1 \leq m_1 + r_1,$
$x_2 \leq m_1 + r_1,\ x_2 \leq m_2 + r_2$

(4) $4 \cdot m_0 + 3 \cdot m_1 + 3 \cdot m_2 = 3$

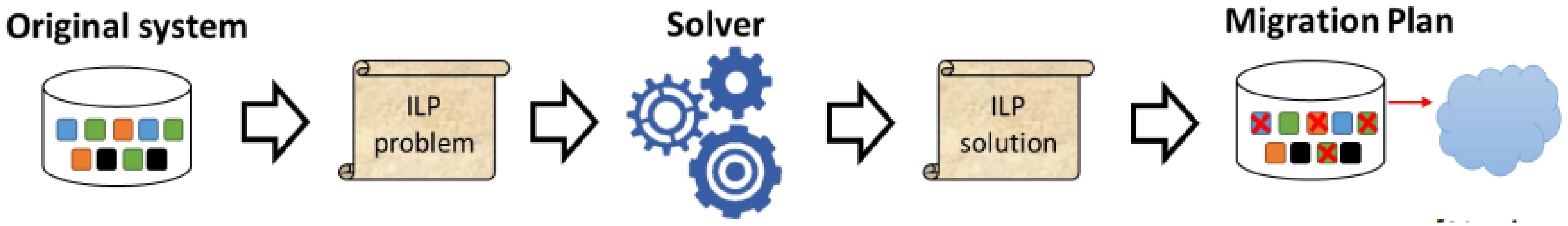
Goal: minimize $4 \cdot r_0 + 3 \cdot r_1 + 3 \cdot r_2$

- *Assigning 1 to $x_i$ means that fi is remapped from V1 to V2.*
- *Assigning 1 to $m_i$ means that bi is migrated from V1 to V2.*
- *Assigning 1 to $r_i$ means that bi is replicated and will be stored in both V1 and V2.*

# Integer Linear Programming

- Continue to use Goseed's three variable.

- The ILP formulation for migration consists of **13 constraint** types. (Goseed is 4)
  - *Traffic constraint: the size of all the copied blocks is not larger than the maximum allowed traffic Tmax.*
  - *Load balancing constraint: for each volume v, $(w_v - \mu) \times Size(S') \leq Size(v') \leq (w_v + \mu) \times Size(S')$, where Size(v') is the volume size after migration. $w_v = 1/|v|$*



# Experimental setup

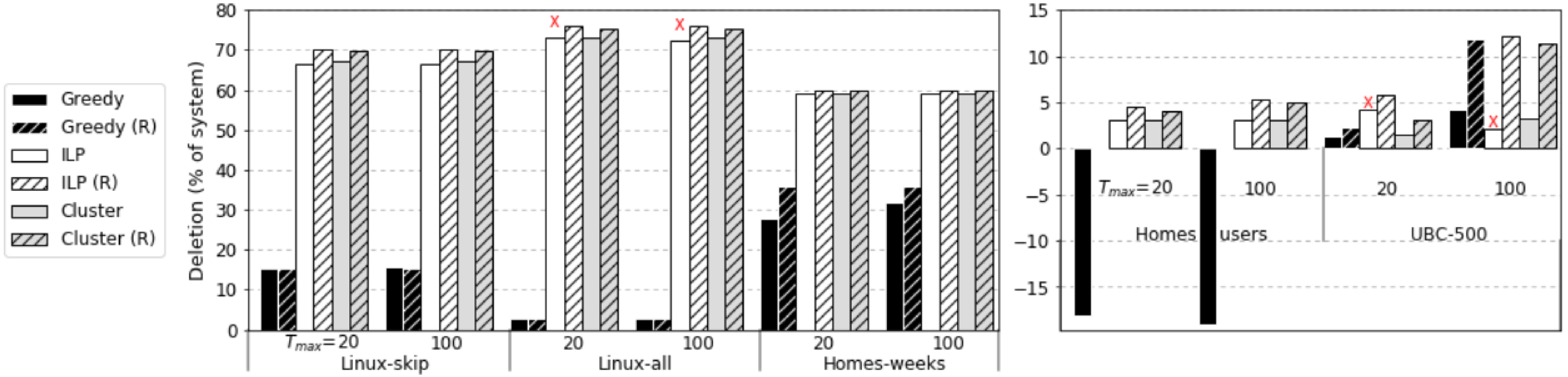
- **Testbed**: Ubuntu 18.04.3, 128GB DDR4 RAM (with 2666 MHz bus speed), Intel® Xeon® Silver 4114 CPU (with hyper-threading functionality) running at 2.20GHz, one Dell®T1WH8 240GB TLC SA TA SSD, and one Micron 5200 Series 960GB 3D TLC NAND Flash SSD.
- **Datasets:**

physical /logical size

| System | Files | $\|V\|$ | Chunks | Dedupe | Logical |
|---|---|---|---|---|---|
| UBC-500 | 500 | 5 | 382M | 0.39 | 19.5 TB |
| Homes-week | 81 | 3 | 19M | 0.38 | 8.9 TB |
| Homes-user | 81 | 3 | 19M | 0.16 | 8.9 TB |
| Linux-skip | 662 | 5 / 10 | 1.76M | 0.12 / 0.19 | 377 GB |
| Linux-all | 2703 | 5 | 1.78M | 0.03 | 1.8 TB |

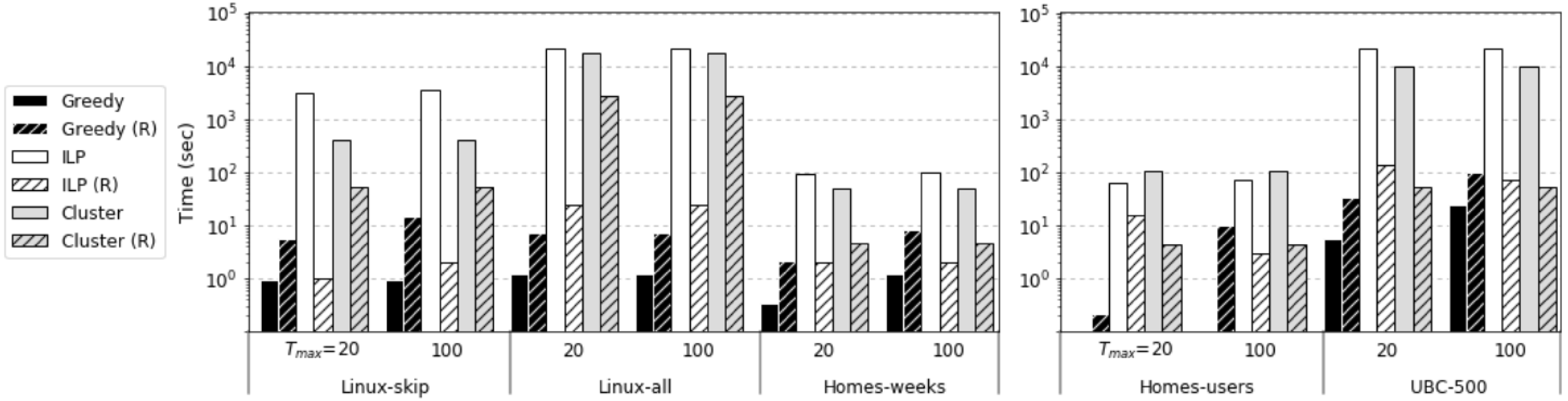
# Evaluation

## ➢ Reduction in system size



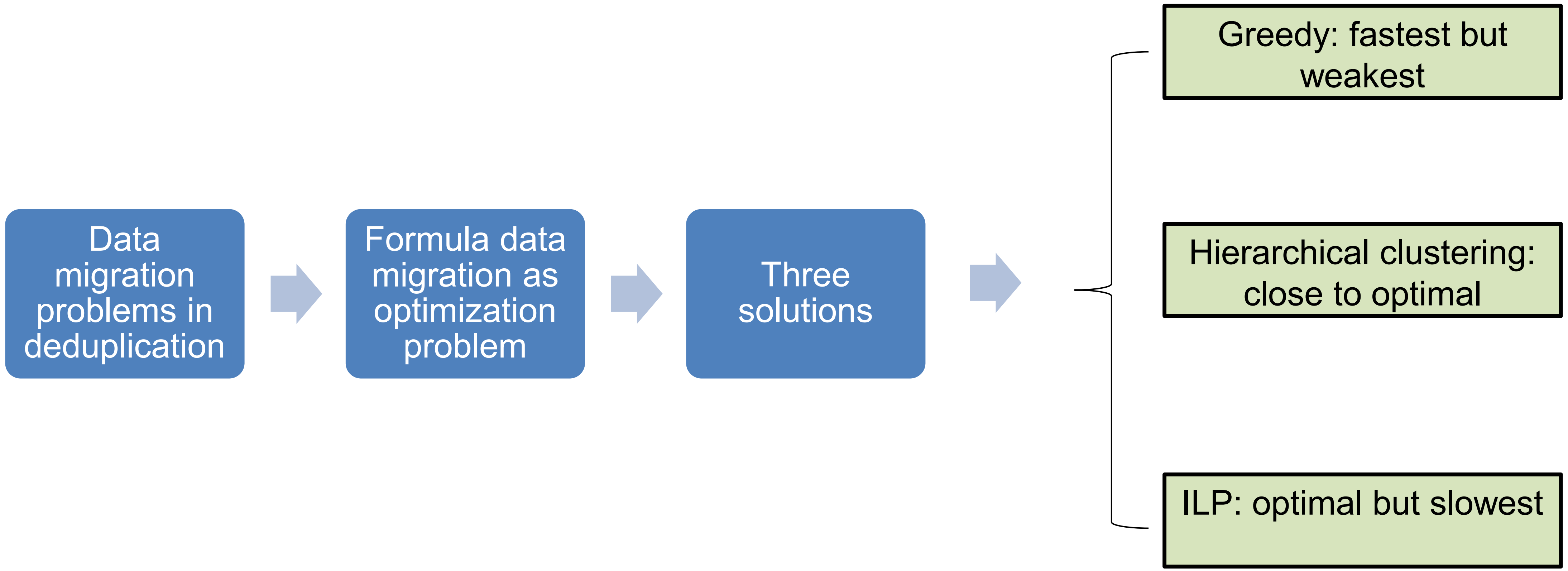
- Clustering ≈ ILP
- Greedy is far behind
- Balance constraint
  - Limits size reduction
- Traffic constraint
  - Little influence

# Runtime



- Greedy generates a migration plan in the shortest runtime: 20 seconds or less.
- ILP requires the longest time, because it attempts to solve an NP-hard problem ILP requires more than an hour, and often halts at the six-hour timeout.
- The runtime of Cluster is longer than that of Greedy, and usually shorter than that of ILP . It is still relatively long, as a result of performing 180 executions of the clustering process.

# Conclusion&Comments


Data migration problems in deduplication
Formula data migration as optimization problem
Three solutions
Greedy: fastest but weakest
Hierarchical clustering: close to optimal
ILP: optimal but slowest