SSE and SSD:

Page-Efficient Searchable Symmetric Encryption

- Angèle Bossuat — Quarkslab
- Raphael Bost — DGA
- Pierre-Alain Fouque — Université de Rennes
- Brice Minaud — ENS — CNRS — PSL University — INRIA
- Michael Reichle — ENS — CNRS — PSL University — INRIA

Motivation

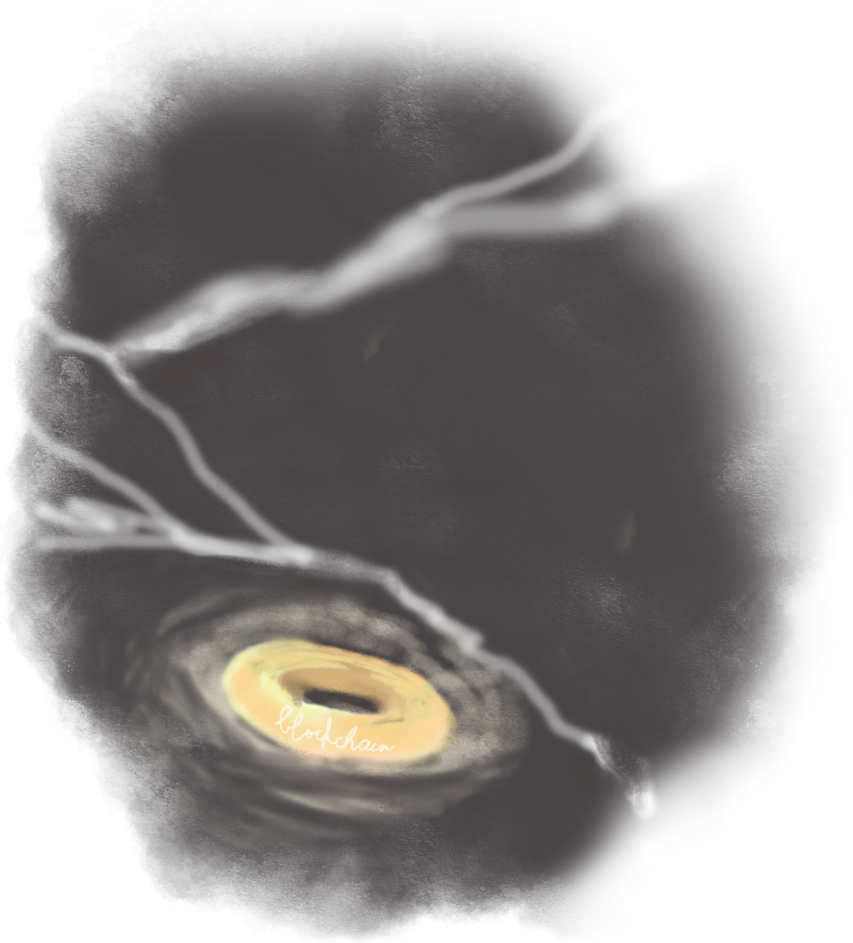
Outsource confidential information

Motivation

Outsource confidential information

Motivation

Outsource confidential information

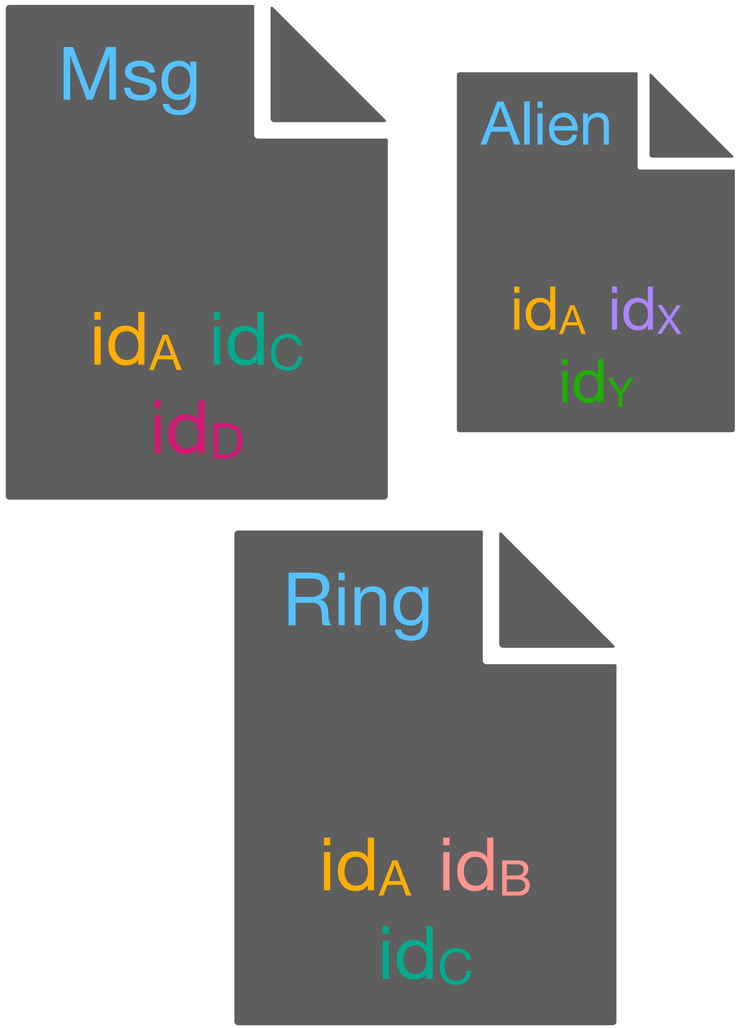


Motivation

Outsource confidential information

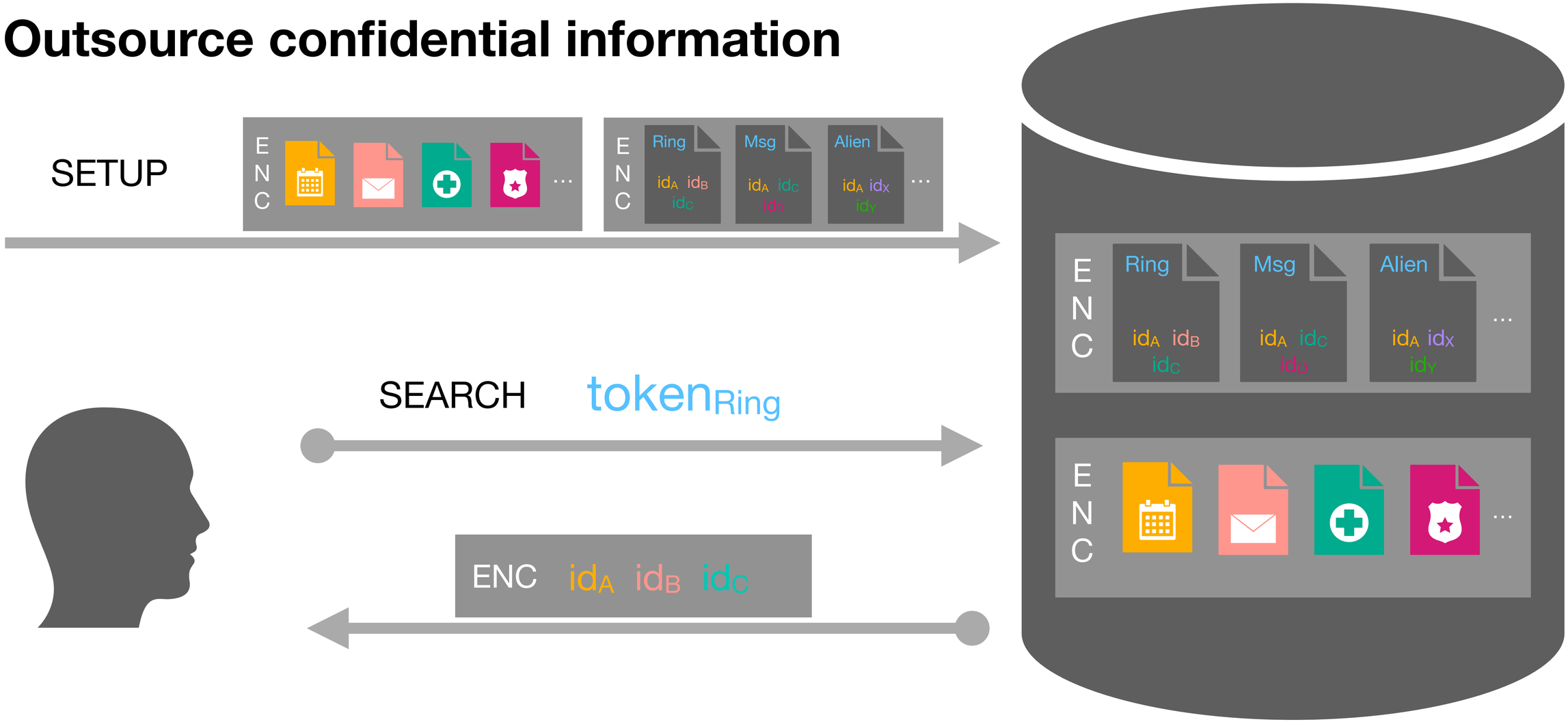
Motivation

Outsource confidential information



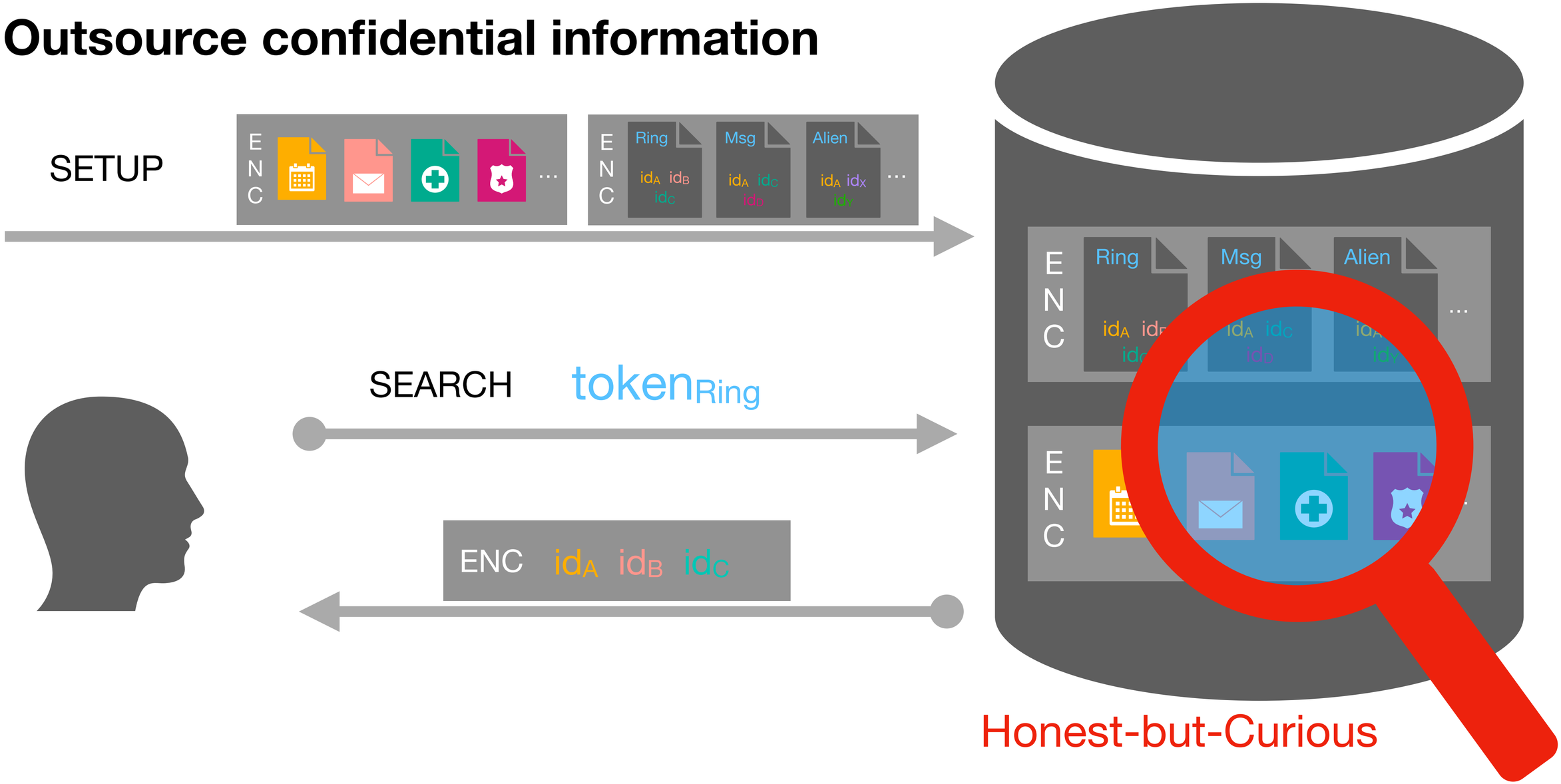
Motivation

Outsource confidential information



Motivation

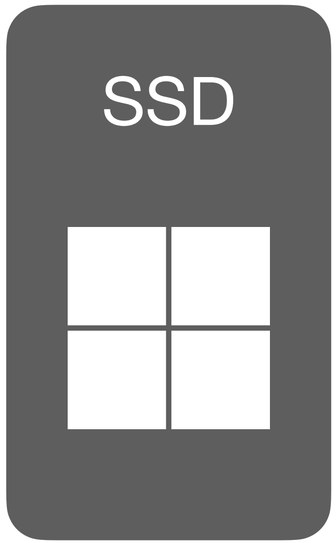
Outsource confidential information



HDDs vs SSDs

Locality:

Number of Read
(non-adjacent)
Memory Locations

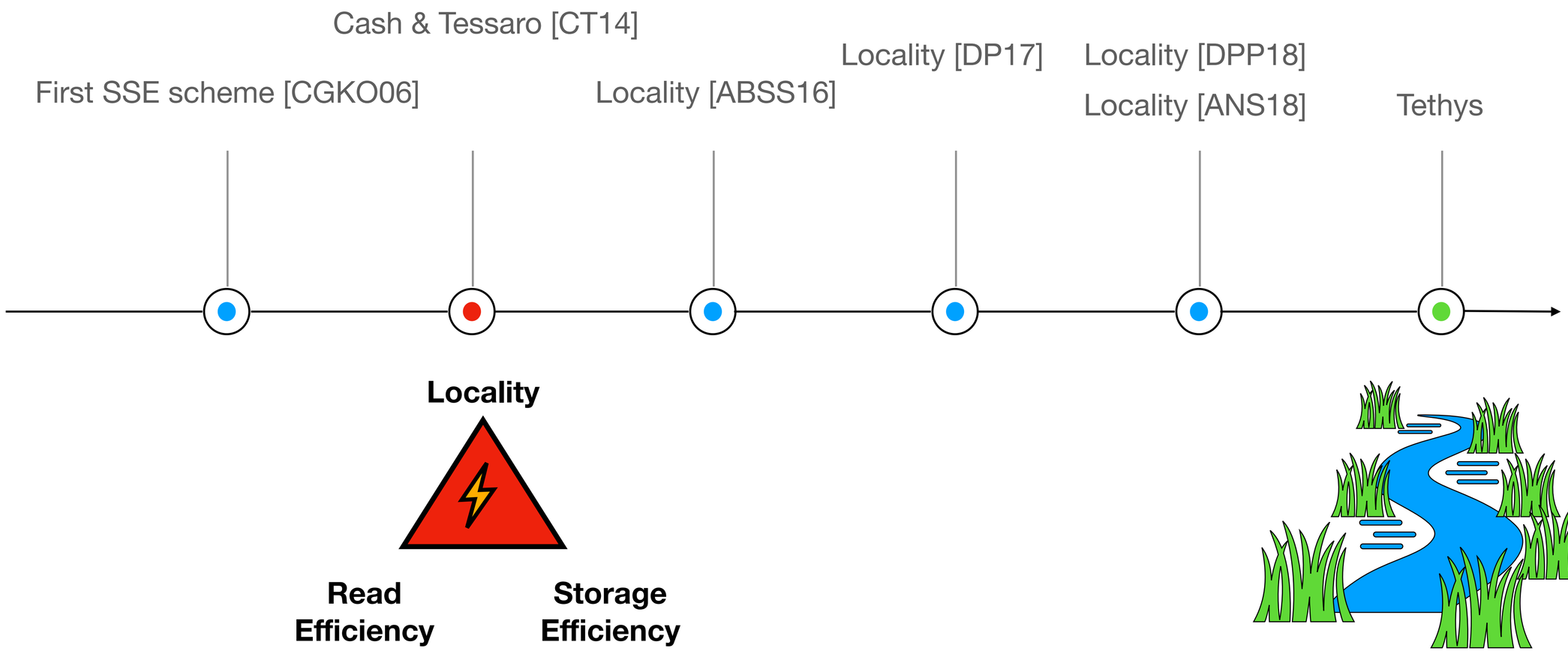


Page Efficiency:

Number of Read
Pages per Query

SSE Research Progress

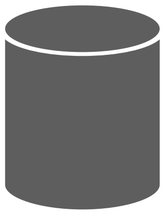
selected articles



Goals

Efficient SSE

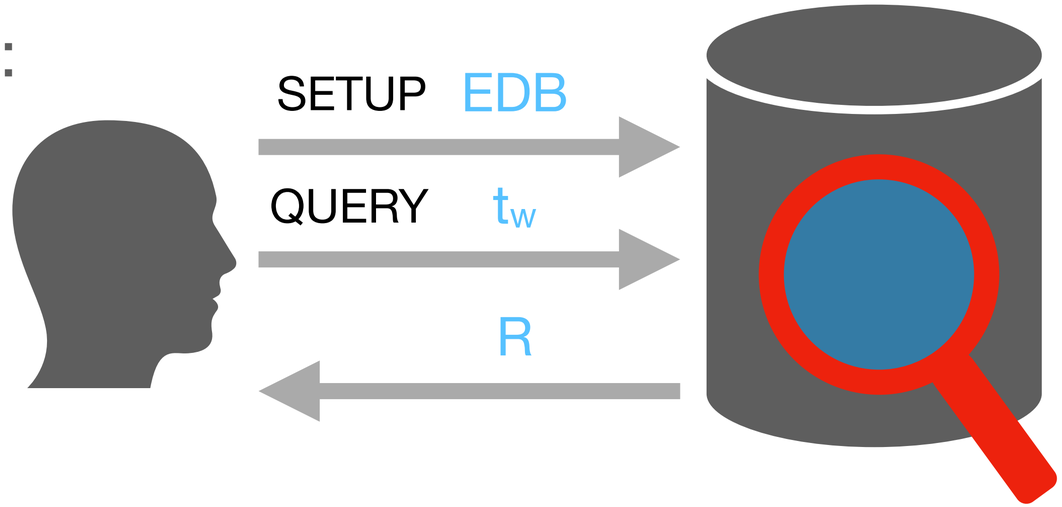
- Page Efficiency:
  - Minimize Page Accesses per Query
- Storage Efficiency:
  - Minimize Server Storage
- Security:
  - Guarantees on Leaked Information



Security

Static SSE

- Game:

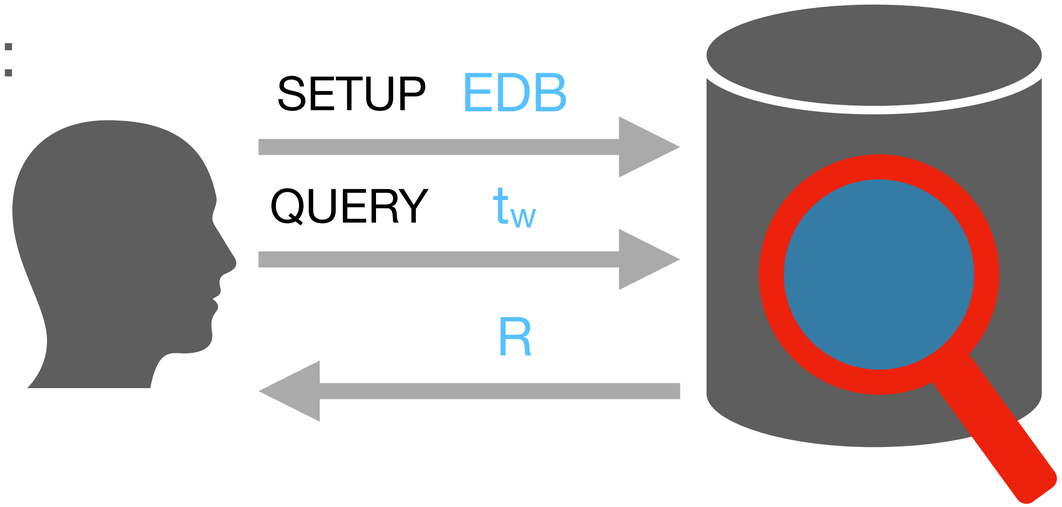


- EDB = SSE.Setup(DB)
- t_w = SSE.Search(DB, w)

Security

Static SSE

- Game:



- $EDB = SSE.Setup(L_{setup}(DB))$
- $t_w = SSE.Search(L_{query}(DB, w))$

- Leakage Function $L = (L_{setup}, L_{query})$:
  - $L_{setup}(DB)$ size(DB)
  - $L_{query}(DB, w)$ size(ids_w), search pattern

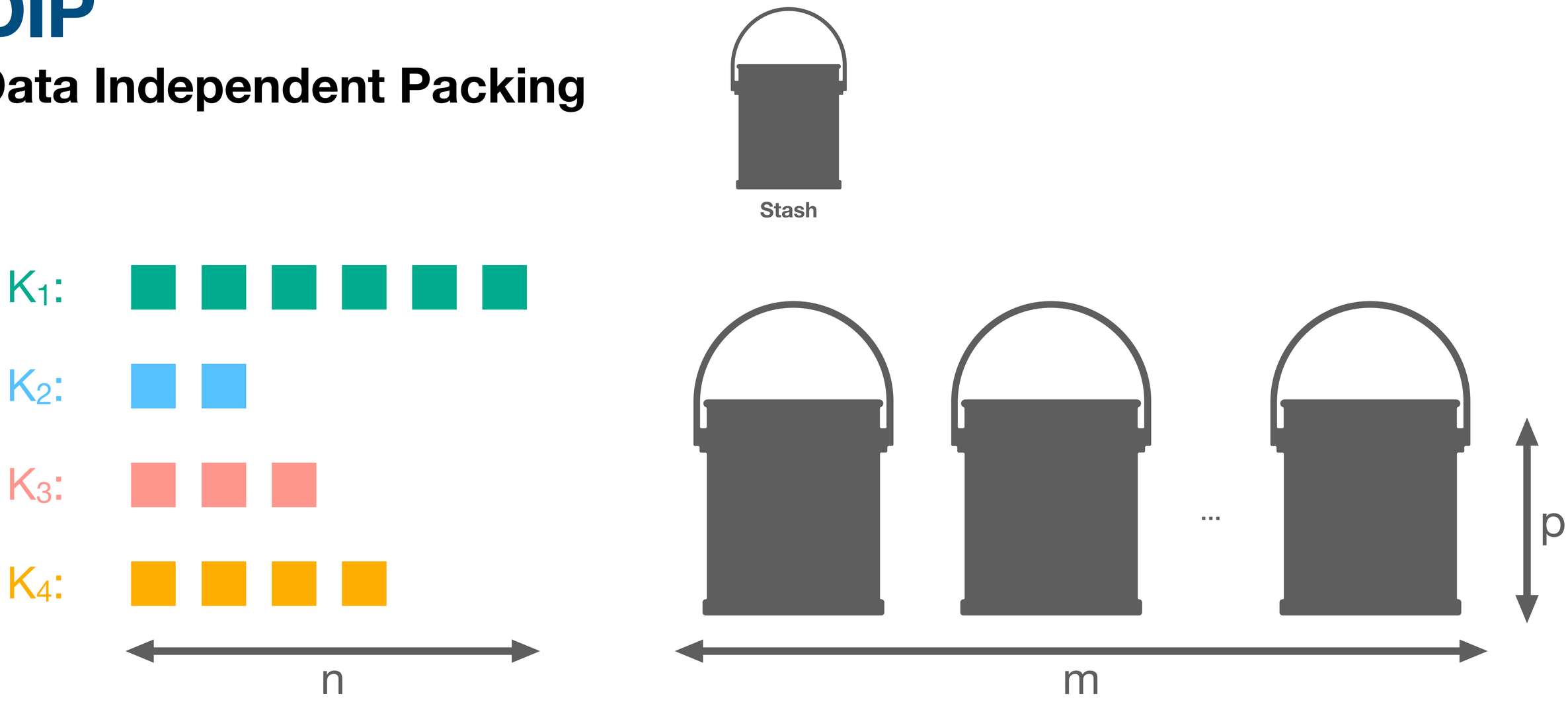
Contribution

This Work

- Page Efficiency:
  - Definition
- Data Independent Packing (DIP):
  - Framework: SSE from DIP
- Tethys:
  - Efficient DIP scheme
  - Based on Cuckoo-Hashing for Weighted Items
  - SSE: optimal page and storage efficiency

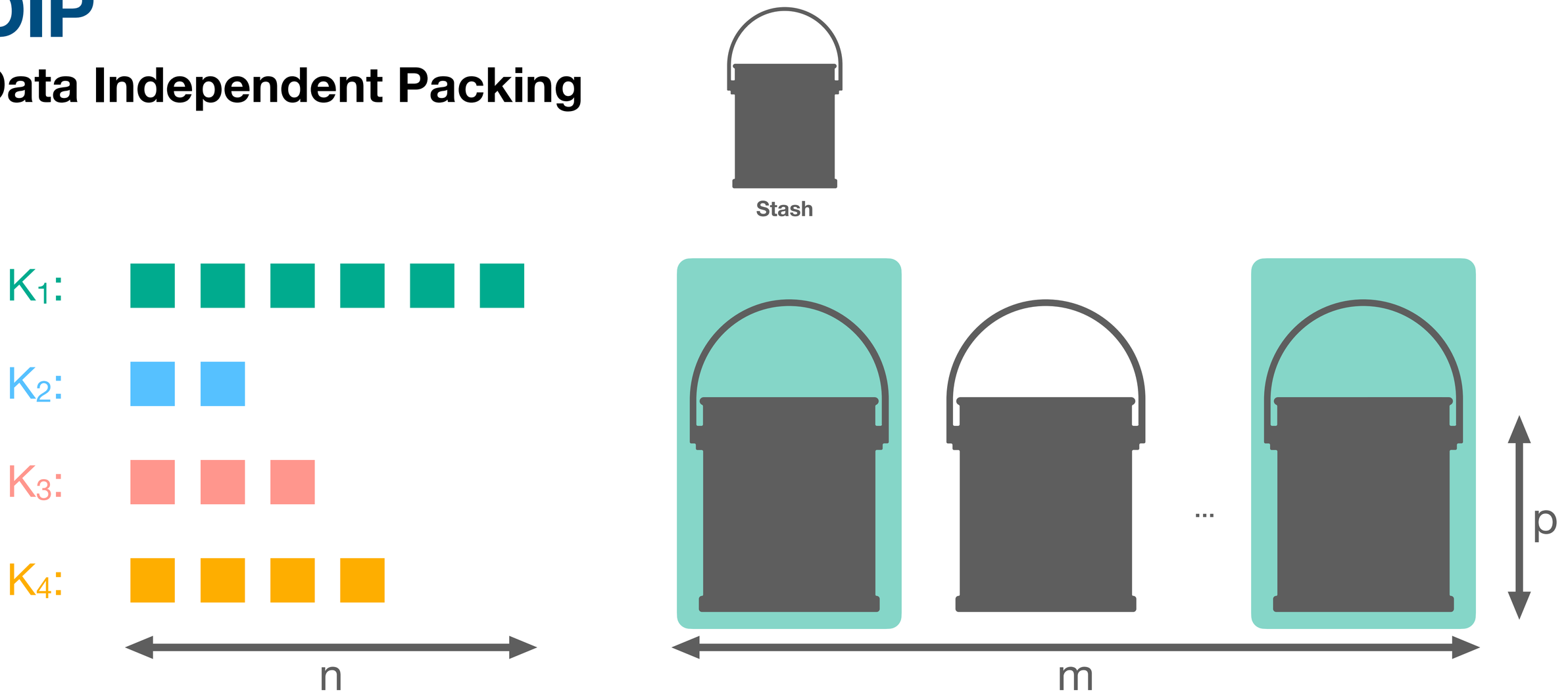
DIP

Data Independent Packing



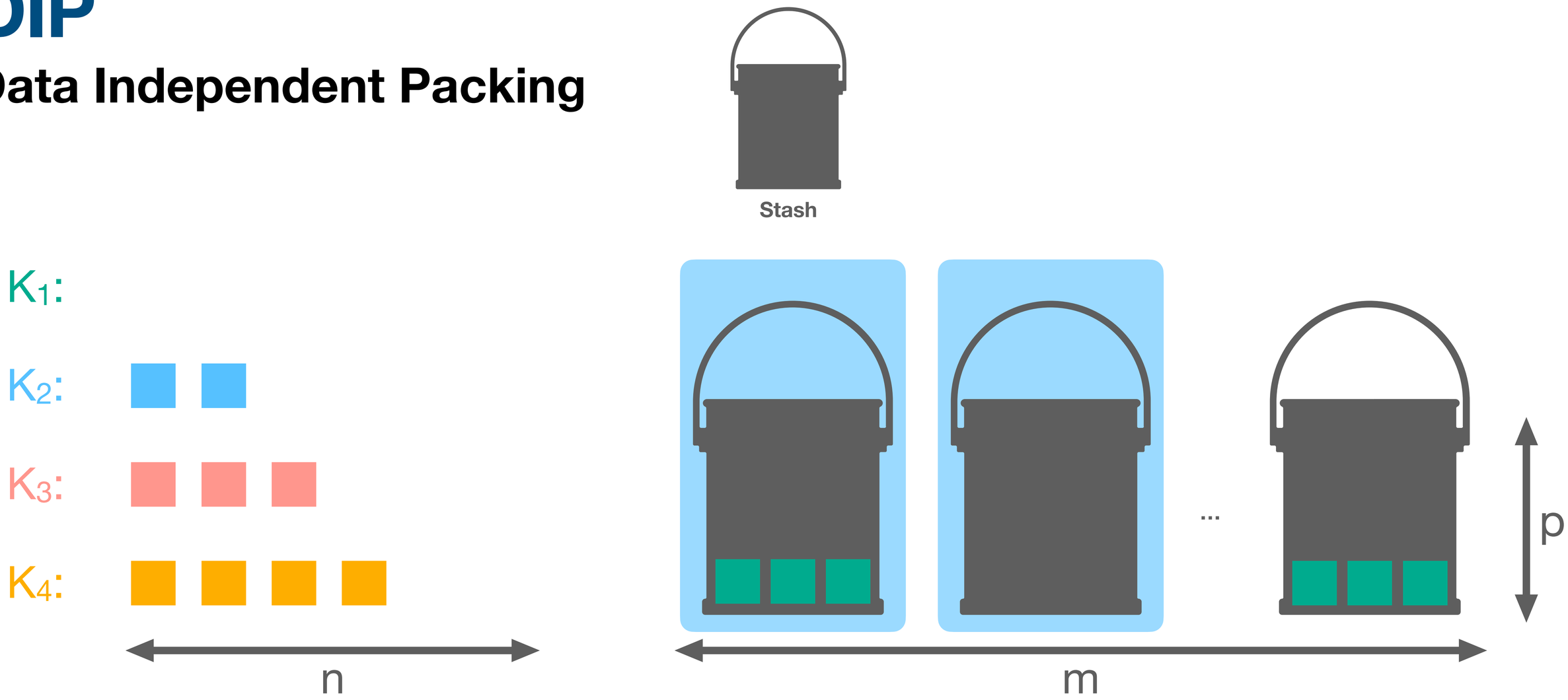
DIP

Data Independent Packing



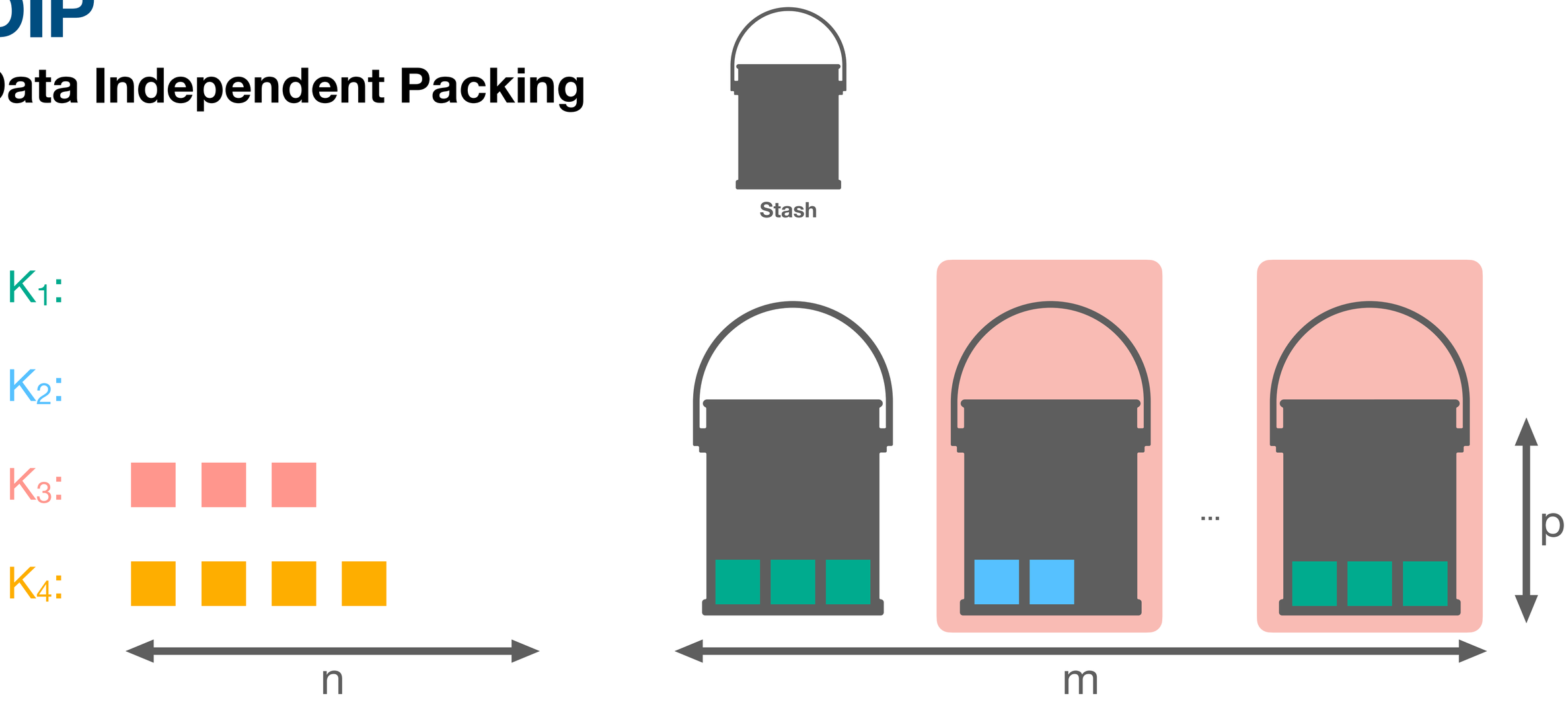
DIP

Data Independent Packing



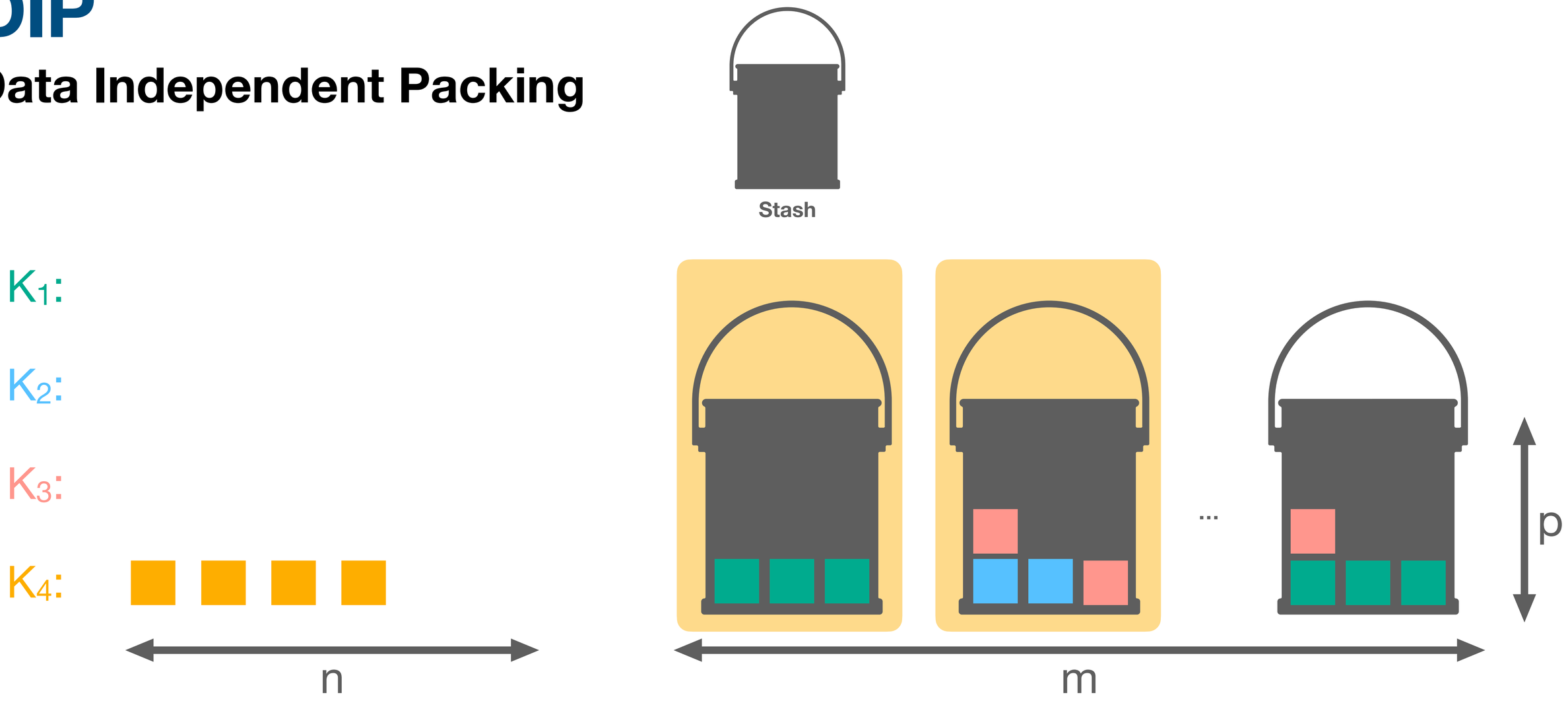
DIP

Data Independent Packing



DIP

Data Independent Packing



DIP

Definition

Size(n):

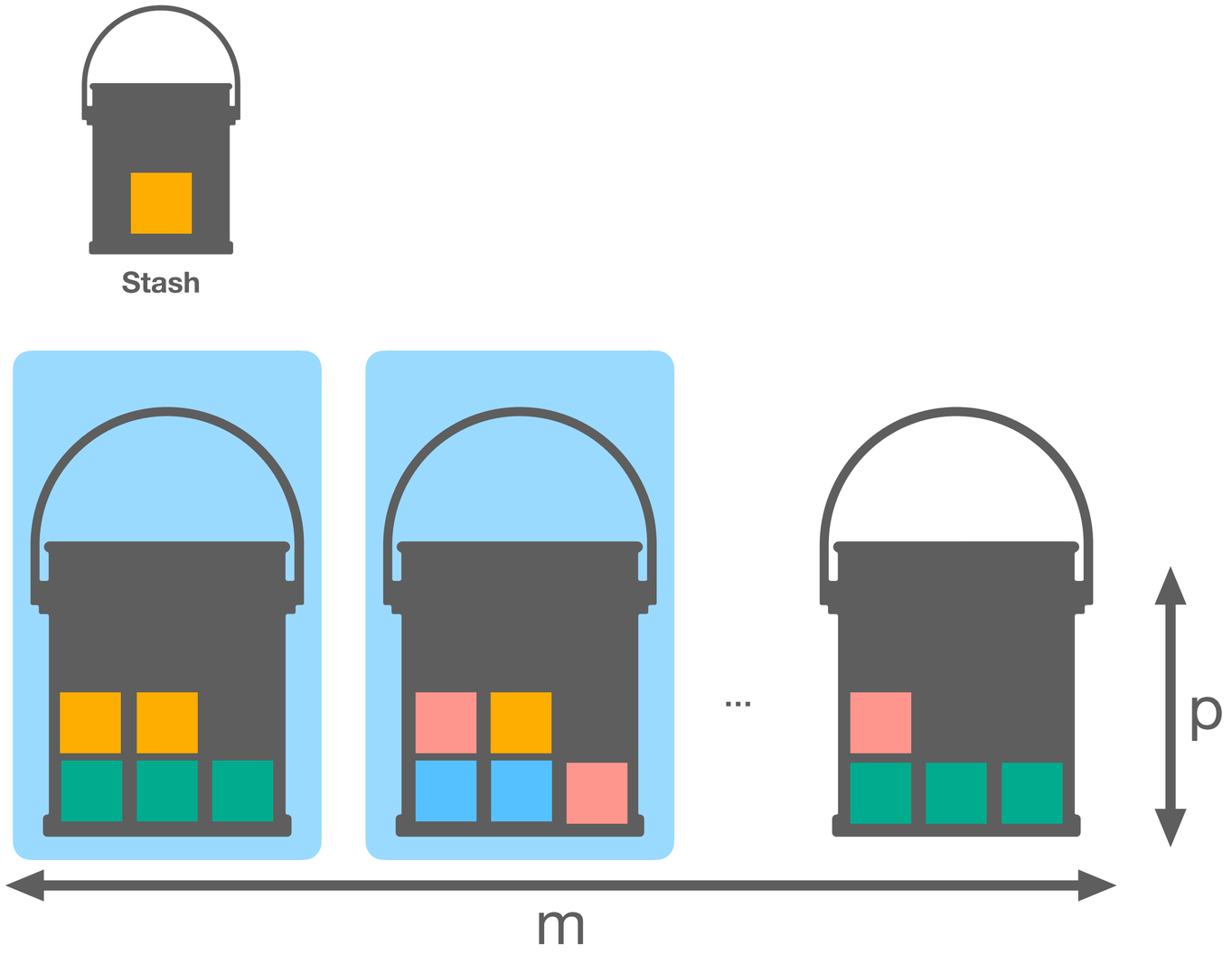
- Return m, the Number of Buckets

Build(M):

- Returns Buckets filled with Items from Multi-Map M
- Allows for Stash S

Lookup(m, K_i, ℓ_i):

- Return Bucket Indices



DIP

Efficiency

Lookup Efficiency:

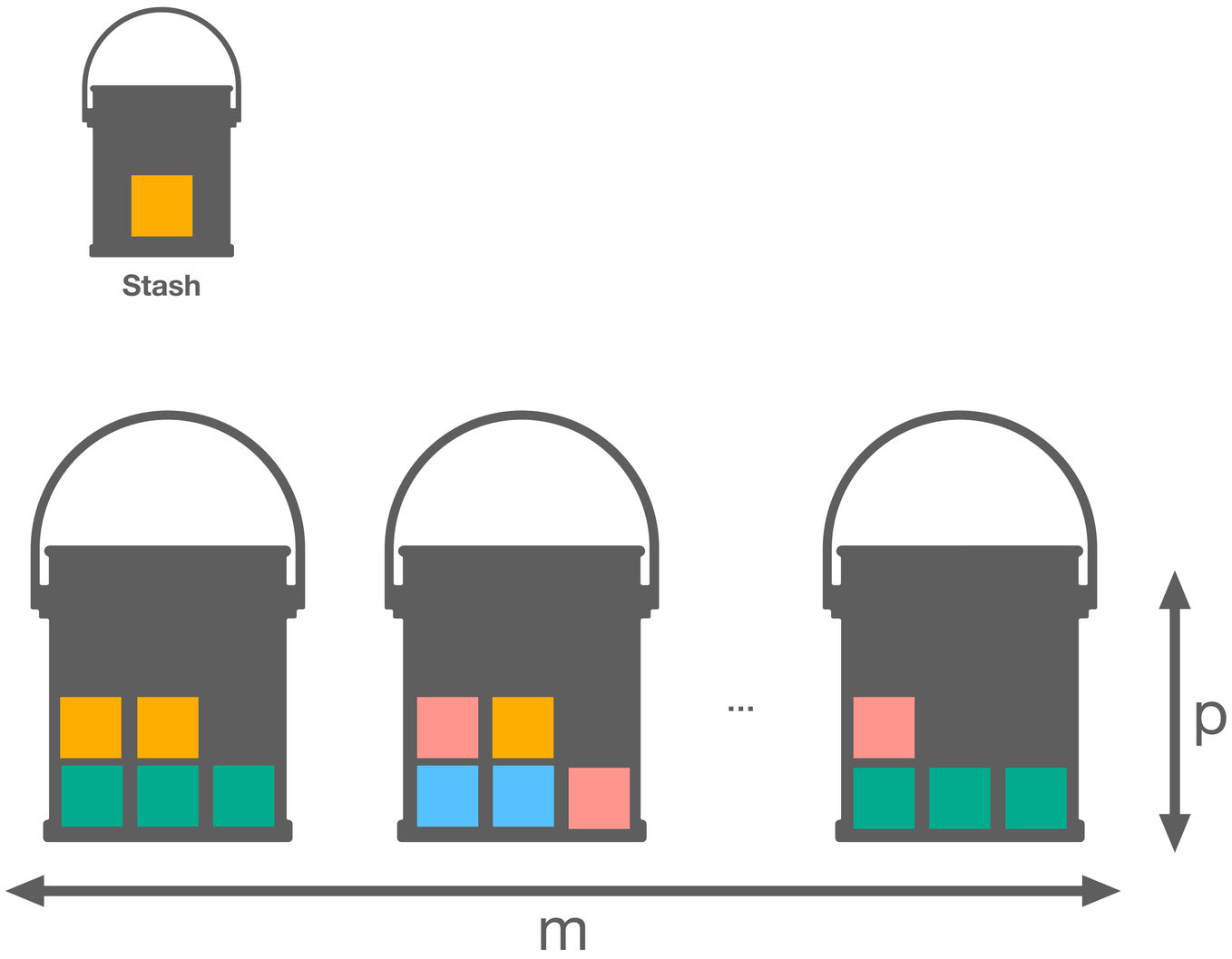
- number of buckets per keyword

Storage Efficiency:

- (m•p) / n

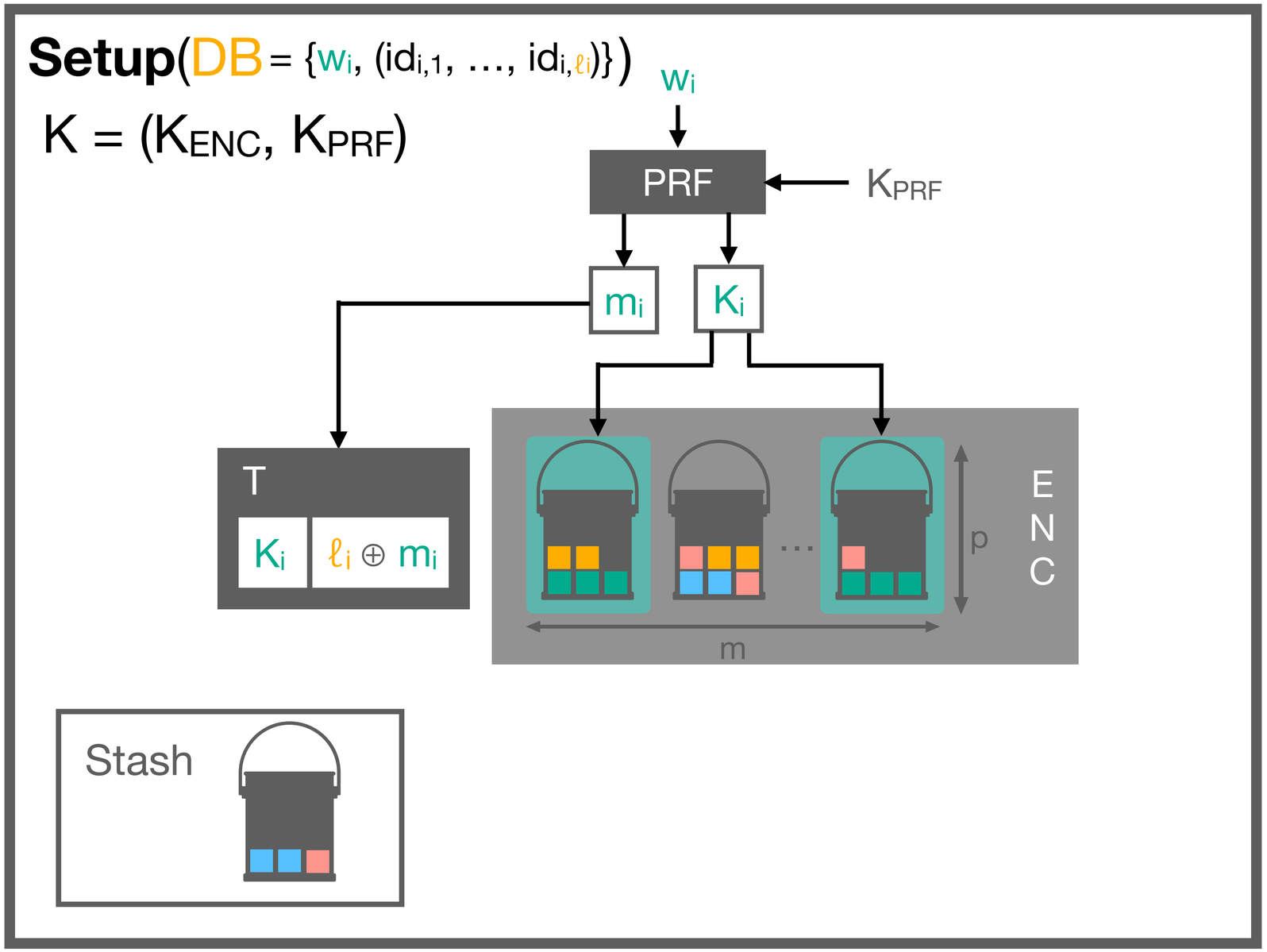
Stash Size:

- Size of Stash S



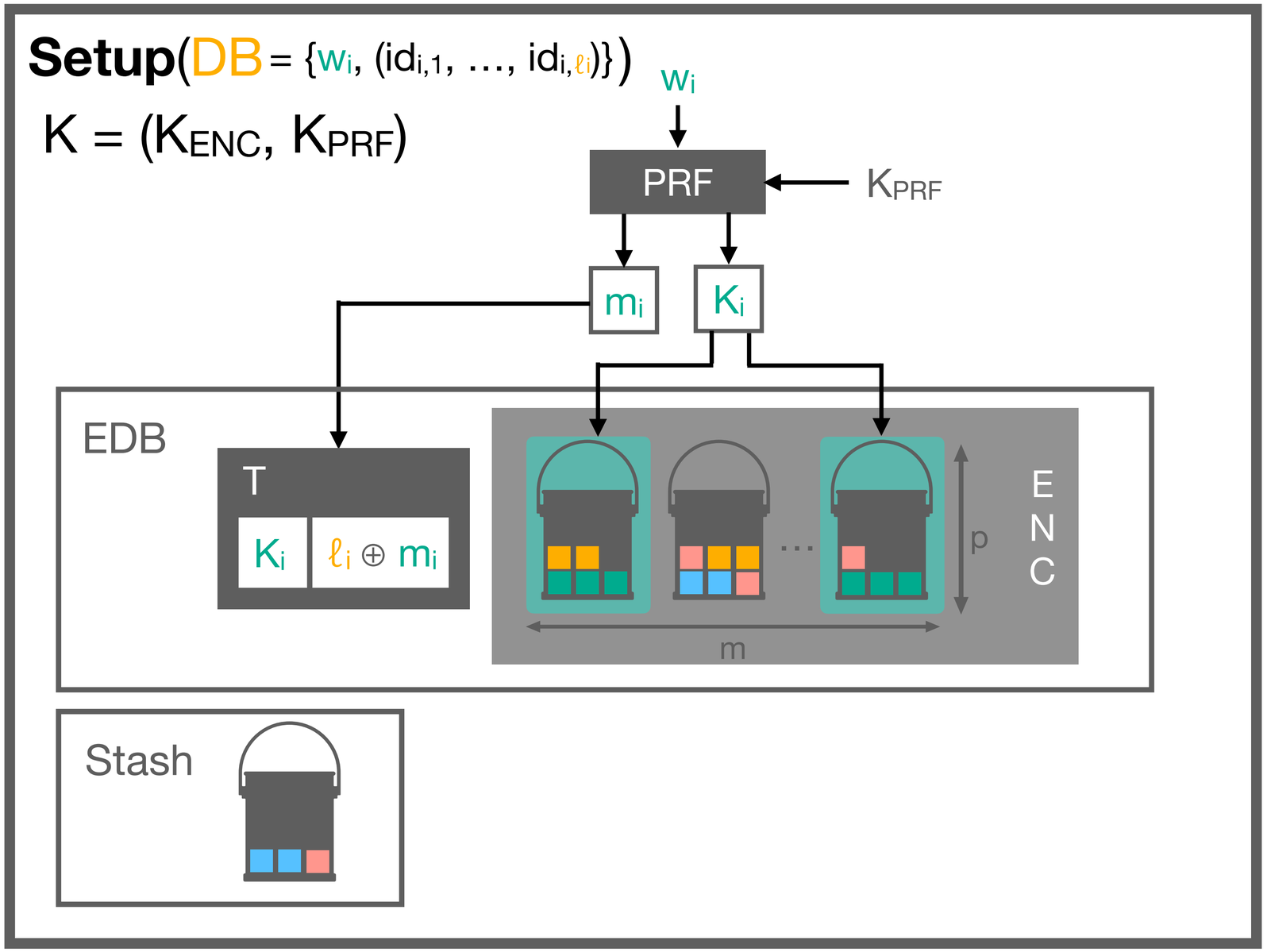
DIP ⇒ SSE

Framework



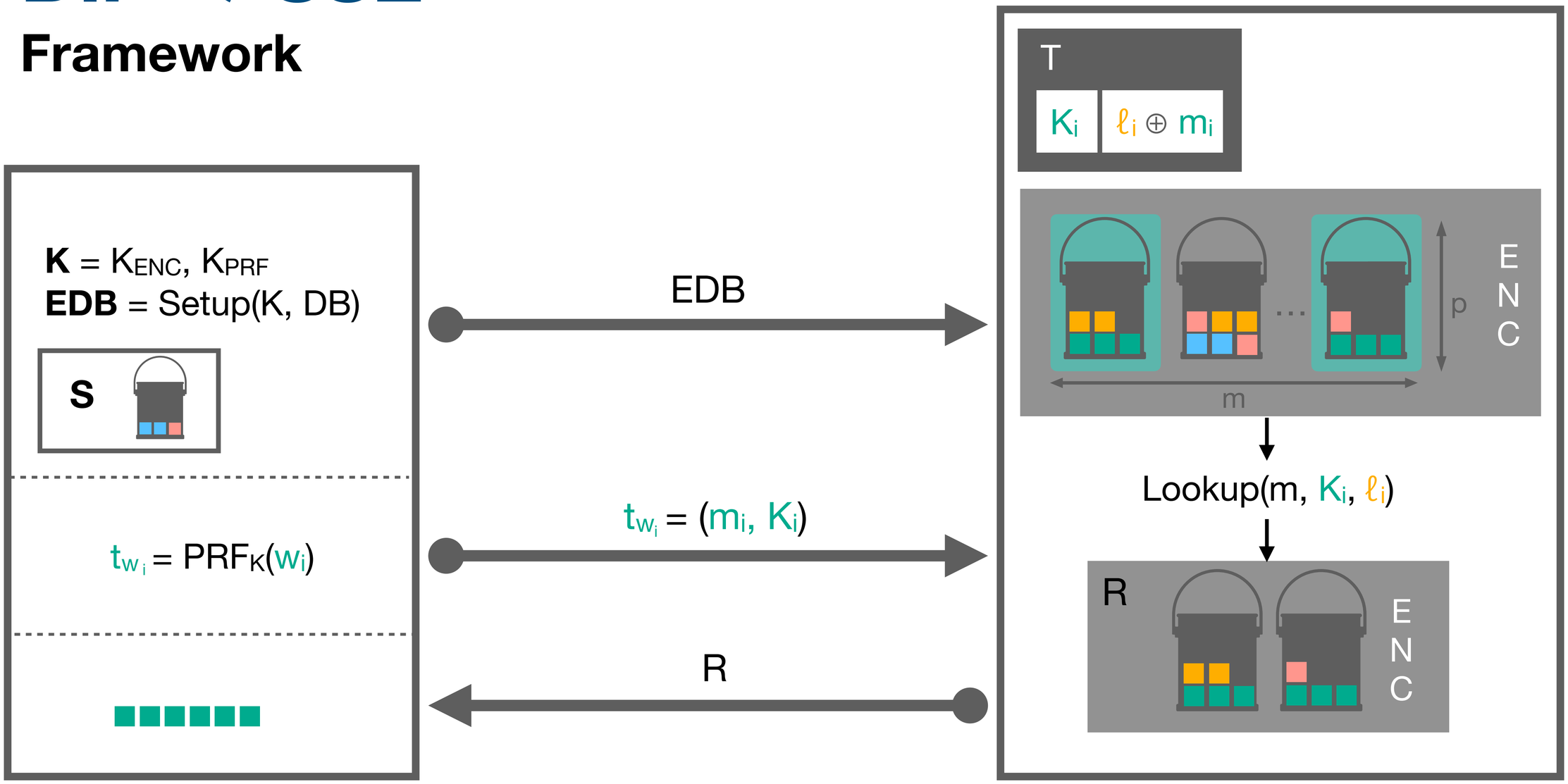
DIP ⇒ SSE

Framework



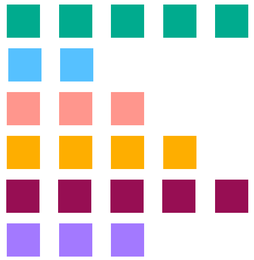
DIP ⇒ SSE

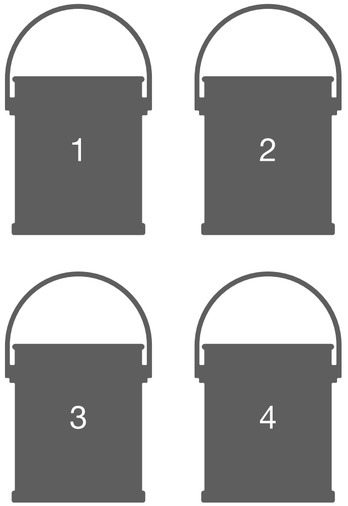
Framework



Tethys

Efficient DIP





p = 5

1

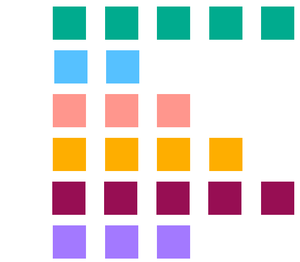
2

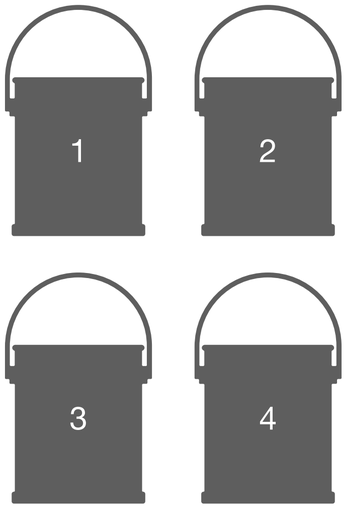
4

3

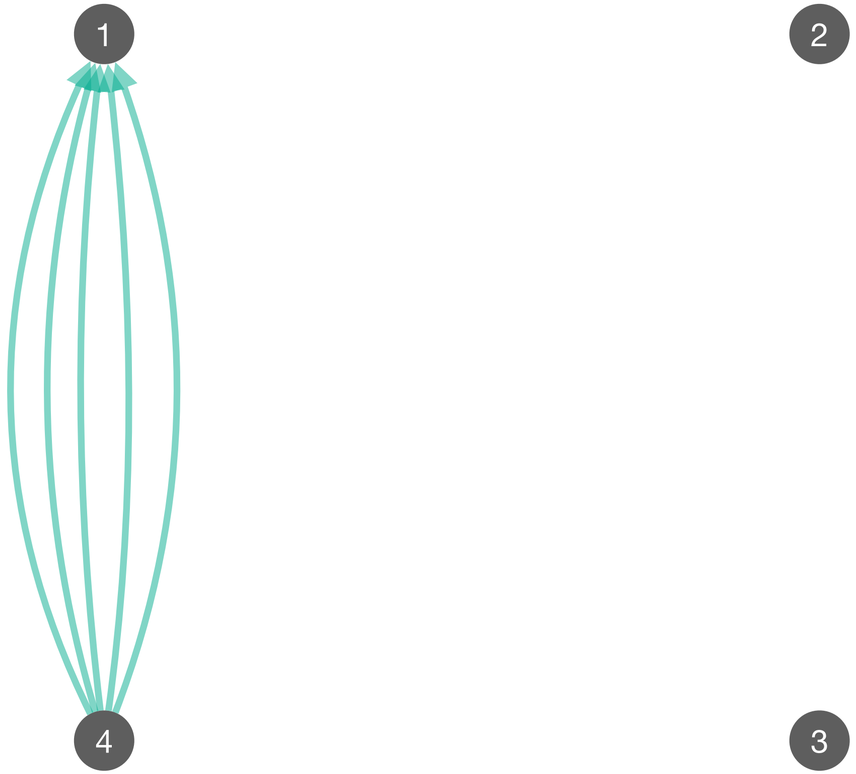
Tethys

Efficient DIP



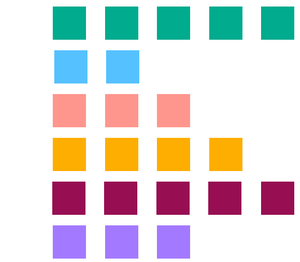


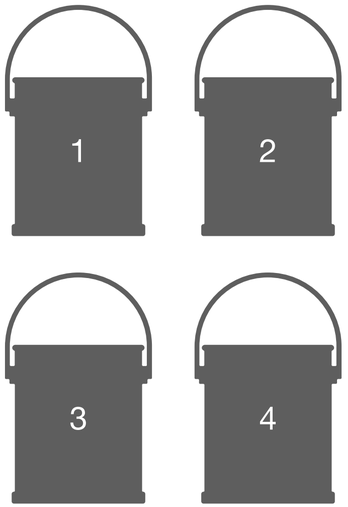
p = 5



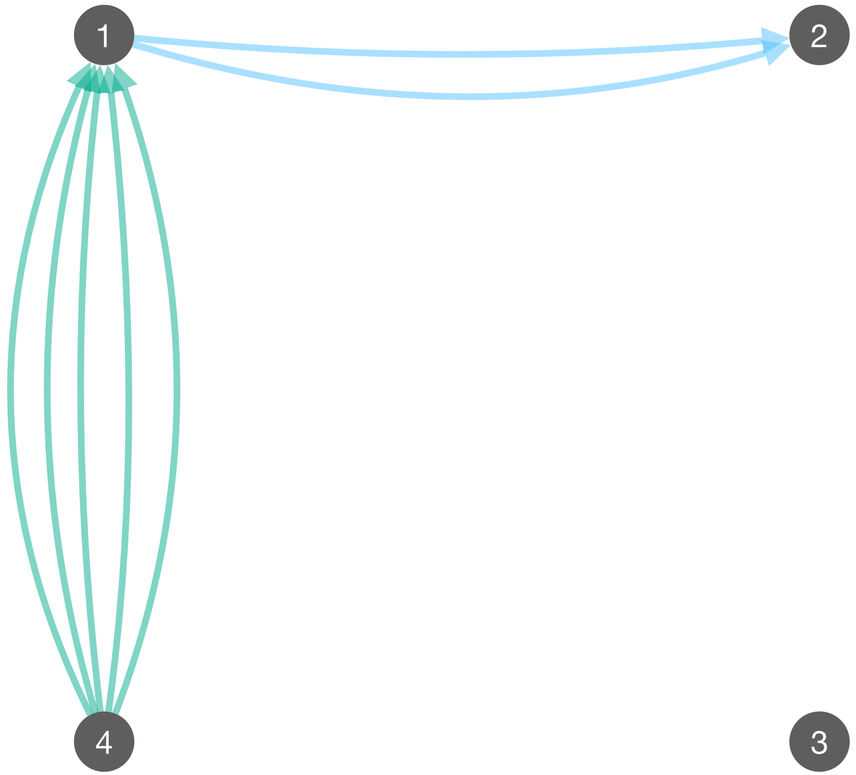
Tethys

Efficient DIP



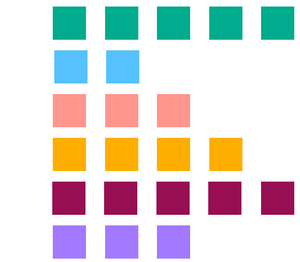


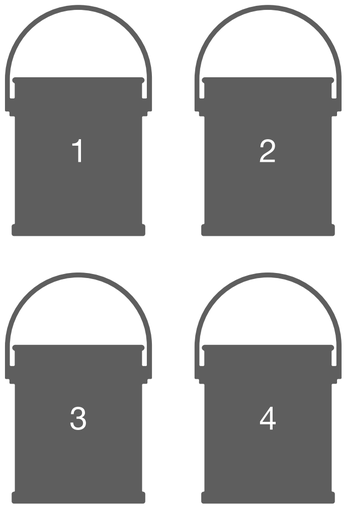
p = 5



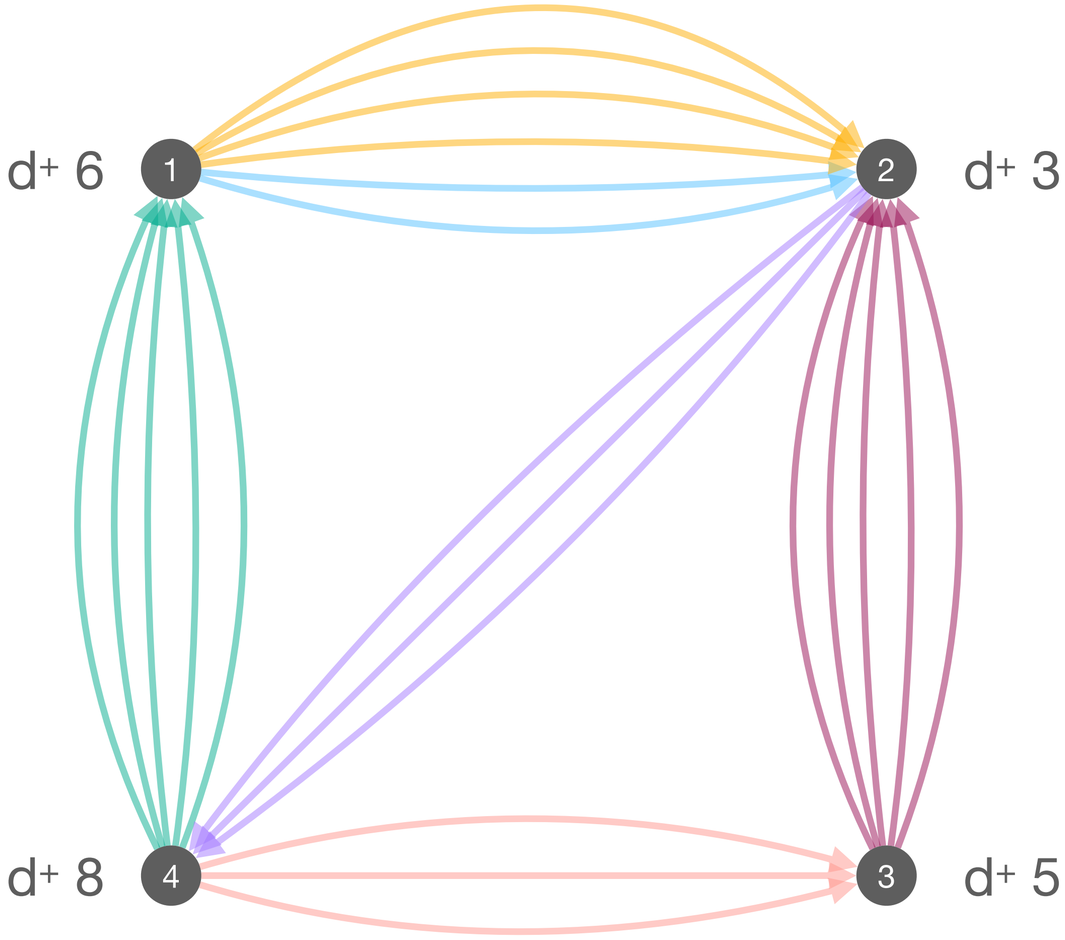
Tethys

Efficient DIP



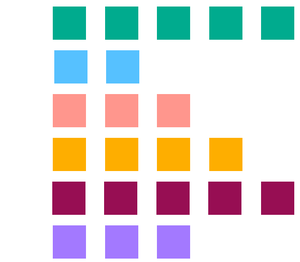


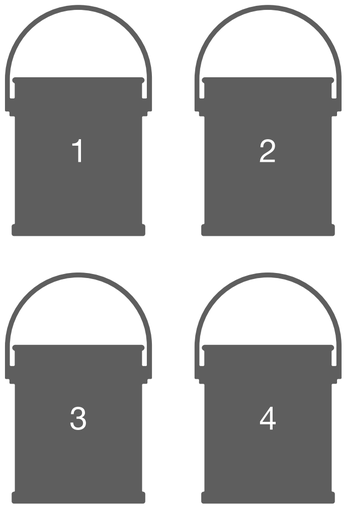
p = 5



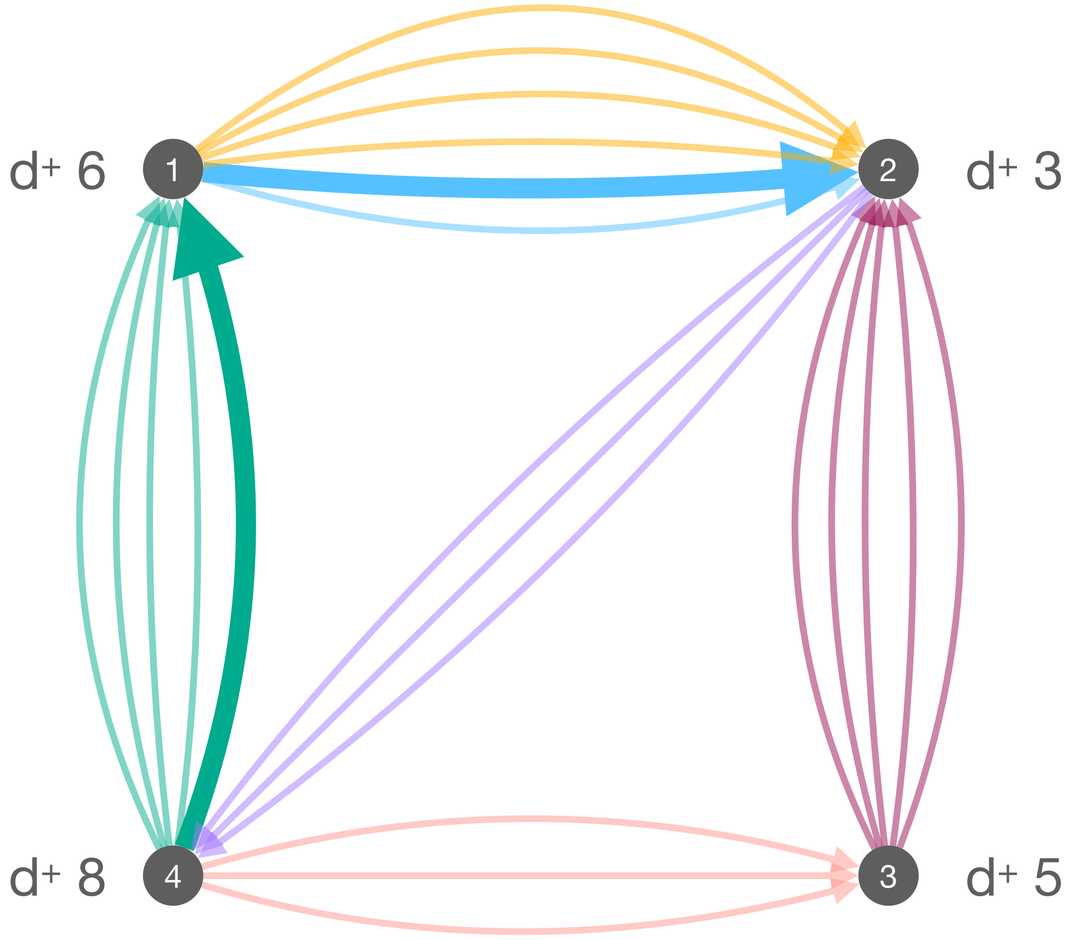
Tethys

Efficient DIP



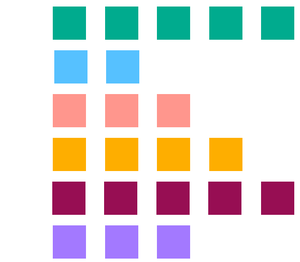


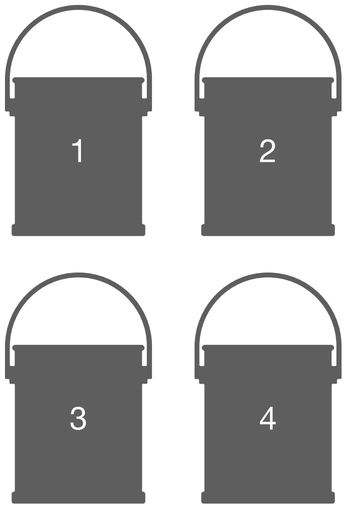
p = 5



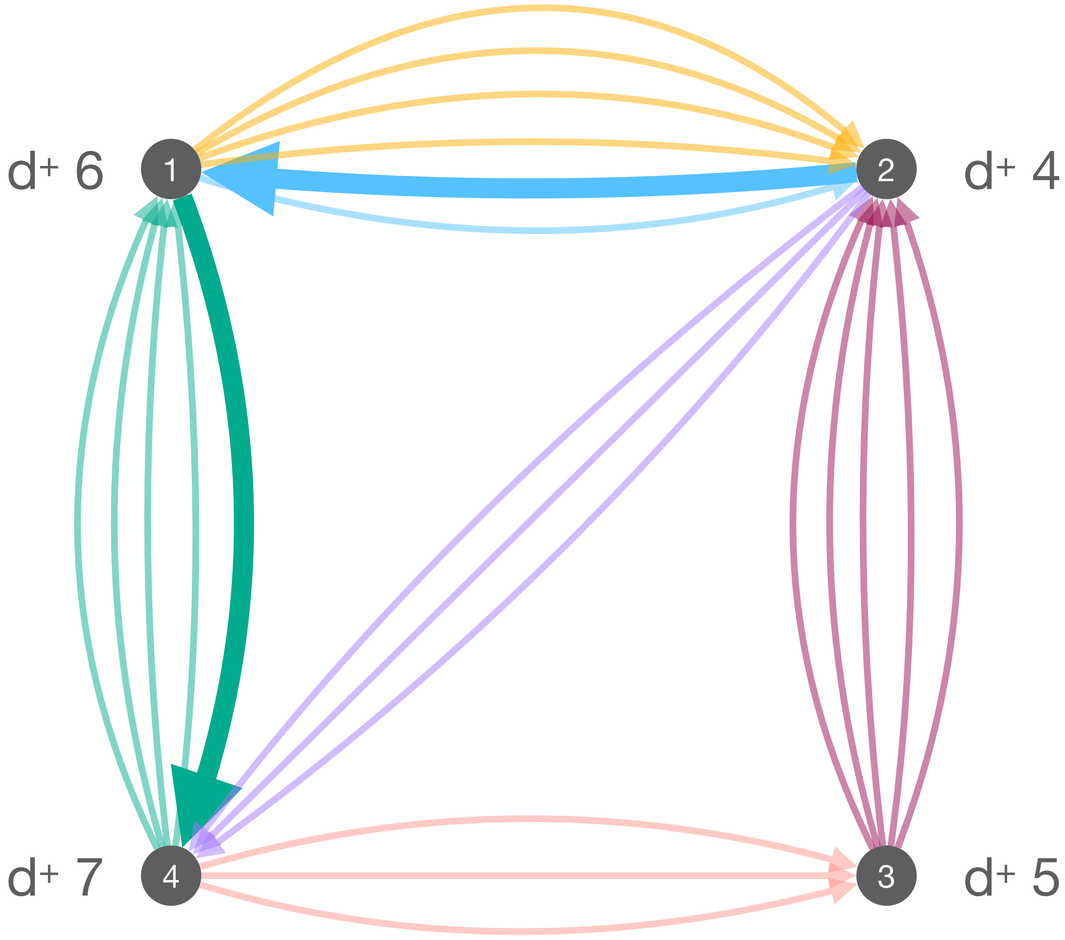
Tethys

Efficient DIP



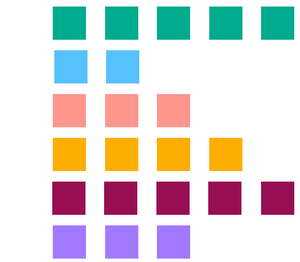


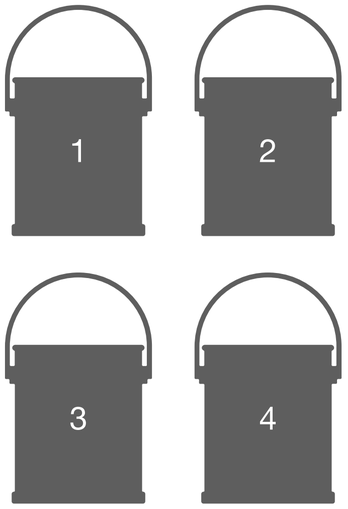
p = 5



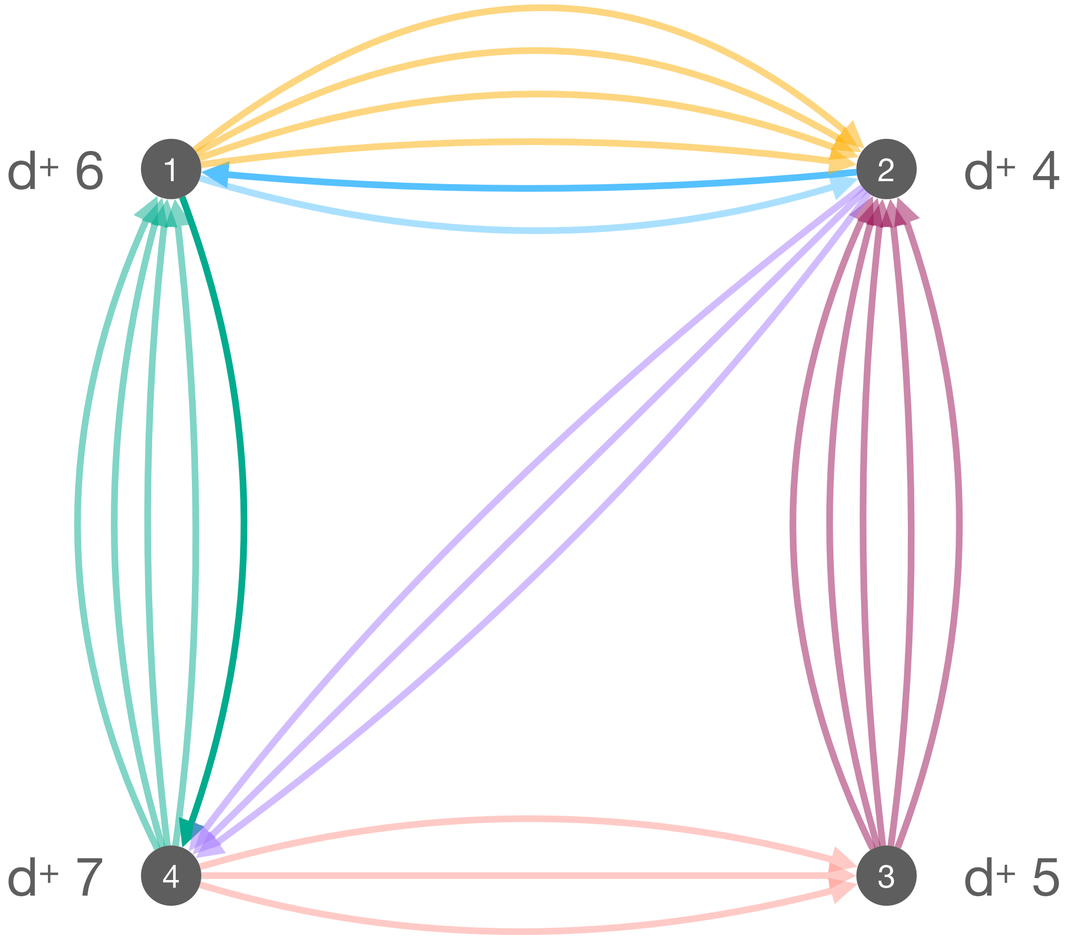
Tethys

Efficient DIP



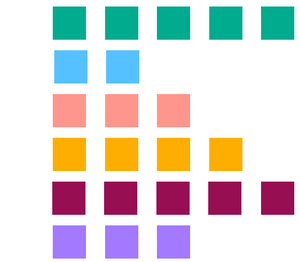


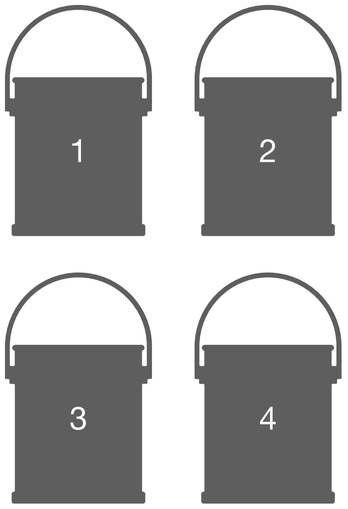
p = 5



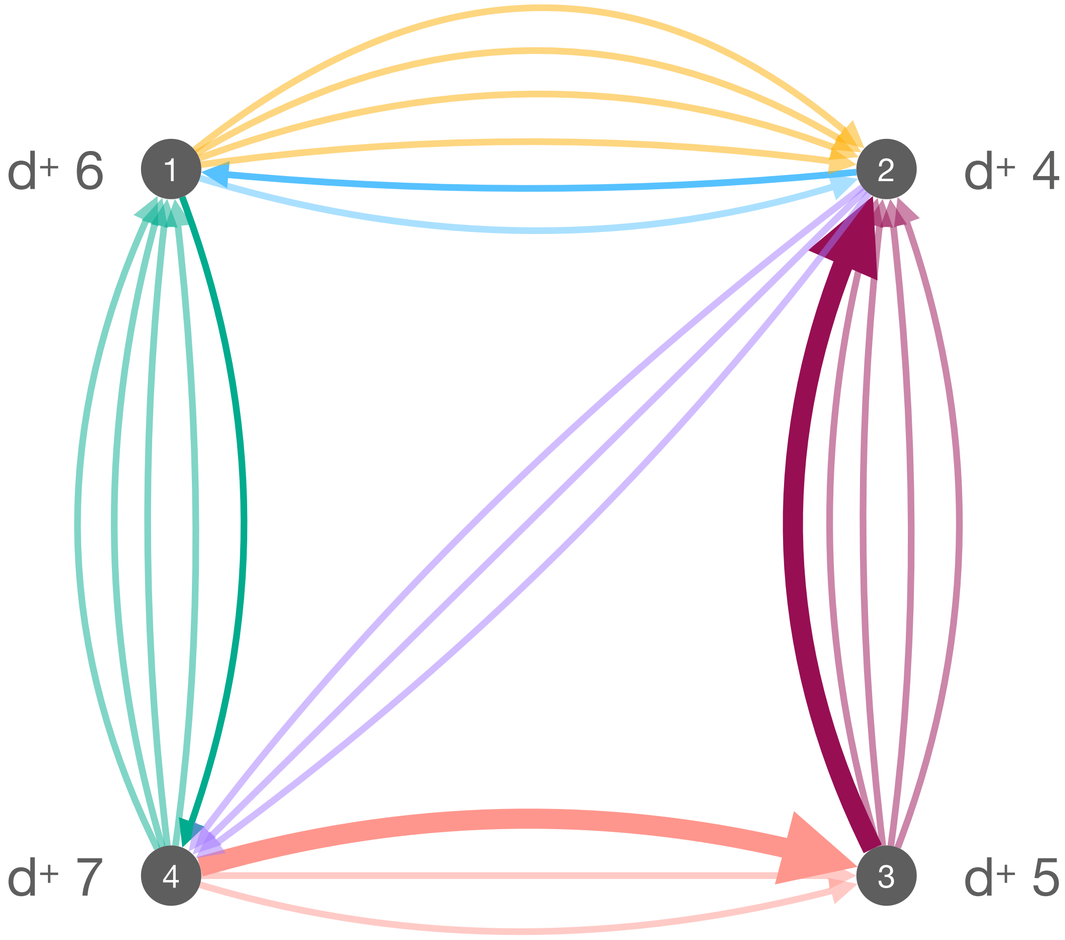
Tethys

Efficient DIP



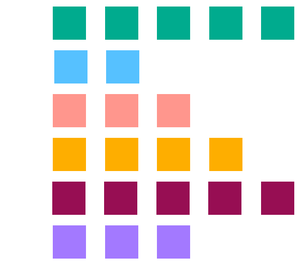


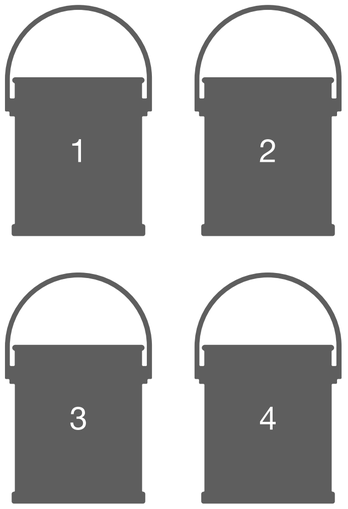
p = 5



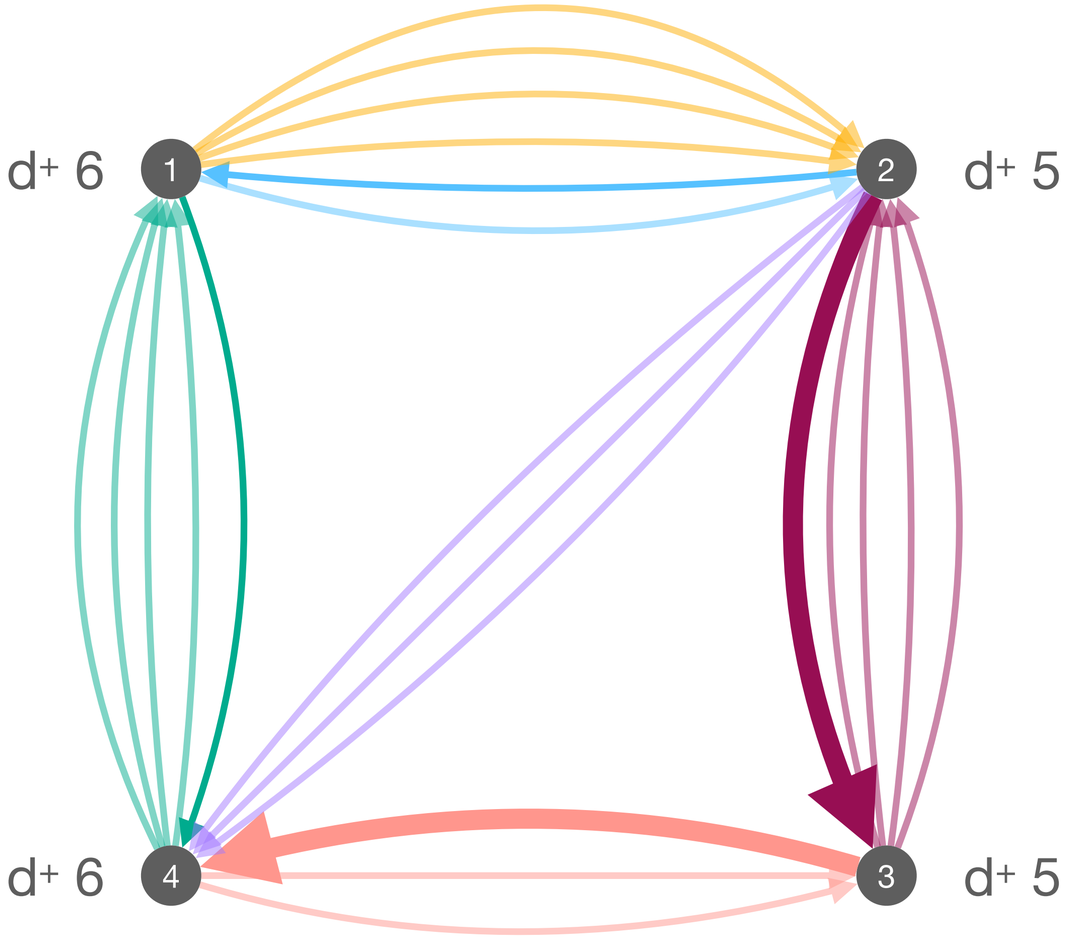
Tethys

Efficient DIP



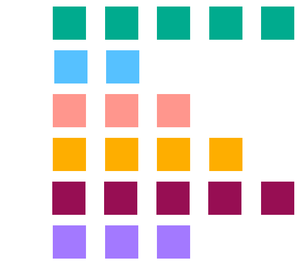


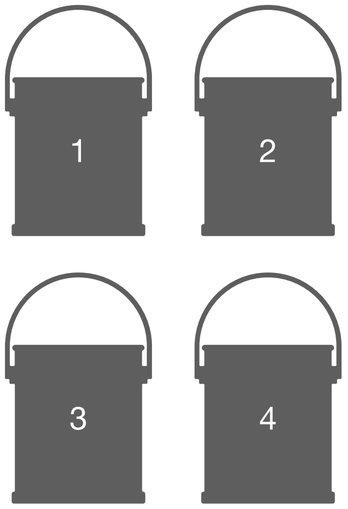
p = 5



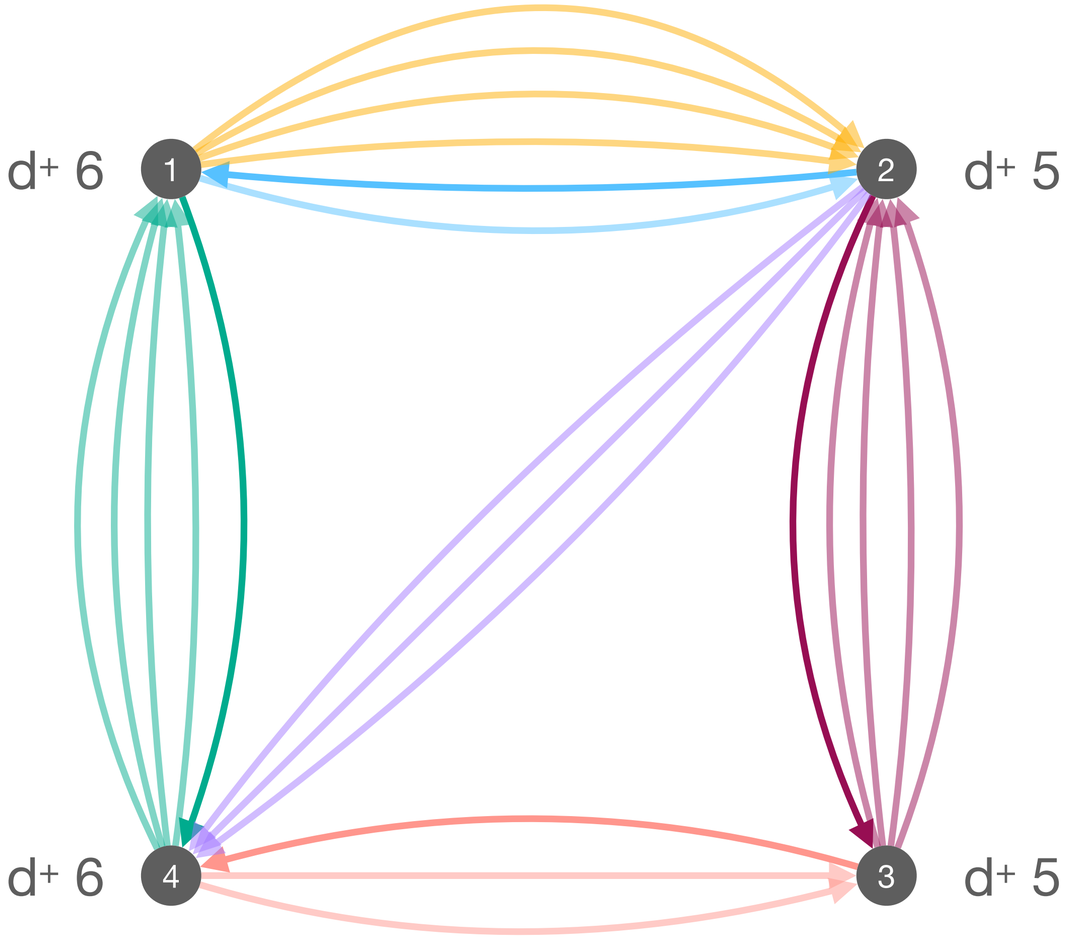
Tethys

Efficient DIP



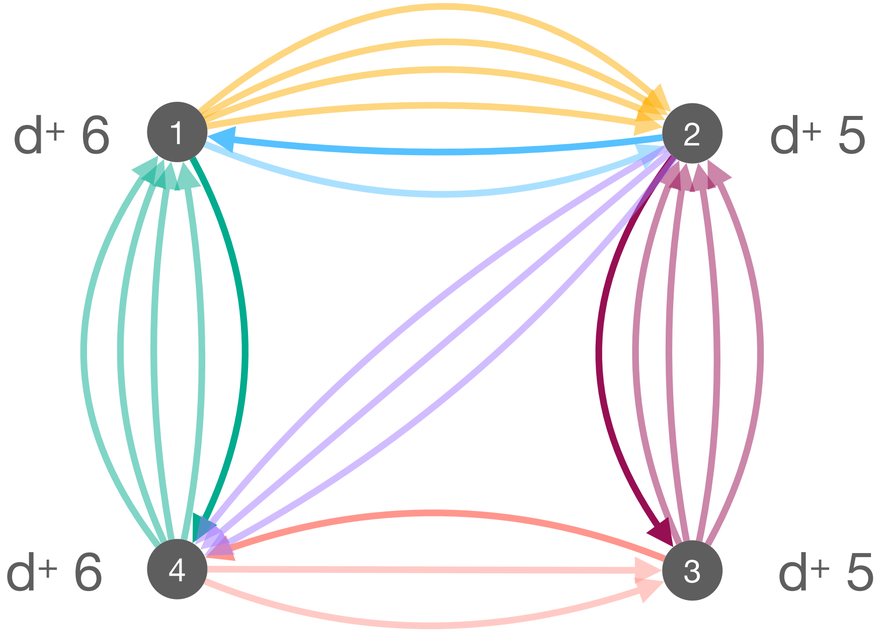


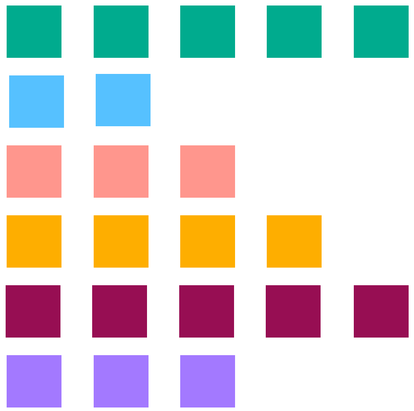
p = 5

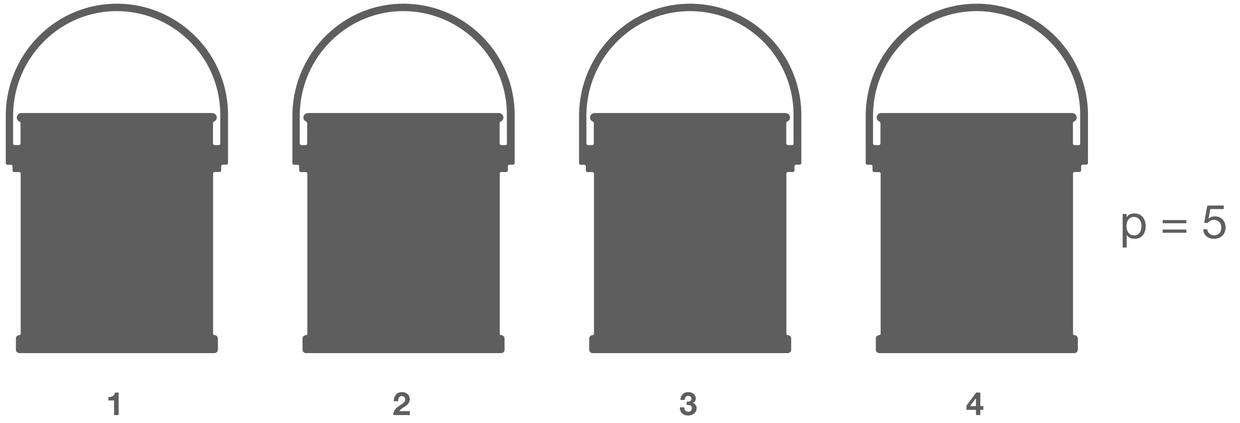


Tethys

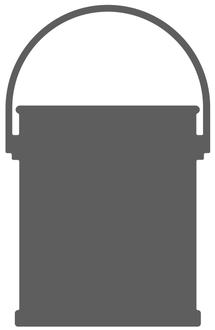
Efficient DIP





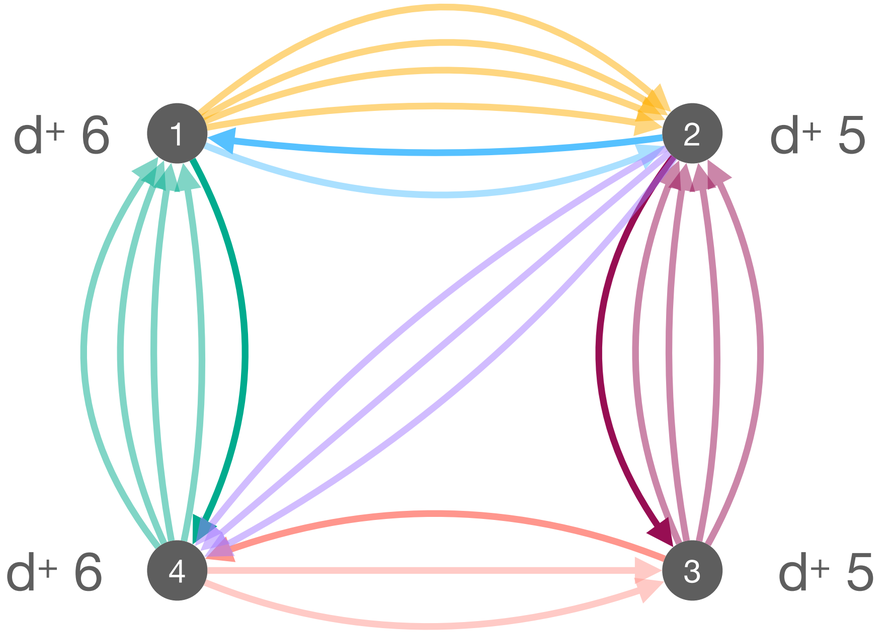


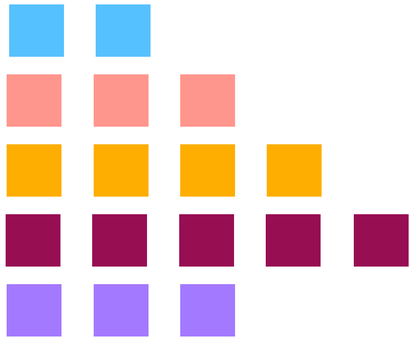
p = 5

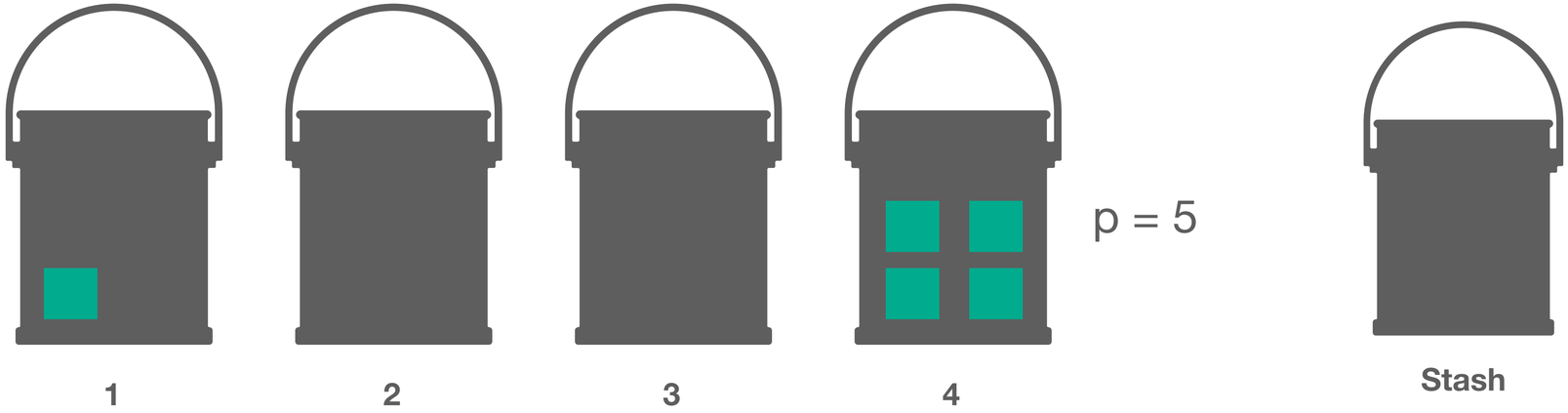


Tethys

Efficient DIP

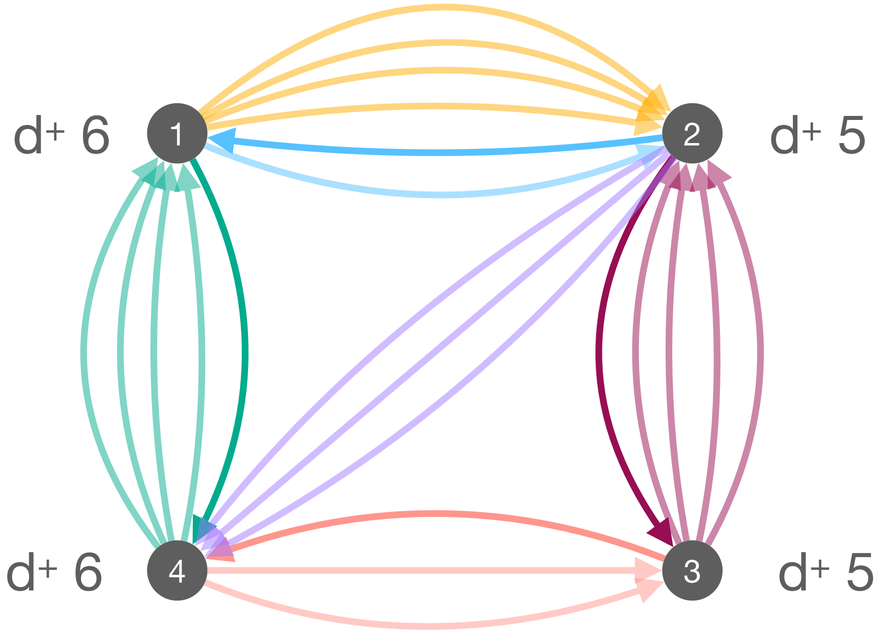


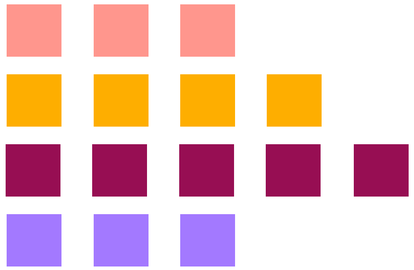


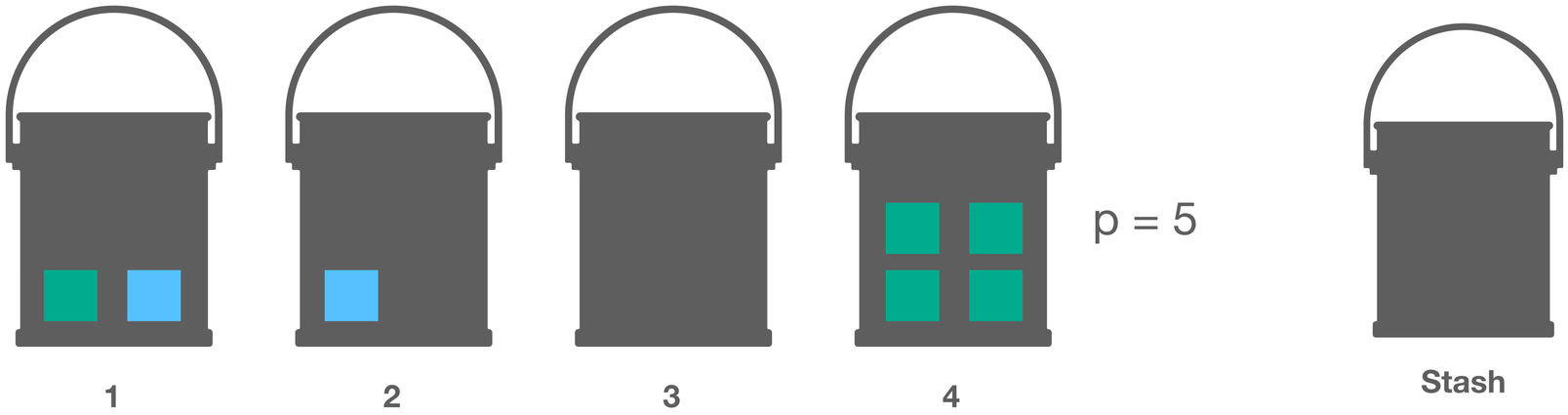


Tethys

Efficient DIP

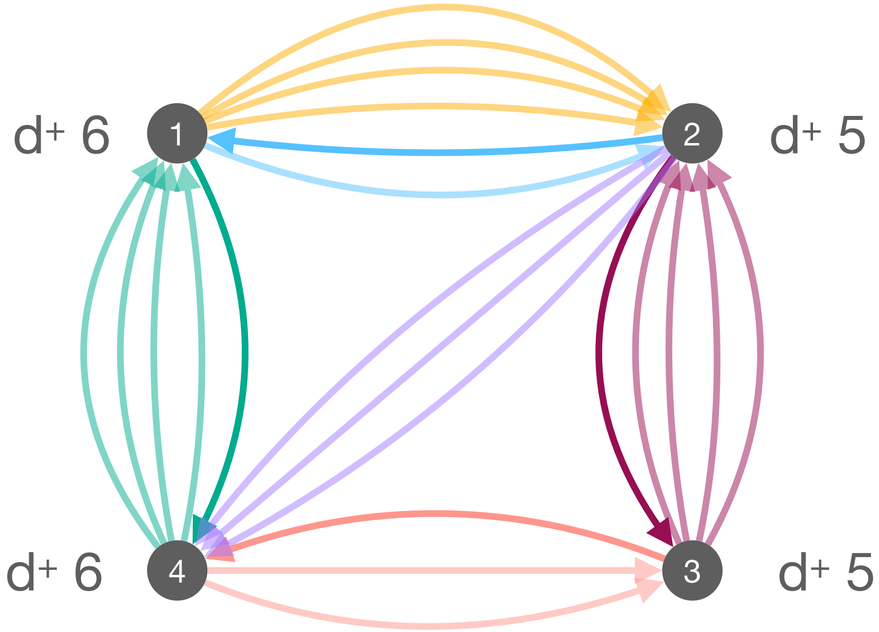


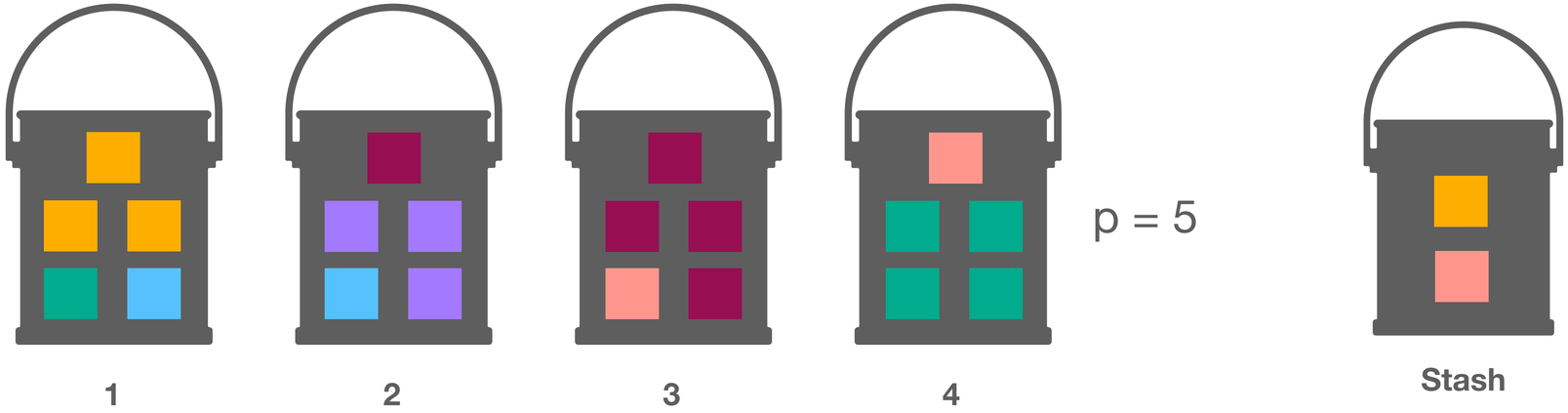




Tethys

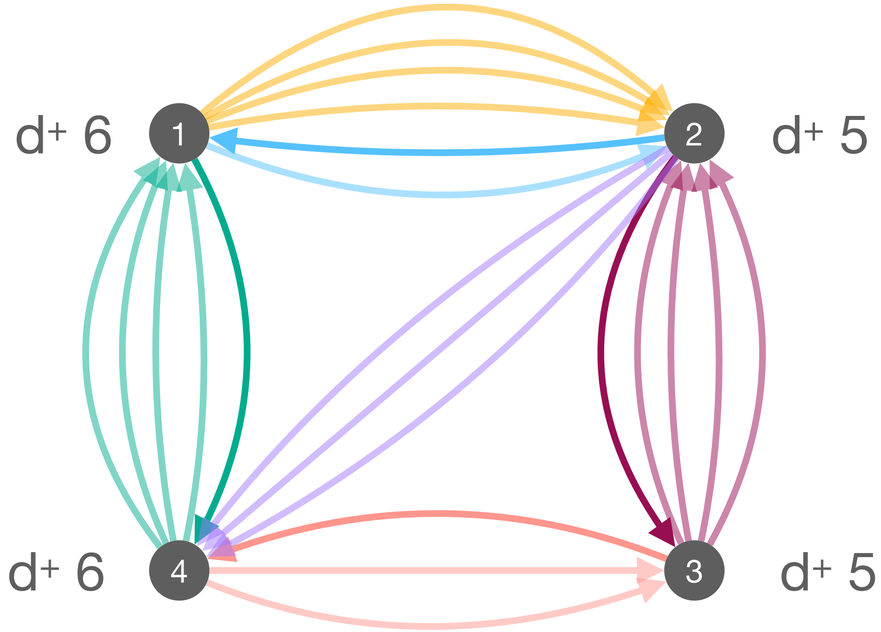
Efficient DIP

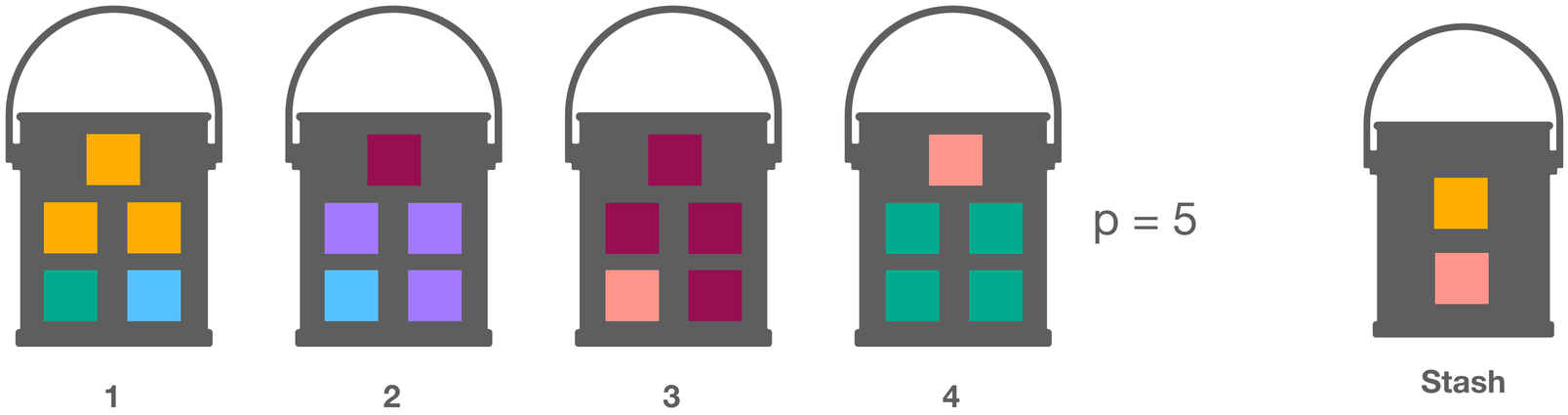




Tethys

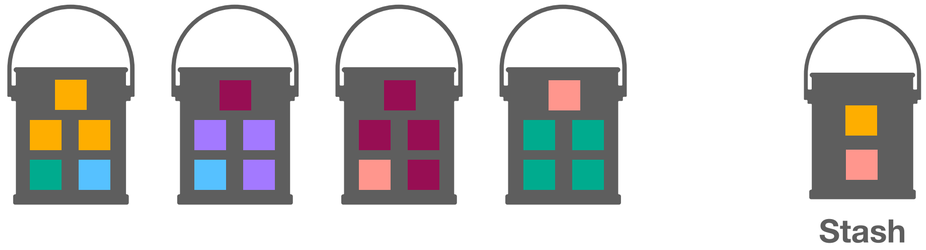
Efficient DIP





Tethys

Efficient DIP



- Main Theorem:
  - There Exists a Valid Assignment w.o.p. such that:
    - **m** = (2 + ε)n/p
    - **Stash**: ω(logλ)/log(n) pages
  - Direct Generalization of Cuckoo-Hashing in the Static Case

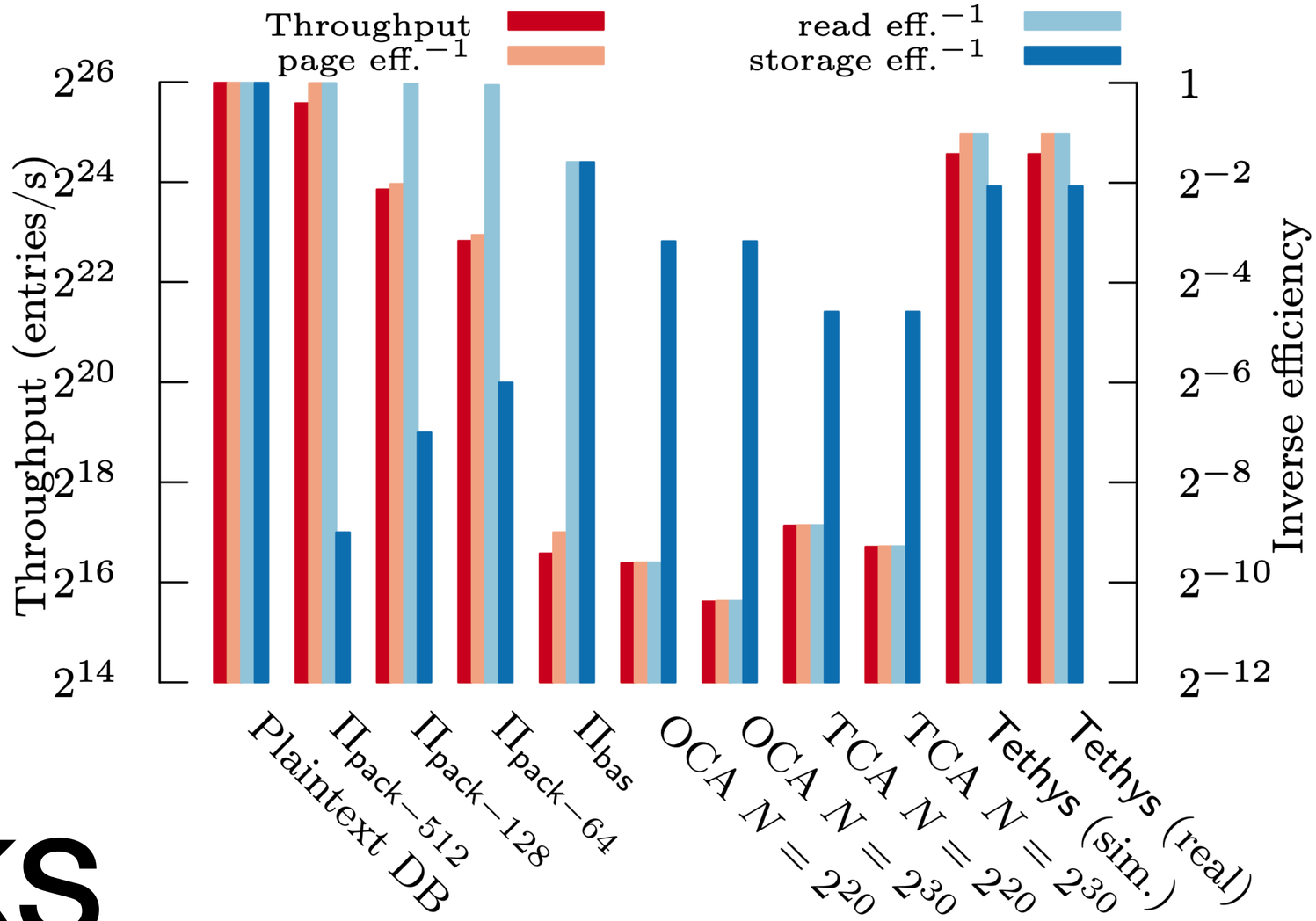
Conclusion

Efficiency

- Cuckoo-Hashing with Stash for Items of Variable Size
- **DIP**: Data-Independent Packing
- **Tethys**: O(1) Storage + Page Efficiency

Conclusion

Efficiency



THANKS