Coded Caching: Dichotomy of the One and the Many

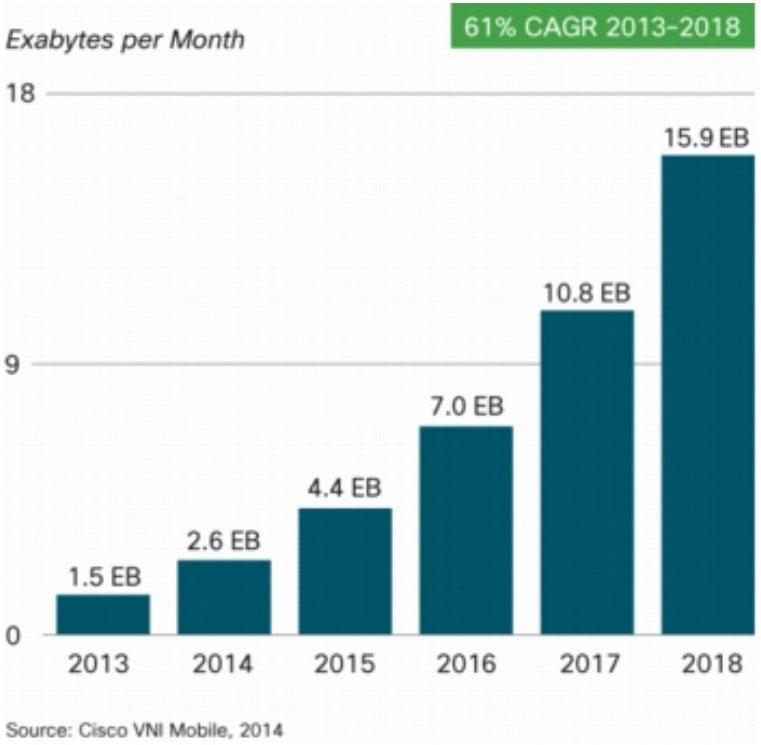
Nikhil Karamchandani

Indian Institute of Technology, Bombay

Joint work with Jad Hachem and Suhas Diggavi, UCLA

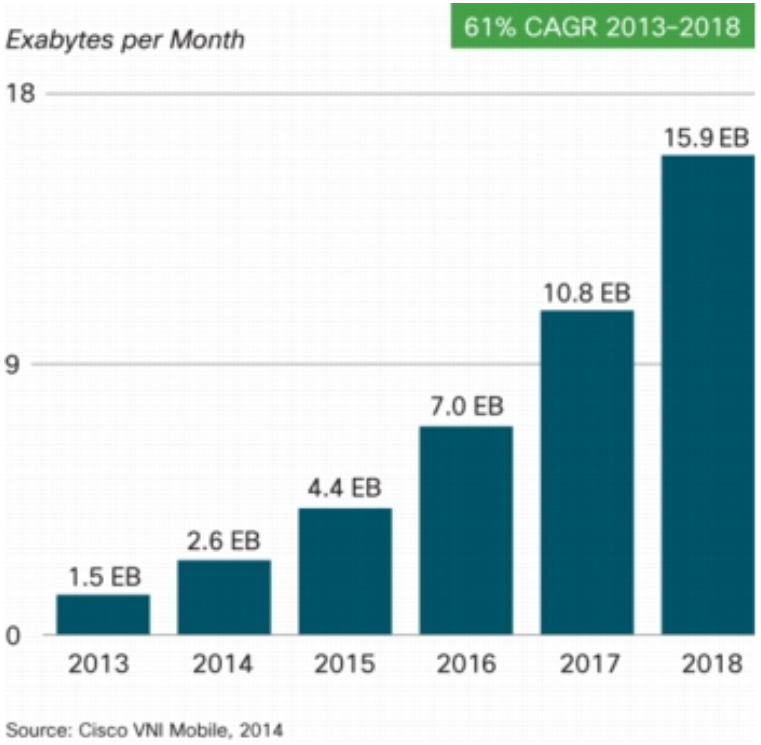
JTG Workshop 2015

Motivation



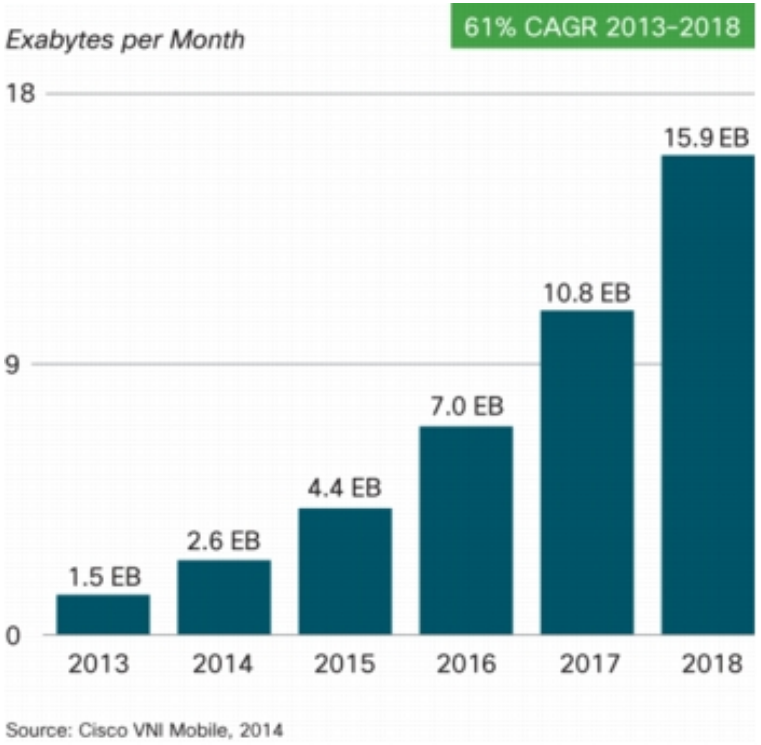
- Multimedia applications fueling increased data consumption

Motivation



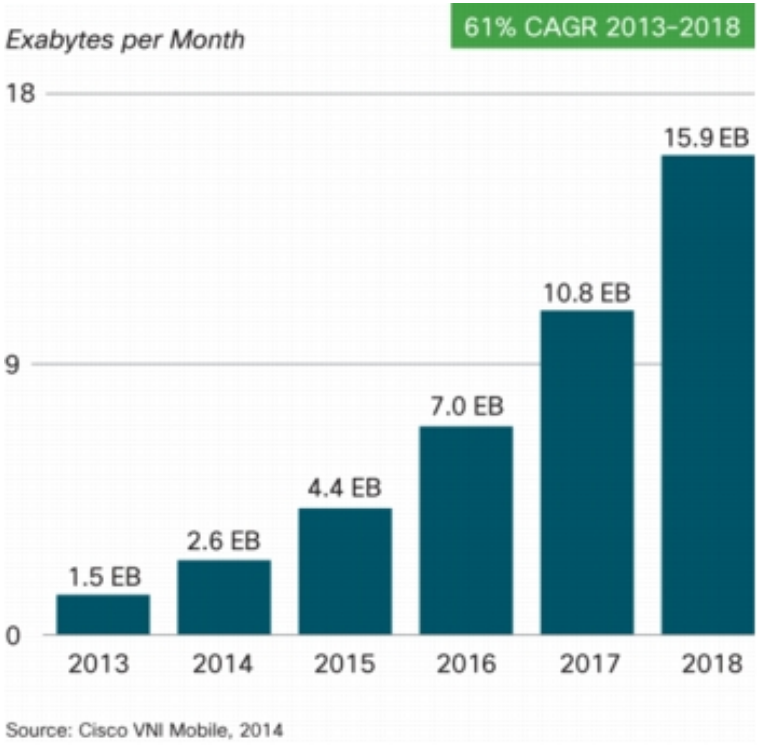
- Multimedia applications fueling increased data consumption
- In-network caching

Motivation



- Multimedia applications fueling increased data consumption
- In-network caching
- Pre-fetch content during off-peak hours

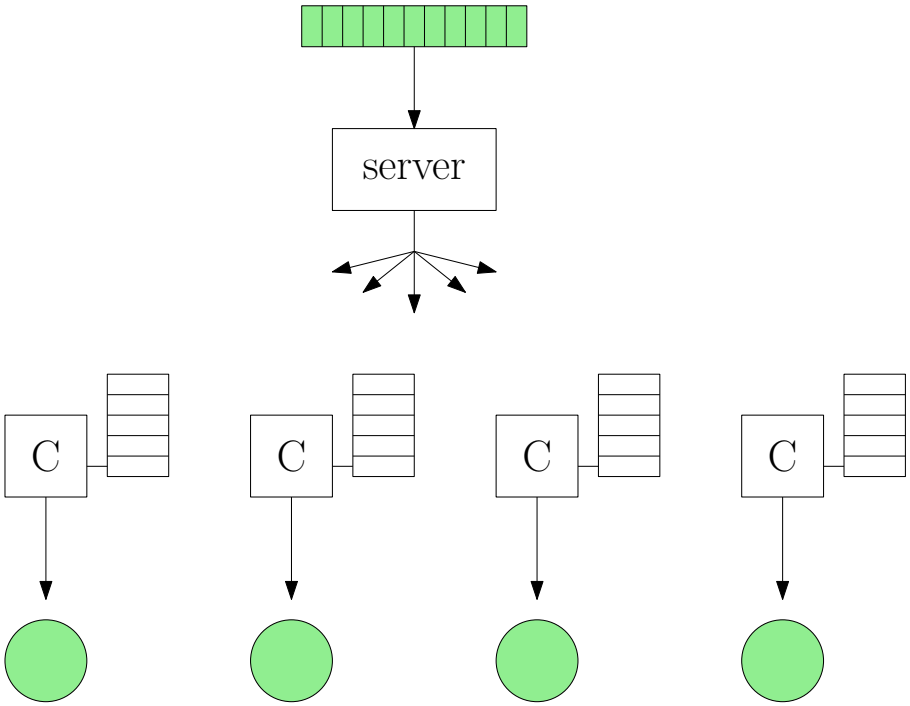
Motivation



- Multimedia applications fueling increased data consumption
- In-network caching
- Pre-fetch content during off-peak hours
- Rate-benefits vs Memory

Effect of number of users in multi-level coded caching

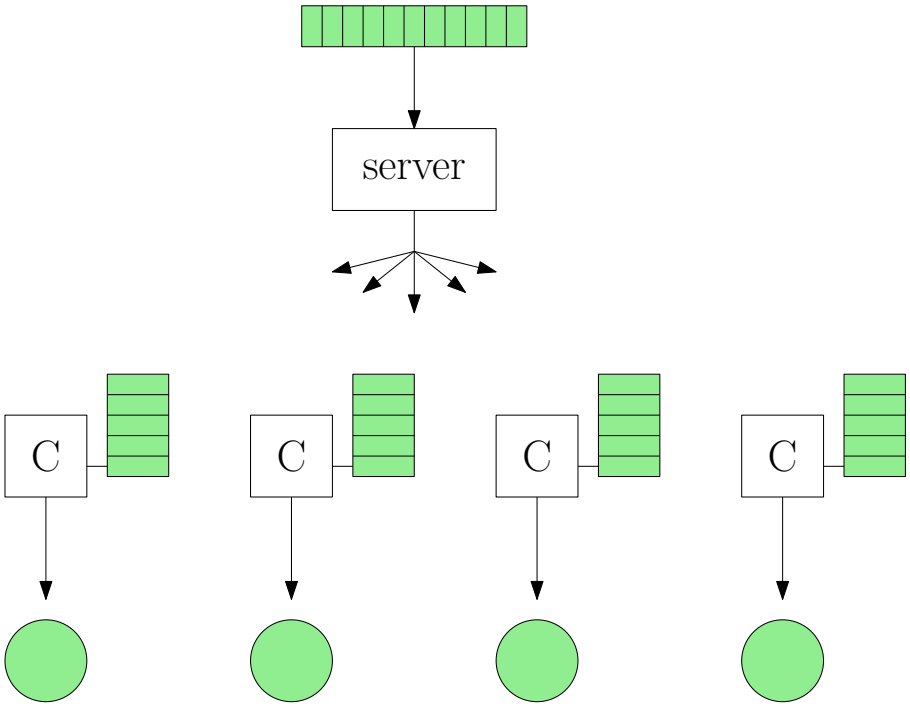
Effect of number of users in multi-level coded caching[1]



- N files, K caches, K users

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

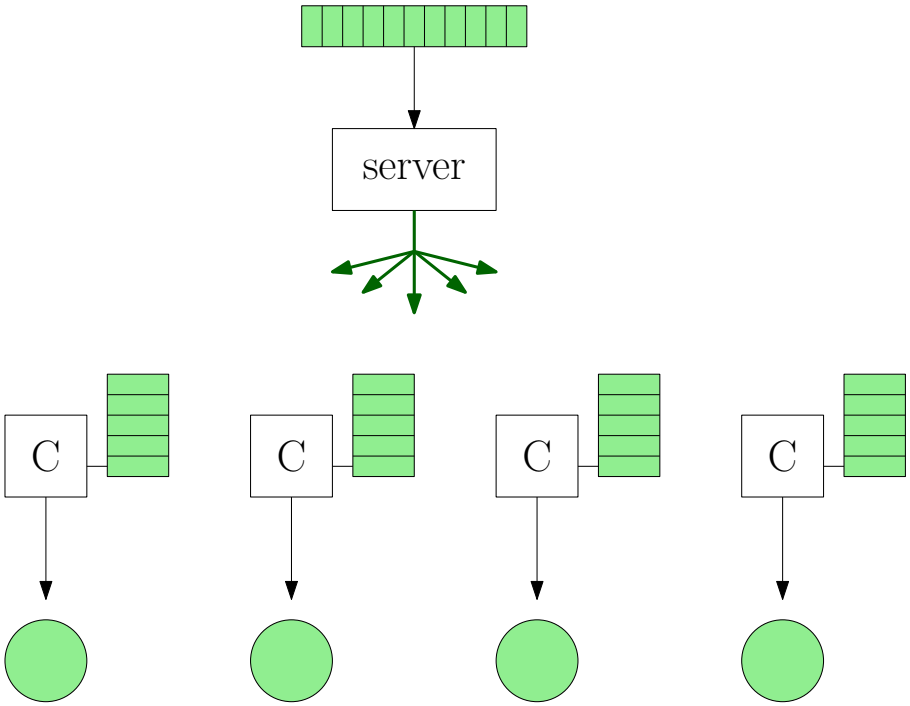
Effect of number of users in multi-level coded caching[1]



- N files, K caches, K users
- **Placement**
  - Place content in caches
  - Done without prior knowledge of user requests

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

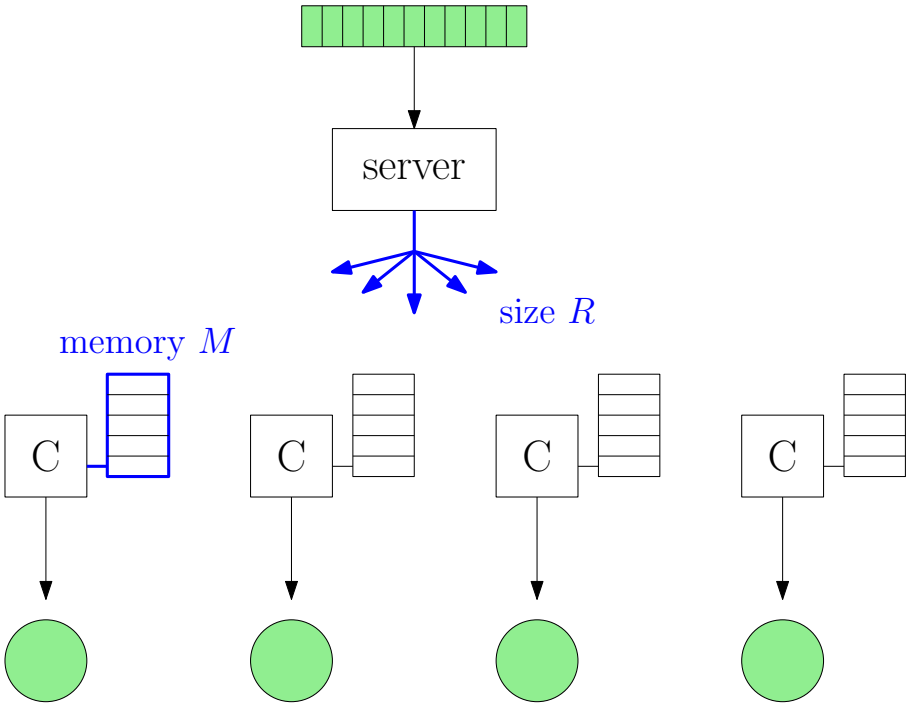
Effect of number of users in multi-level coded caching[1]



- N files, K caches, K users
- **Placement**
  - Place content in caches
  - Done without prior knowledge of user requests
- **Delivery**
  - Each user requests a file
  - Server assists in delivery

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

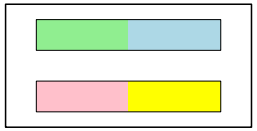
Effect of number of users in multi-level coded caching[1]



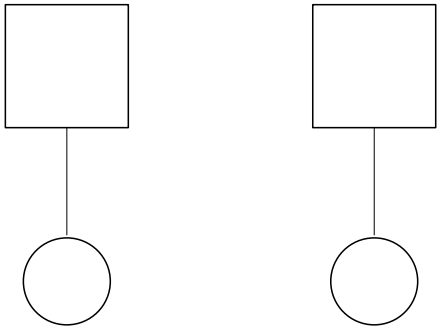
- N files, K caches, K users
- **Placement**
  - Place content in caches
  - Done without prior knowledge of user requests
- **Delivery**
  - Each user requests a file
  - Server assists in delivery
- Given memory M, smallest rate R?

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Uncoded caching example

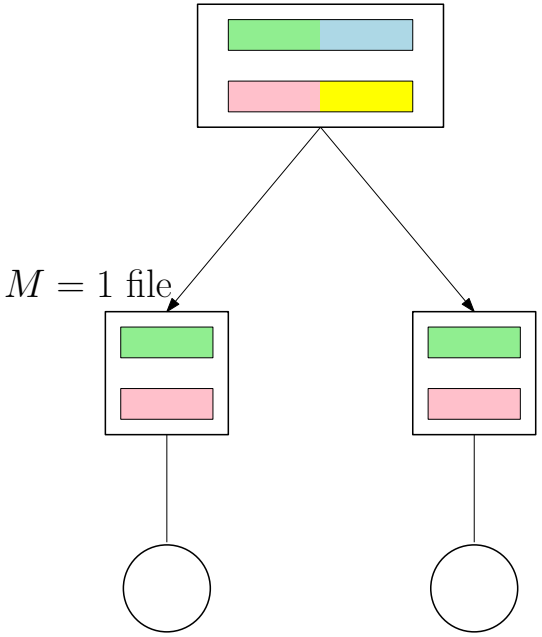


$M = 1$ file



- $N = 2$ files, $K = 2$ caches

Uncoded caching example

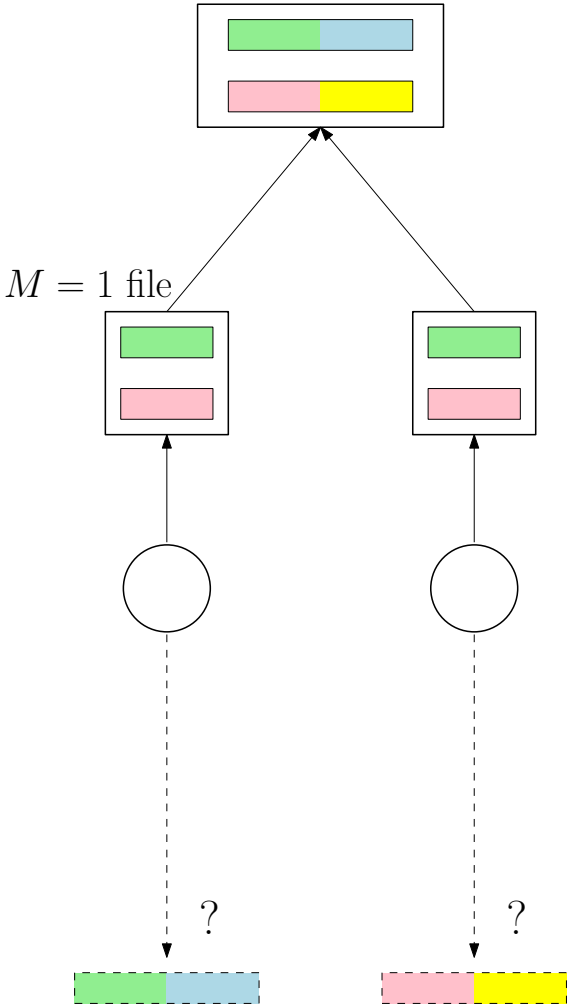


- $N = 2$ files, $K = 2$ caches

Placement phase

- Place first half of each file in each cache

Uncoded caching example



- $N = 2$ files, $K = 2$ caches

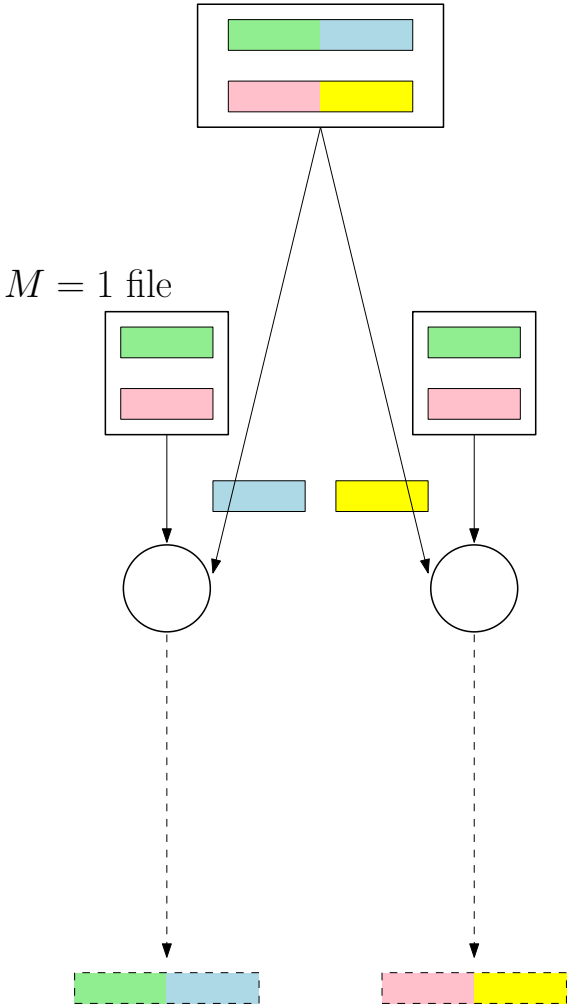
Placement phase

- Place first half of each file in each cache

Delivery phase

- Each user makes their request

Uncoded caching example



- $N = 2$ files, $K = 2$ caches

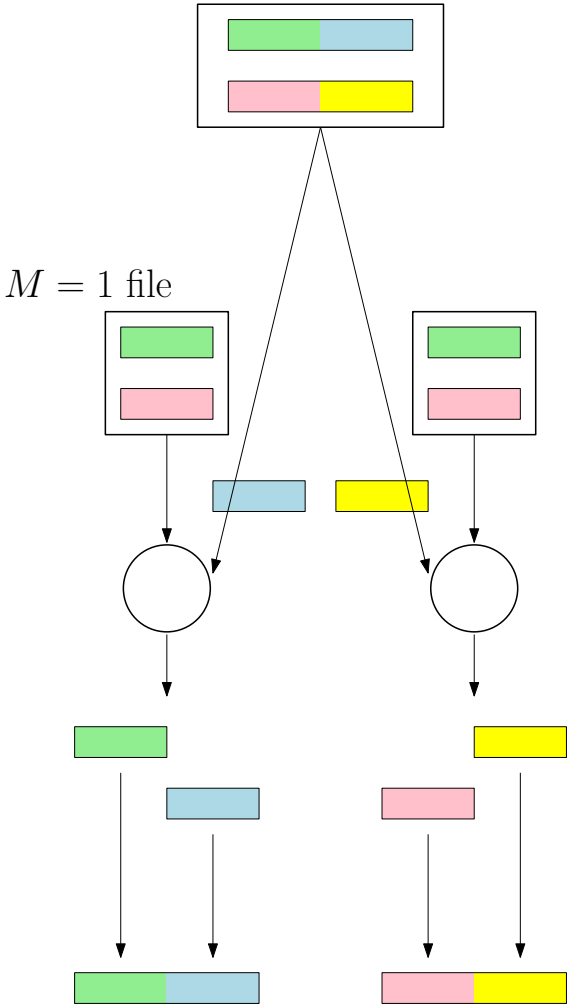
Placement phase

- Place first half of each file in each cache

Delivery phase

- Each user makes their request
- Server unicasts missing piece for each user

Uncoded caching example



- $N = 2$ files, $K = 2$ caches

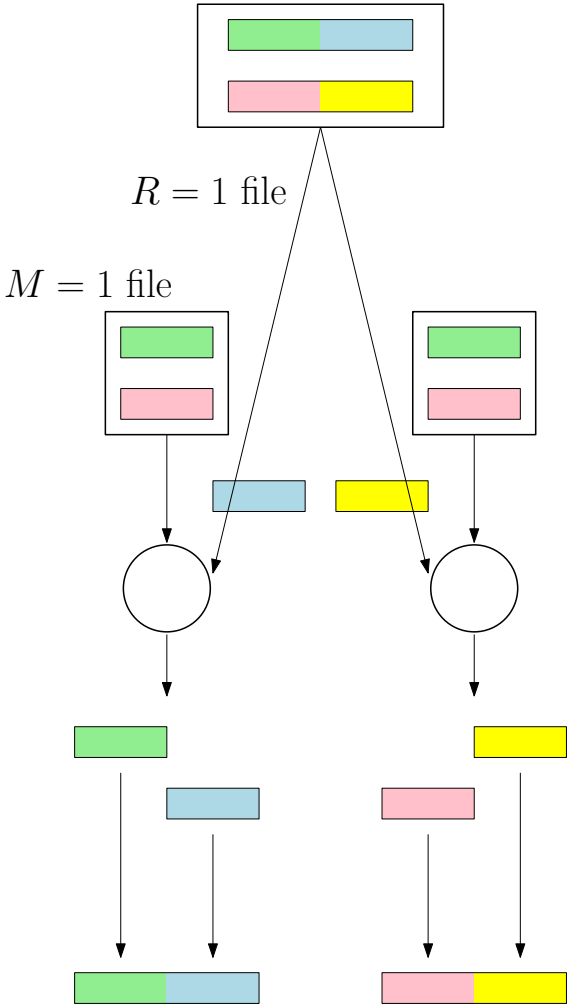
Placement phase

- Place first half of each file in each cache

Delivery phase

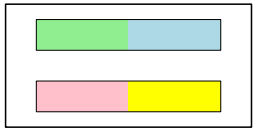
- Each user makes their request
- Server unicasts missing piece for each user
- Users recover requested files

Uncoded caching example

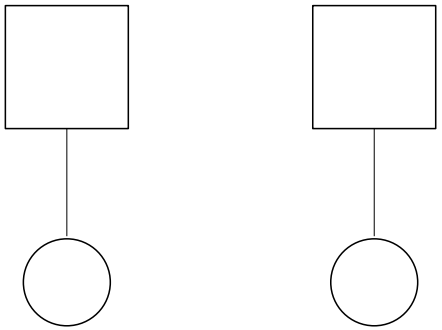


Total rate: $R = 1$ **file**

Coded caching example[2]



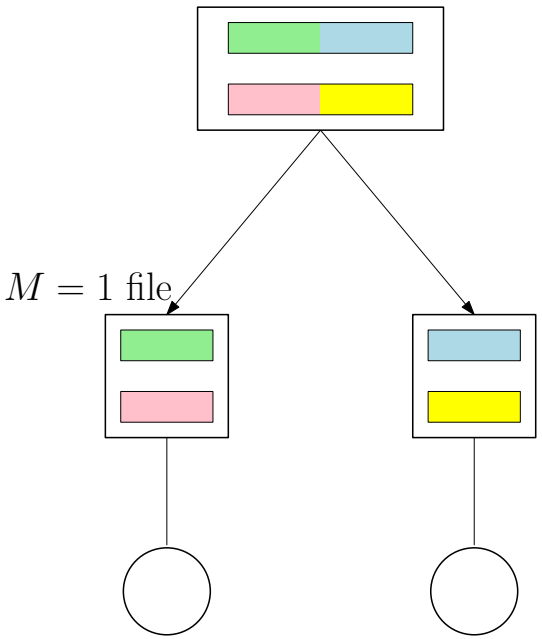
$M = 1$ file



- $N = 2$ files, $K = 2$ caches

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded caching example[2]



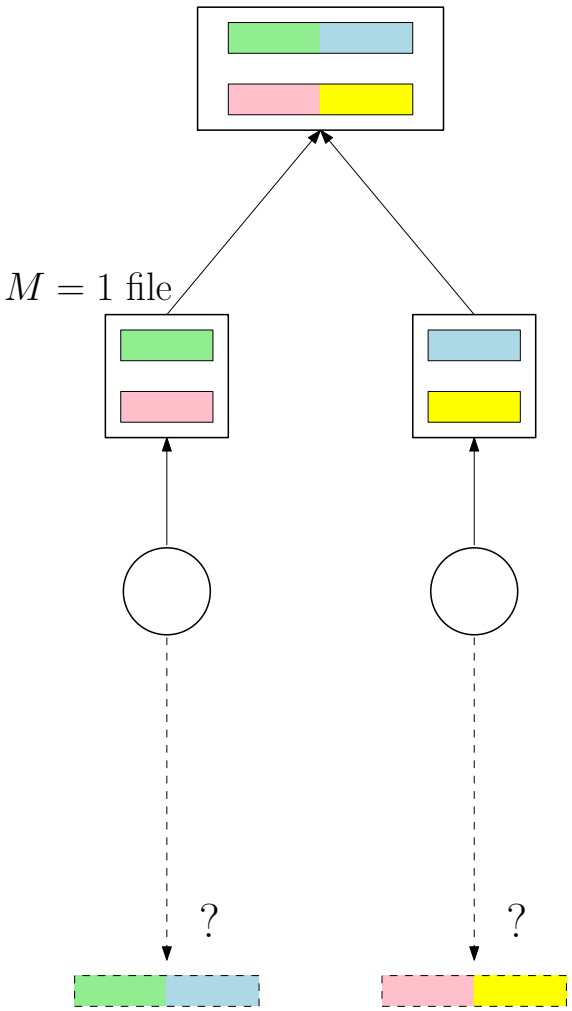
- $N = 2$ files, $K = 2$ caches

Placement phase

- Place different halves of each file in the caches

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded caching example[2]



- $N = 2$ files, $K = 2$ caches

Placement phase

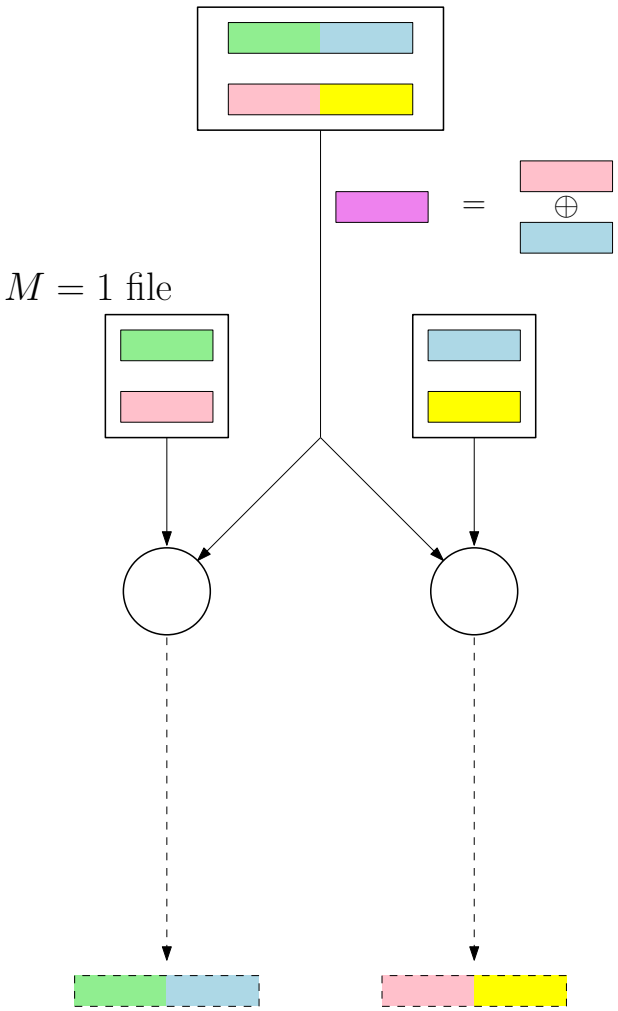
- Place different halves of each file in the caches

Delivery phase

- Each user makes their request

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded caching example[2]



- $N = 2$ files, $K = 2$ caches

Placement phase

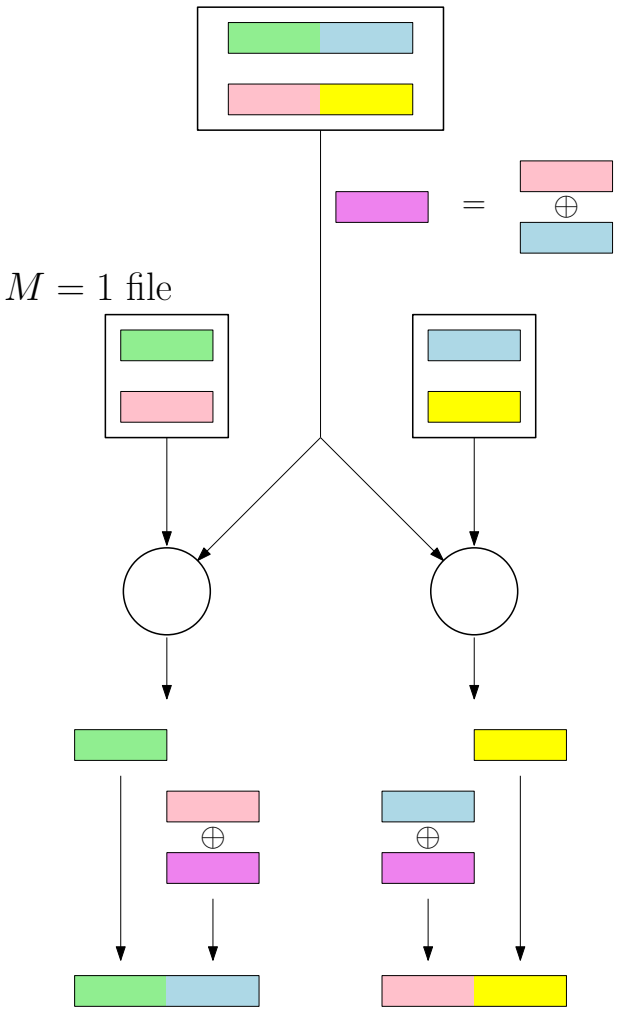
- Place different halves of each file in the caches

Delivery phase

- Each user makes their request
- Server broadcasts common coded message

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded caching example[2]



- $N = 2$ files, $K = 2$ caches

Placement phase

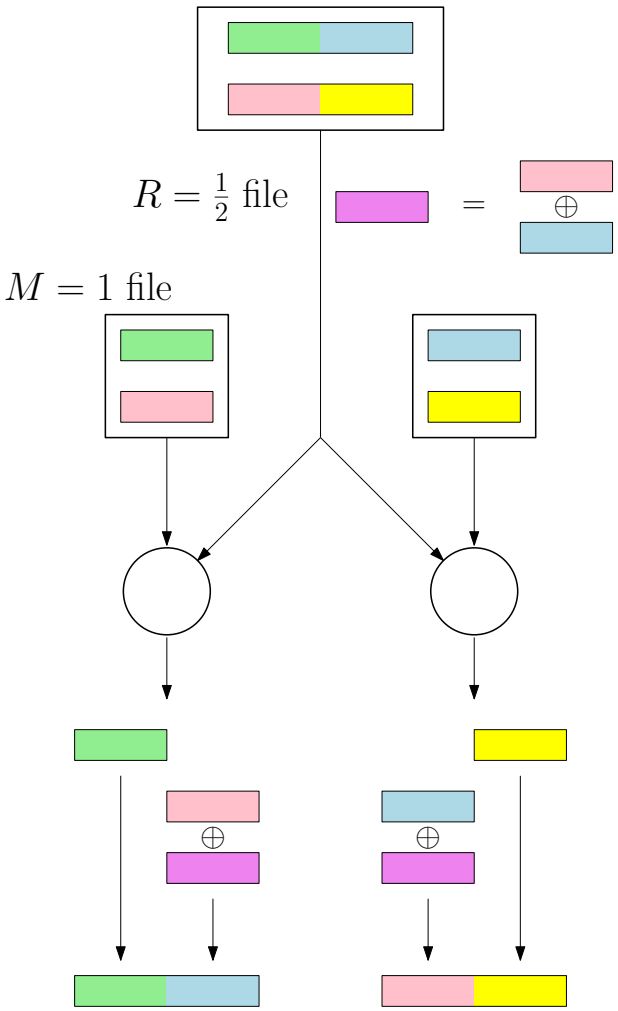
- Place different halves of each file in the caches

Delivery phase

- Each user makes their request
- Server broadcasts common coded message
- Users recover requested files

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

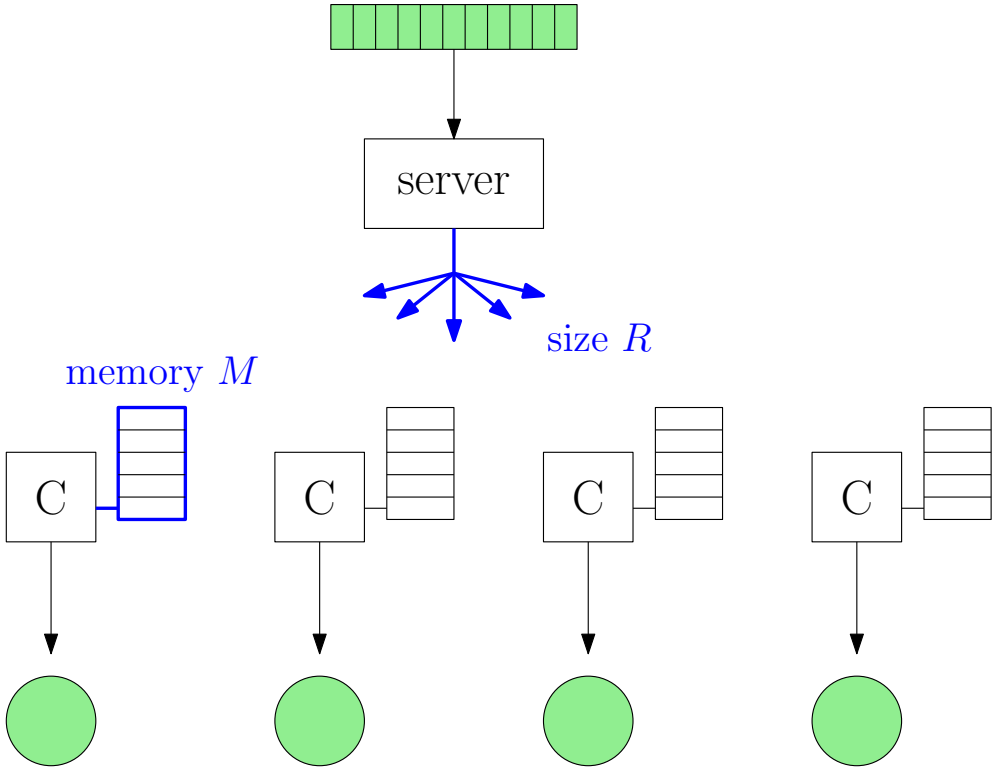
Coded caching example[2]



Total rate: $R = \frac{1}{2}$ **file**

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded Caching[3]



- N files
- K caches
- K users

[3] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded Caching[3]

- Store content to create coded multicasting opportunities

[3] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded Caching[3]

- Store content to create coded multicasting opportunities
- Use coding in the broadcast

[3] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

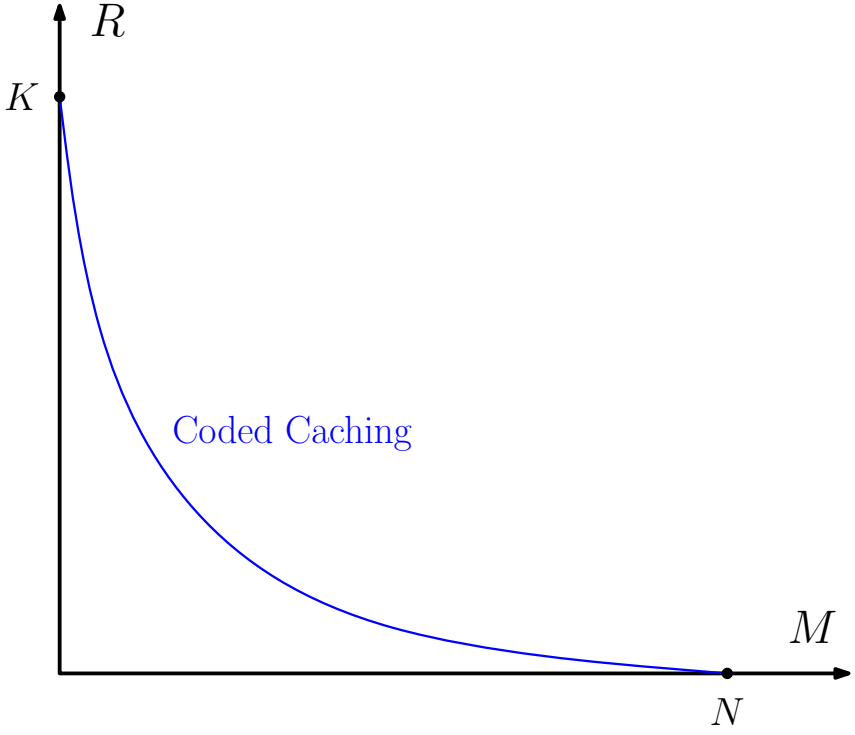
Coded Caching[3]

- Store content to create coded multicasting opportunities
- Use coding in the broadcast
- Achievable rate:

[3] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded Caching[3]

- Store content to create coded multicasting opportunities
- Use coding in the broadcast
- Achievable rate:

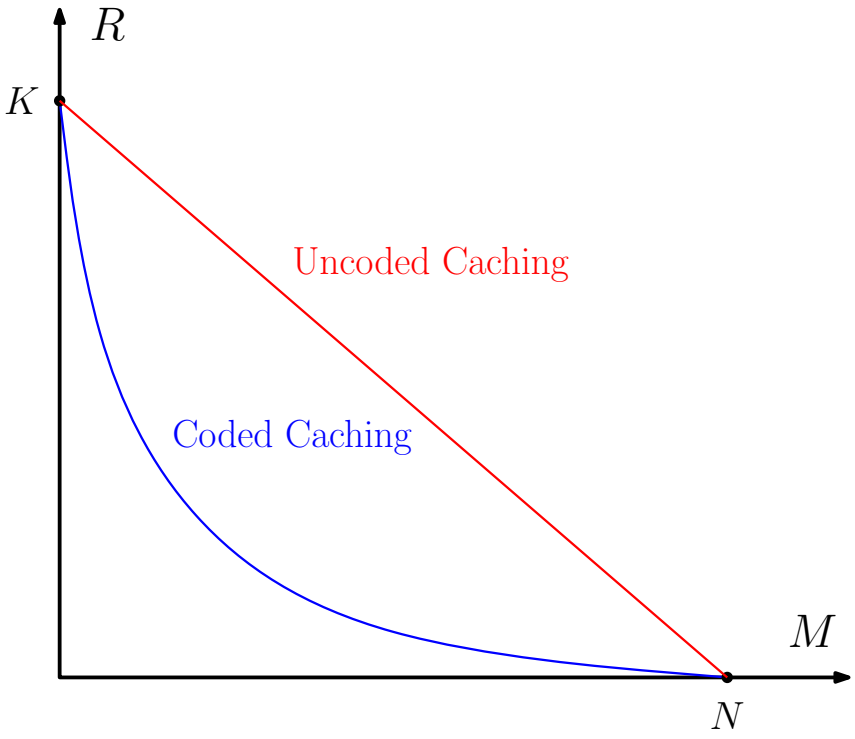


$$R \approx \min\left\{\frac{N}{M} - 1\,,\, K\right\}$$

[3] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded Caching[3]

- Store content to create coded multicasting opportunities
- Use coding in the broadcast
- Achievable rate:

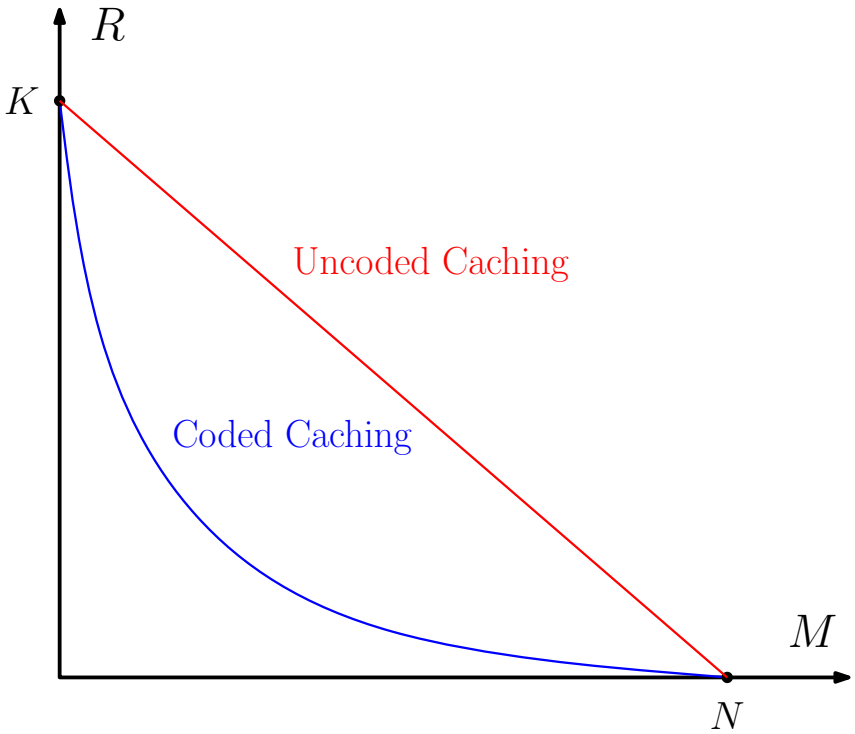


$$R \approx \min\left\{\frac{N}{M} - 1\,,\, K\right\}$$

[3] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Coded Caching[3]

- Store content to create coded multicasting opportunities
- Use coding in the broadcast
- Achievable rate:

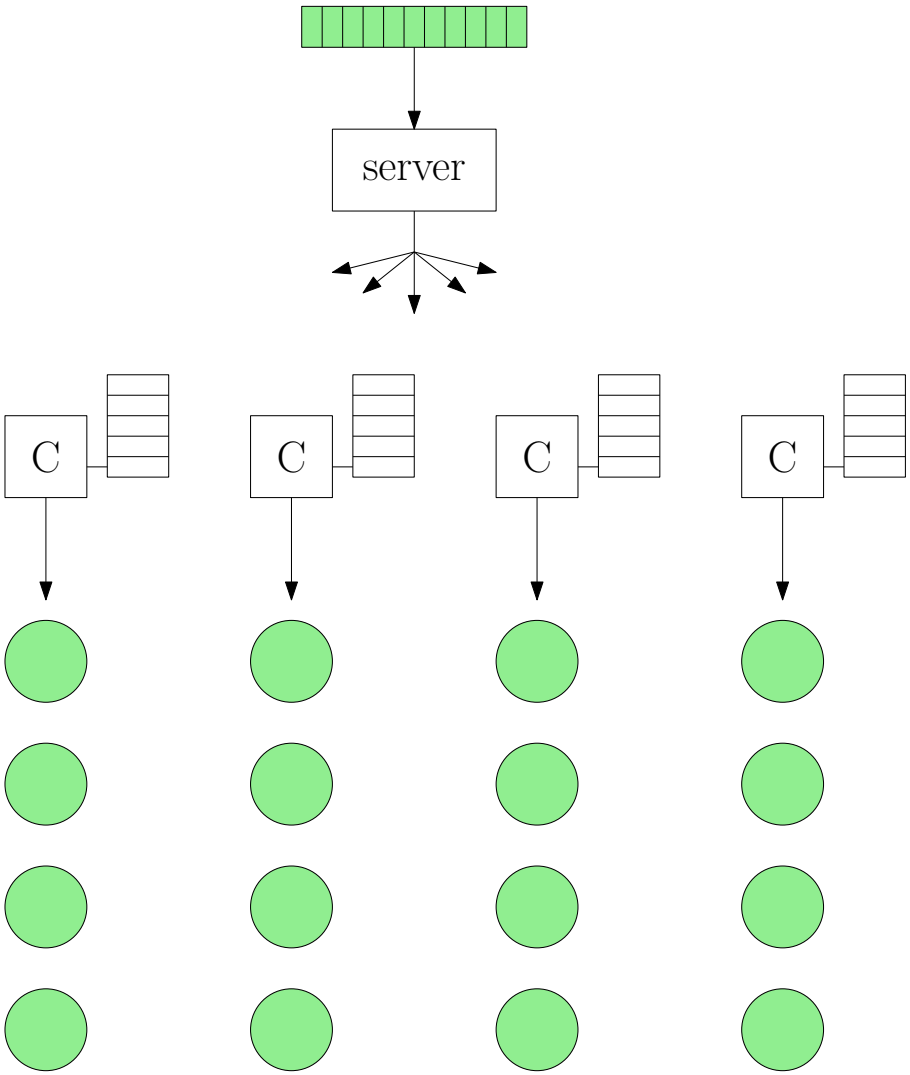


$$R \approx \min\left\{\frac{N}{M} - 1\,,\, K\right\}$$

- Scheme is order-optimal w.r.t information-theoretic bounds.

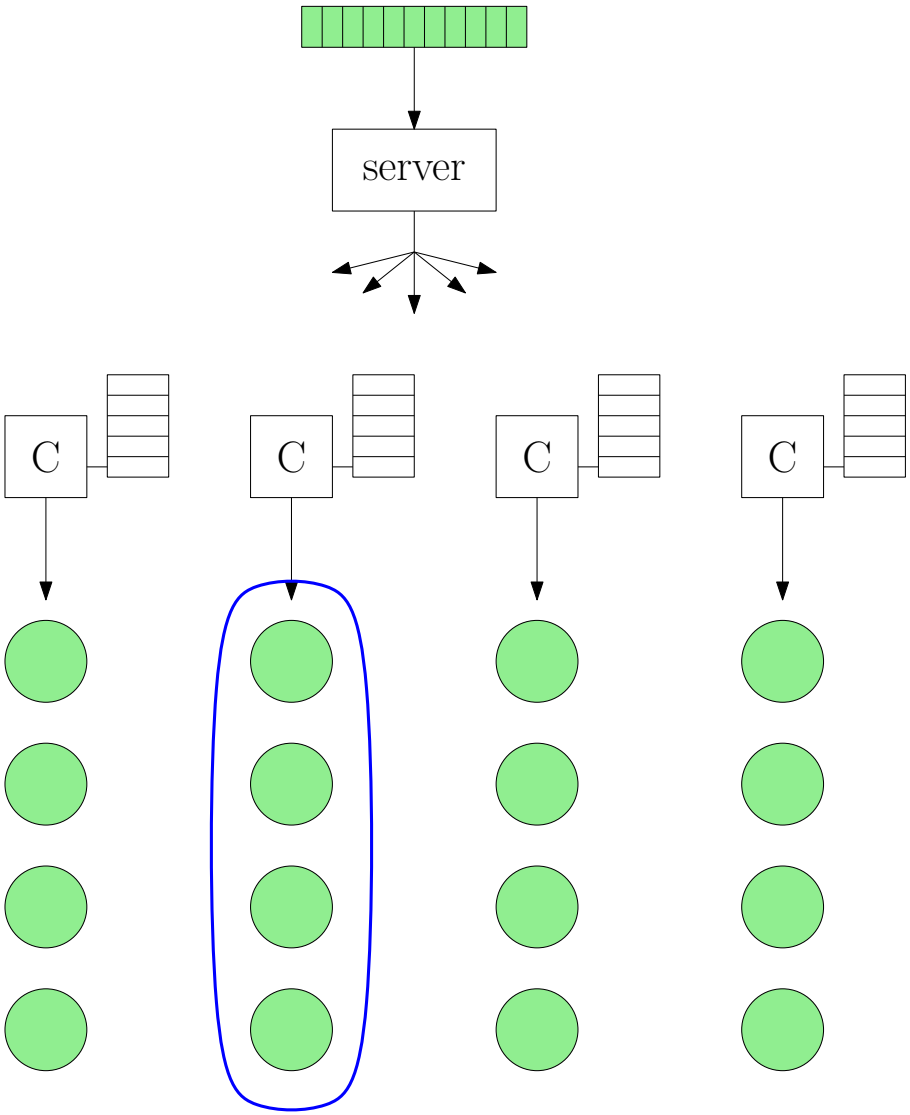
[3] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," ISIT 2013

Effect of number of users in multi-level coded caching



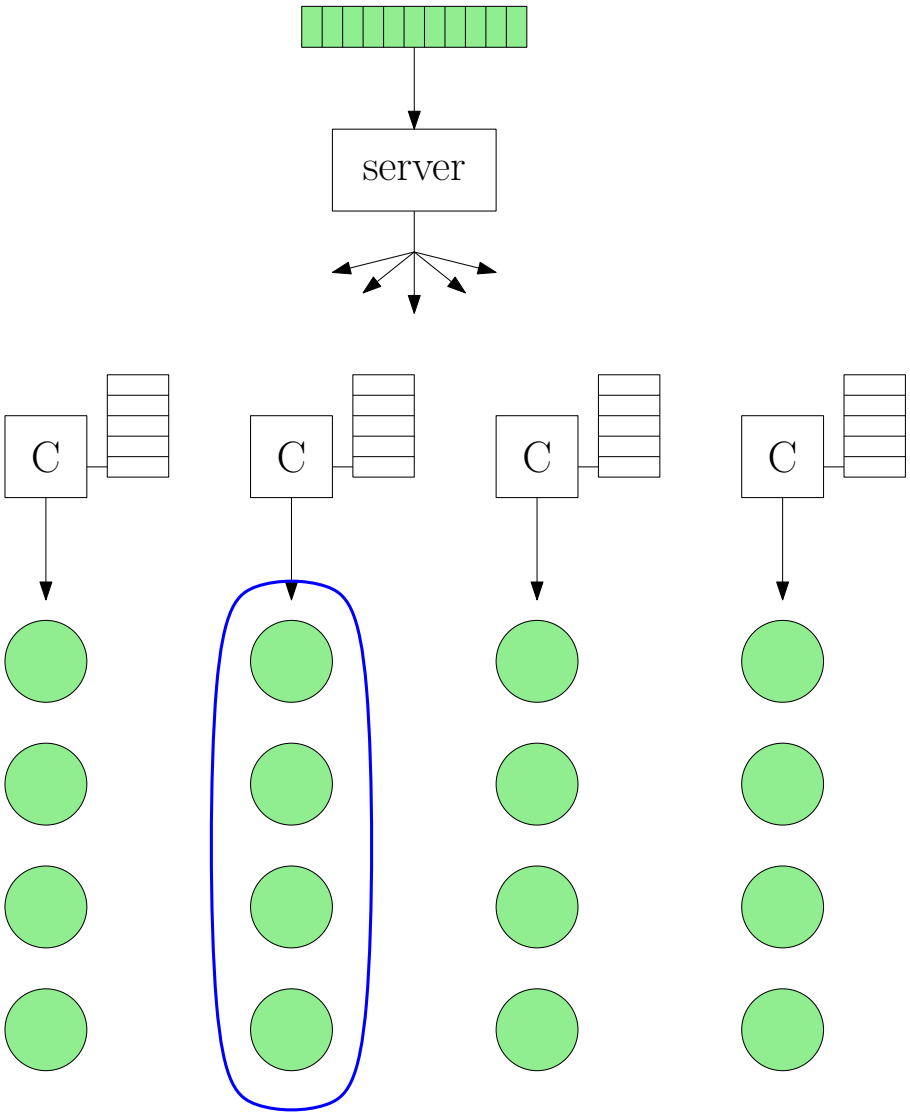
- Simple extension: multiple users per cache [H., K., D., 2014]

Effect of number of users in multi-level coded caching



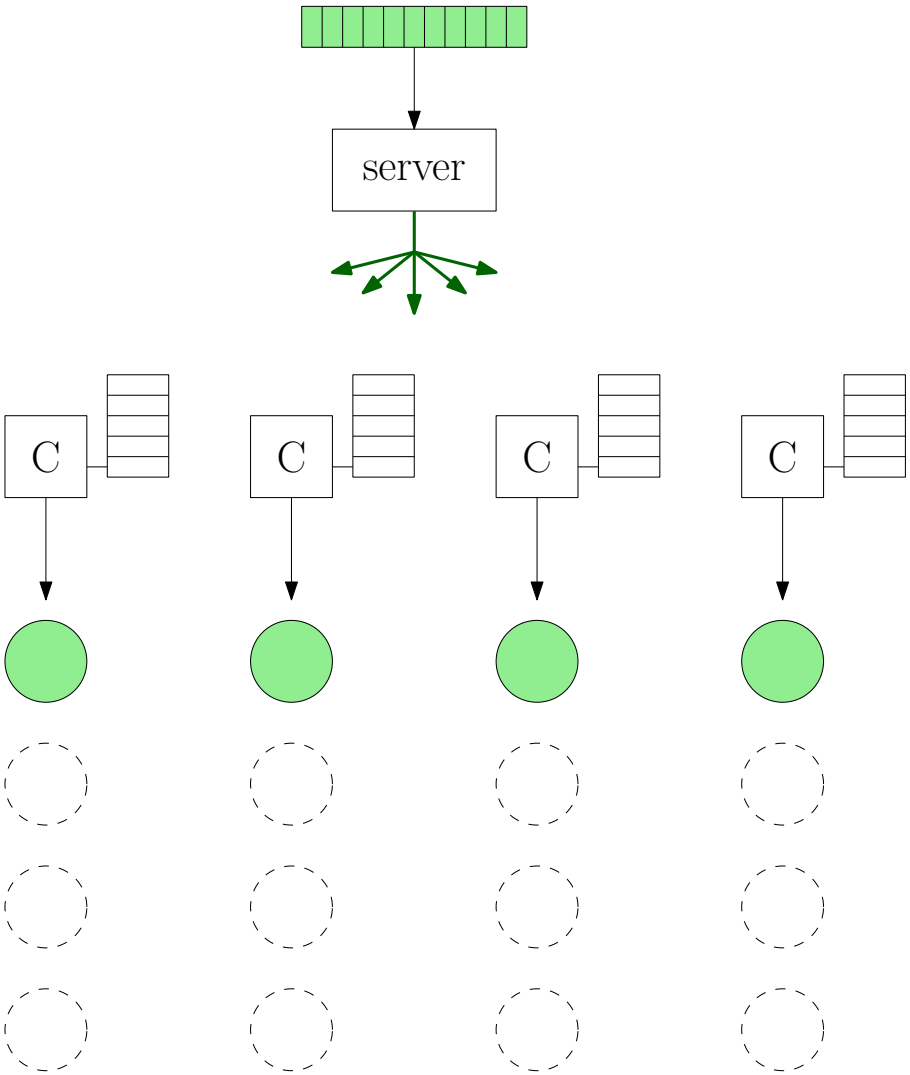
- Simple extension: multiple users per cache [H., K., D., 2014]
- Users with identical side information

Effect of number of users in multi-level coded caching



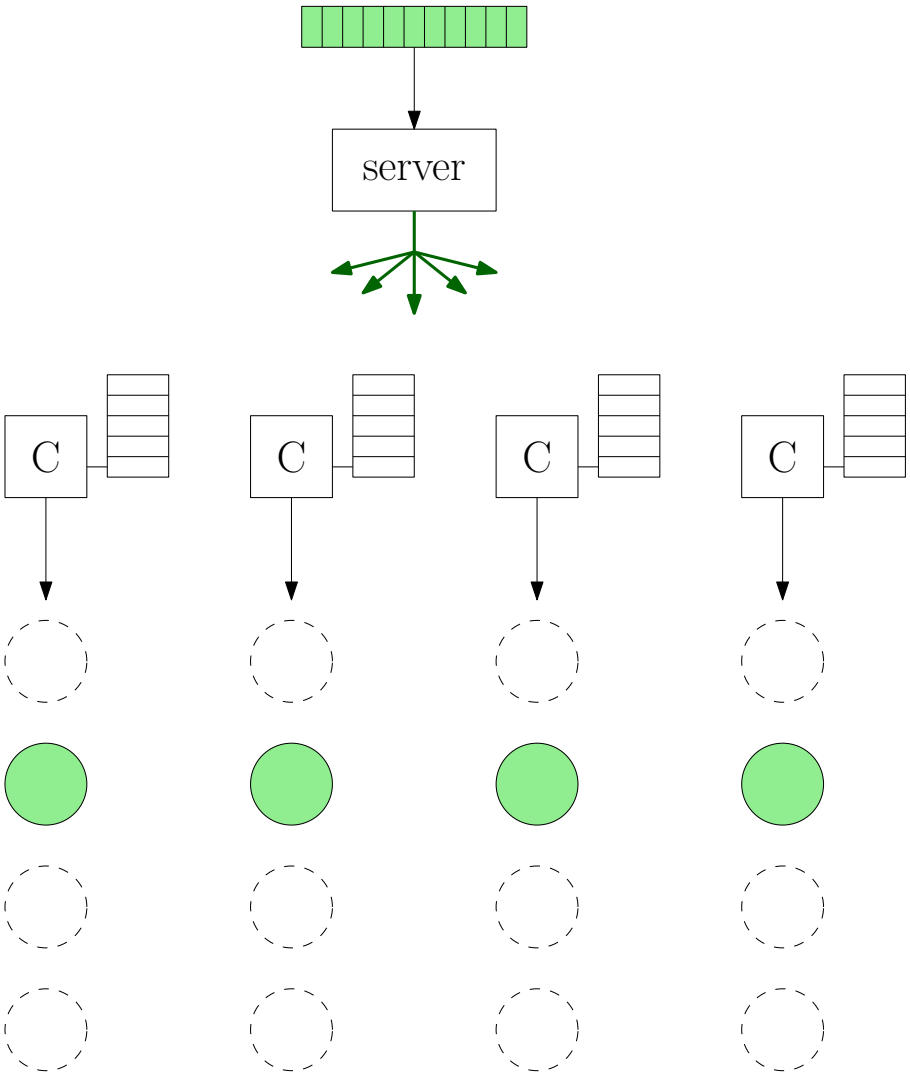
- Simple extension: multiple users per cache [H., K., D., 2014]
- Users with identical side information
- No coding opportunities possible

Effect of number of users in multi-level coded caching



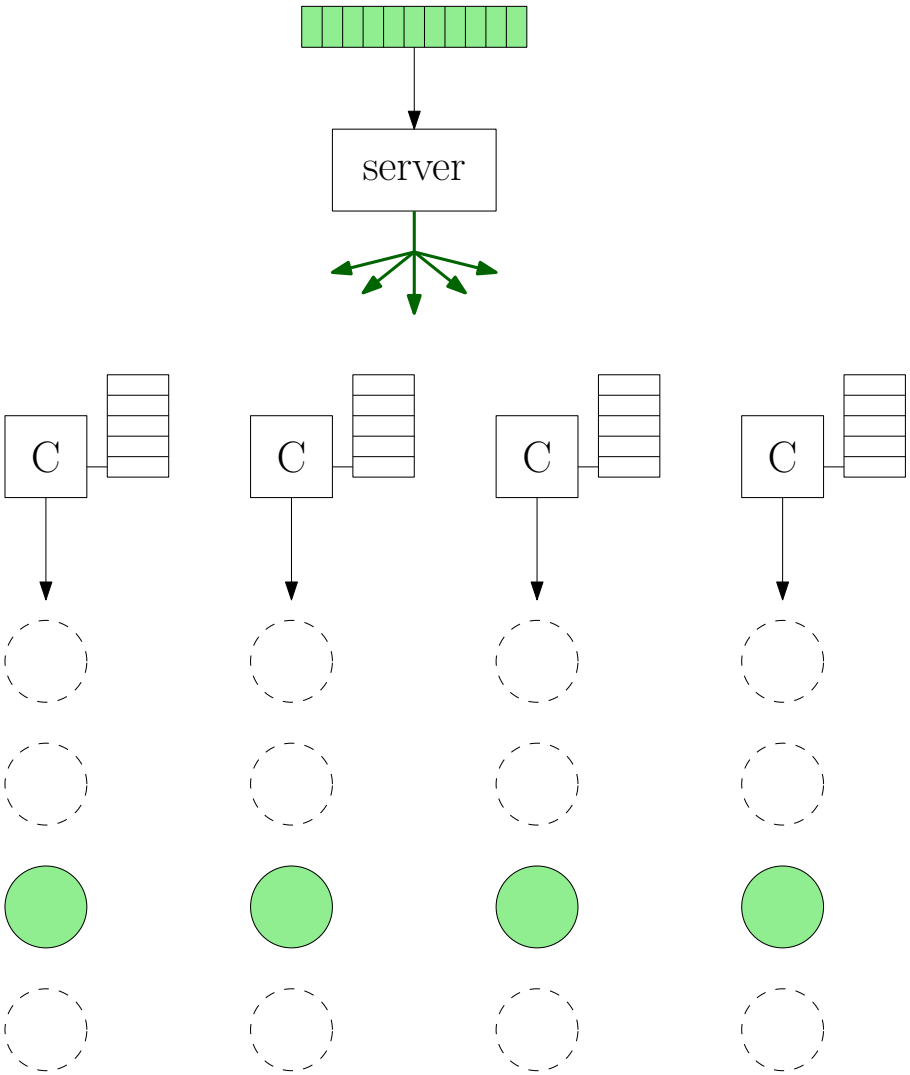
- Simple extension: multiple users per cache [H., K., D., 2014]
- Users with identical side information
- No coding opportunities possible
- Treat each row separately

Effect of number of users in multi-level coded caching



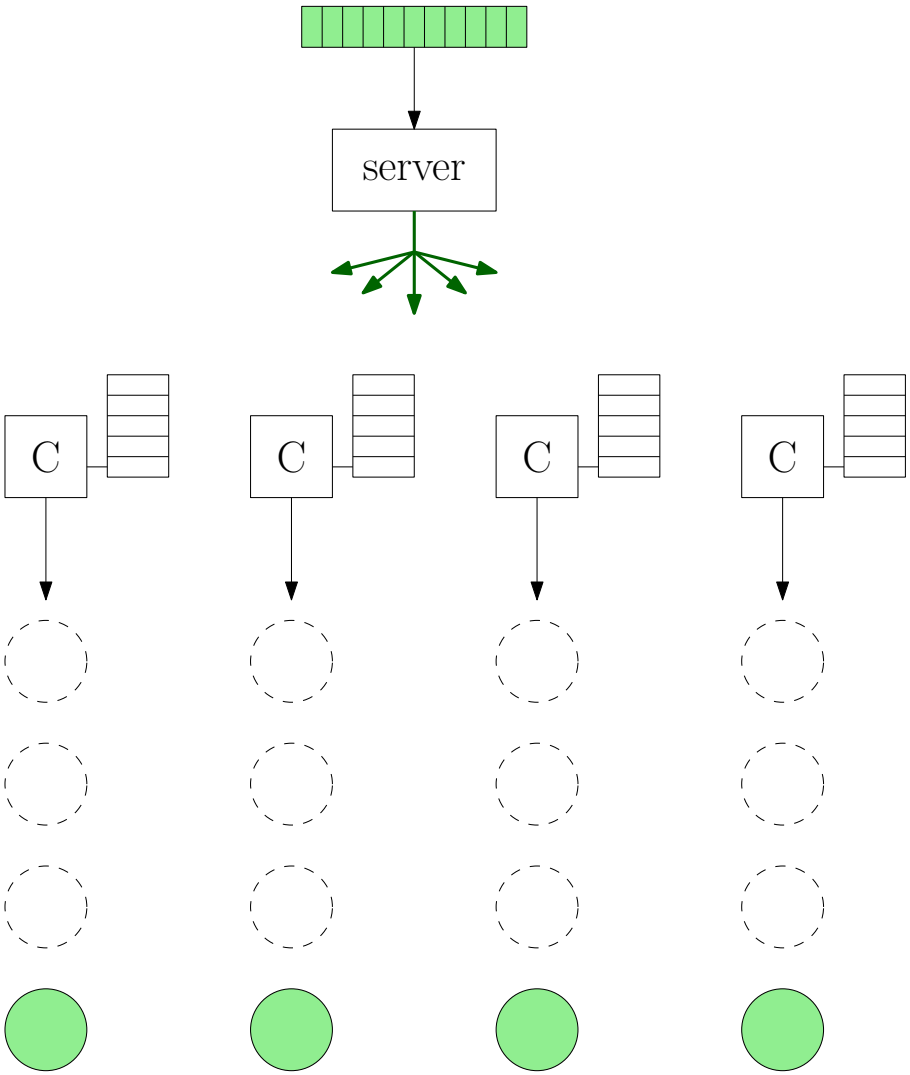
- Simple extension: multiple users per cache [H., K., D., 2014]
- Users with identical side information
- No coding opportunities possible
- Treat each row separately

Effect of number of users in multi-level coded caching



- Simple extension: multiple users per cache [H., K., D., 2014]
- Users with identical side information
- No coding opportunities possible
- Treat each row separately

Effect of number of users in multi-level coded caching



- Simple extension: multiple users per cache [H., K., D., 2014]
- Users with identical side information
- No coding opportunities possible
- Treat each row separately

Effect of number of users in multi-level coded caching

File popularity:
Likelihood of being requested by a user

Effect of number of users in multi-level coded caching

File popularity:
Likelihood of being requested by a user

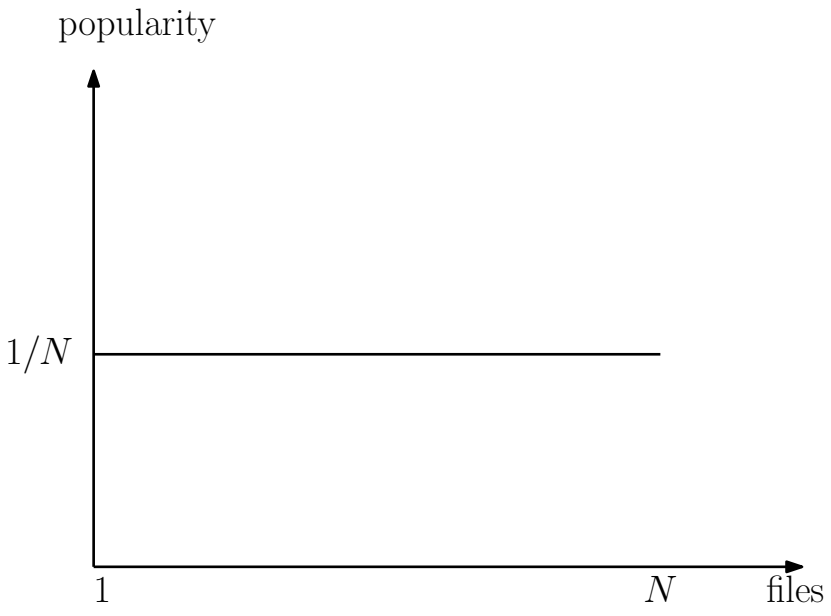
Different models in the literature:

Effect of number of users in multi-level coded caching

File popularity:
Likelihood of being requested by a user

Different models in the literature:

- Uniform



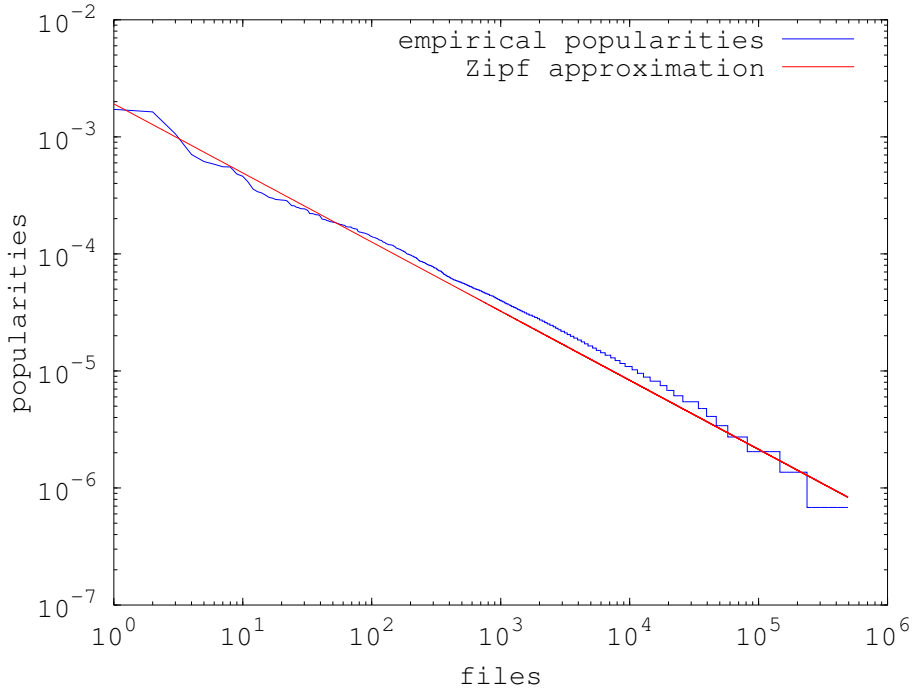
Effect of number of users in multi-level coded caching

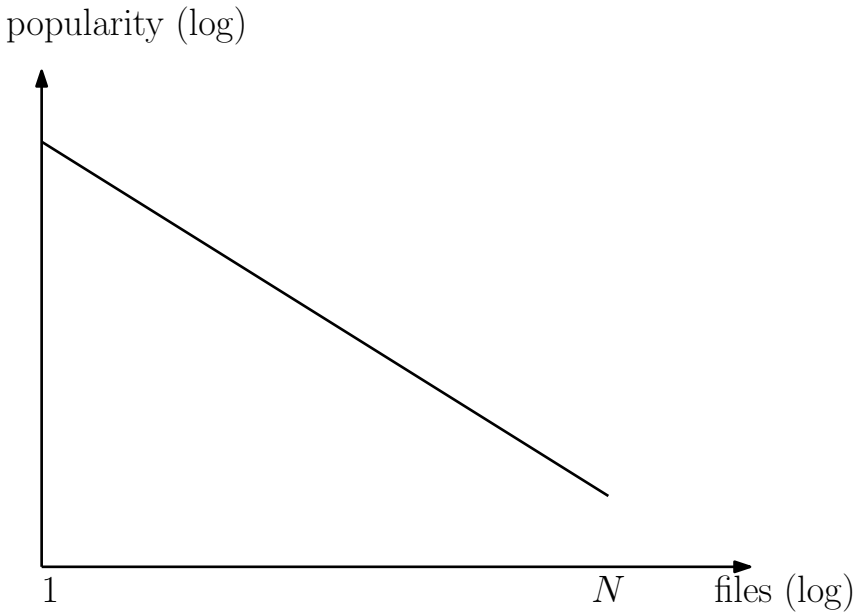
File popularity:
Likelihood of being requested by a user

Different models in the literature:

- Uniform
- Zipf (power law)

Example: YouTube





Effect of number of users in multi-level coded caching

File popularity:
Likelihood of being requested by a user

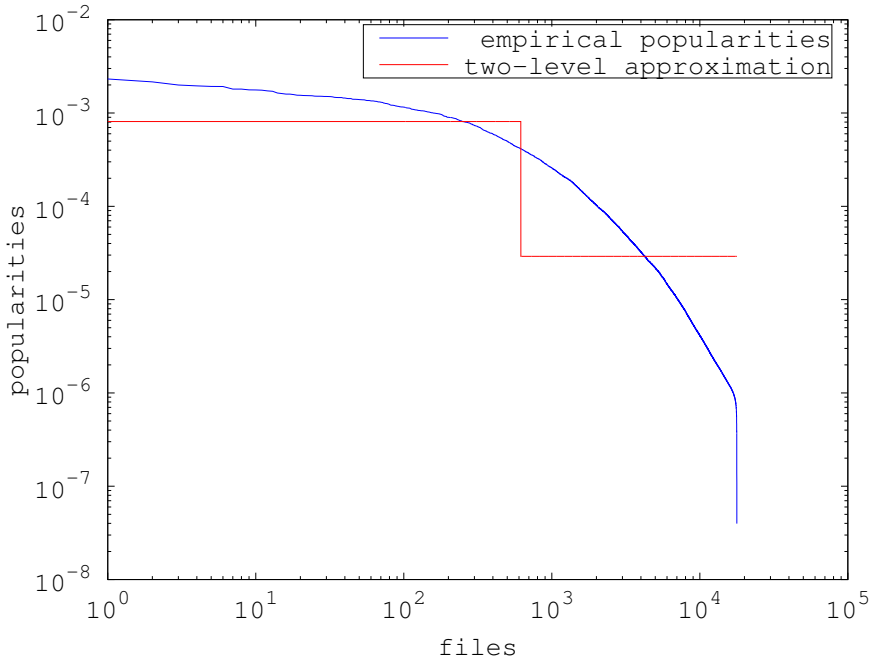
Different models in the literature:

- Uniform
- Zipf (power law)
- Arbitrary

Effect of number of users in multi-level coded caching

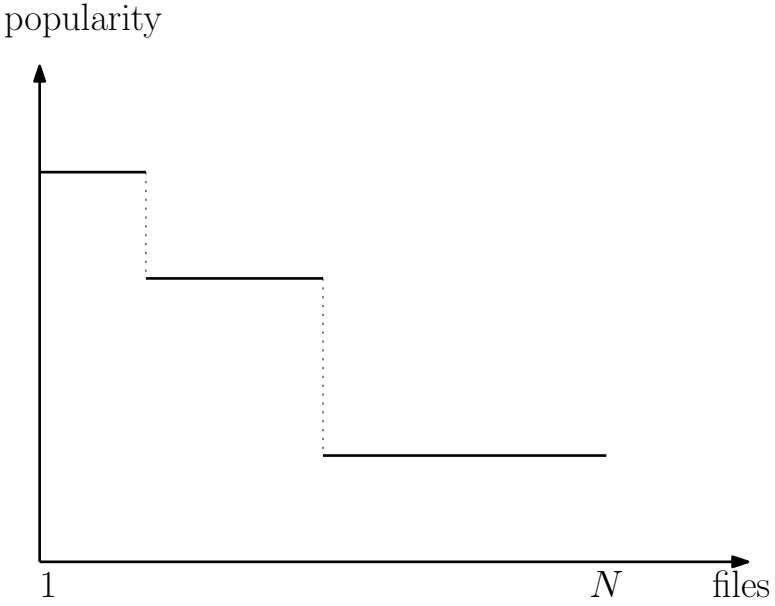
File popularity:
Likelihood of being requested by a user

Example: Netflix



Different models in the literature:

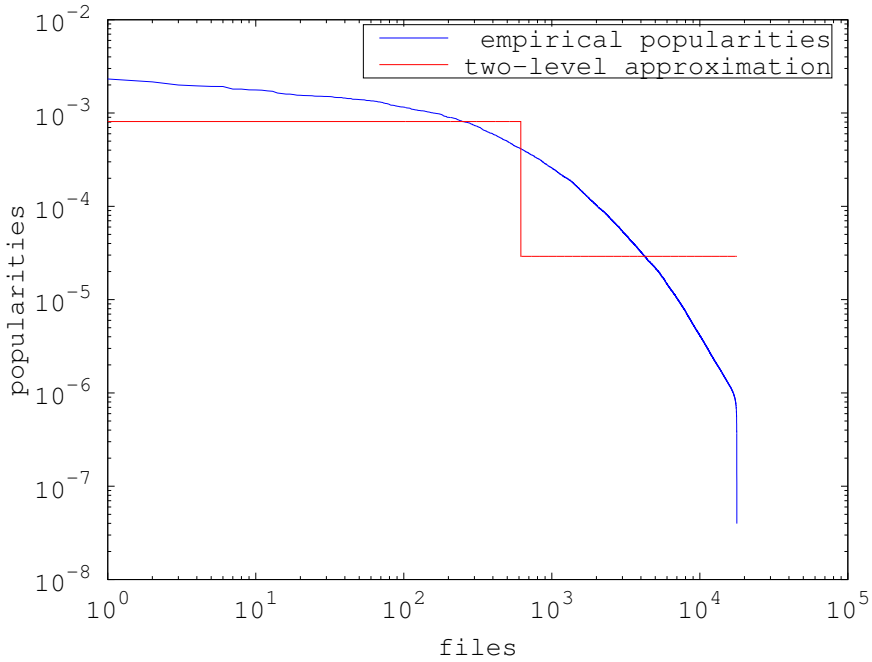
- Uniform
- Zipf (power law)
- Arbitrary
- Multi-level [H., K., D., 2014]



Effect of number of users in multi-level coded caching

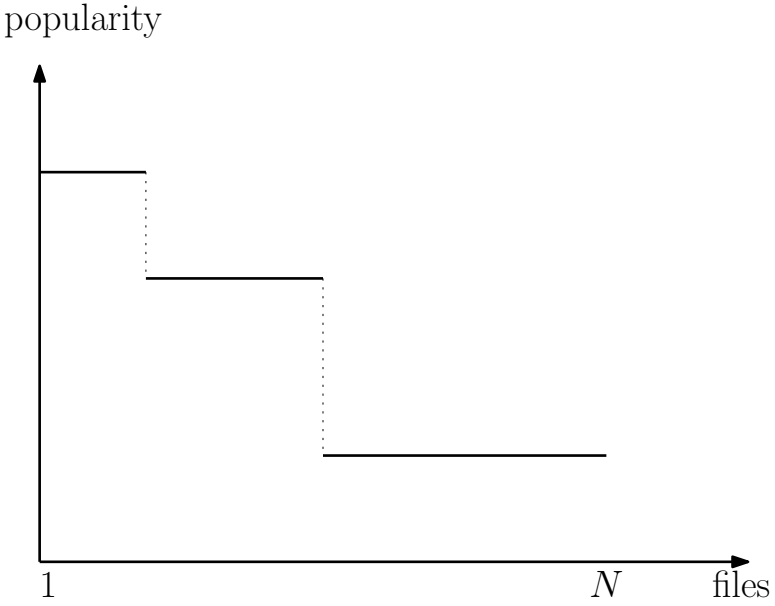
File popularity:
Likelihood of being requested by a user

Example: Netflix



Different models in the literature:

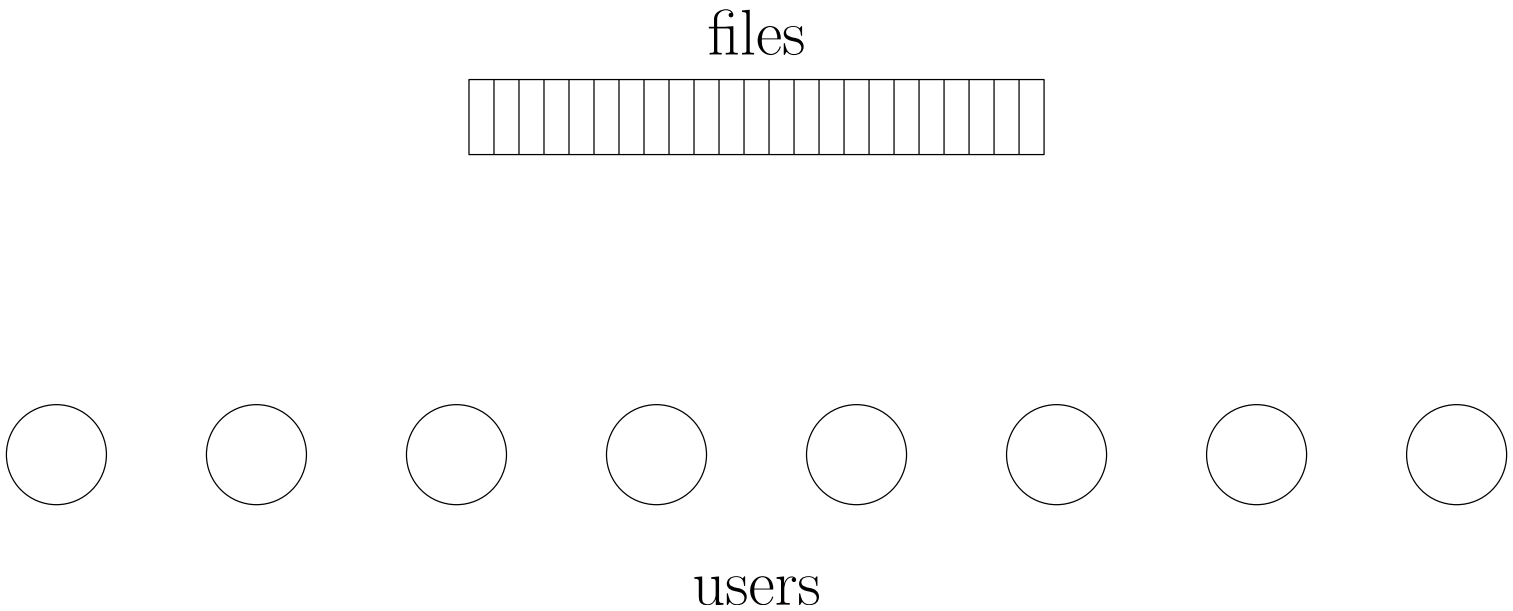
- Uniform
- Zipf (power law)
- Arbitrary
- Multi-level [H., K., D., 2014]



Effect of number of users in multi-level coded caching

General

Example



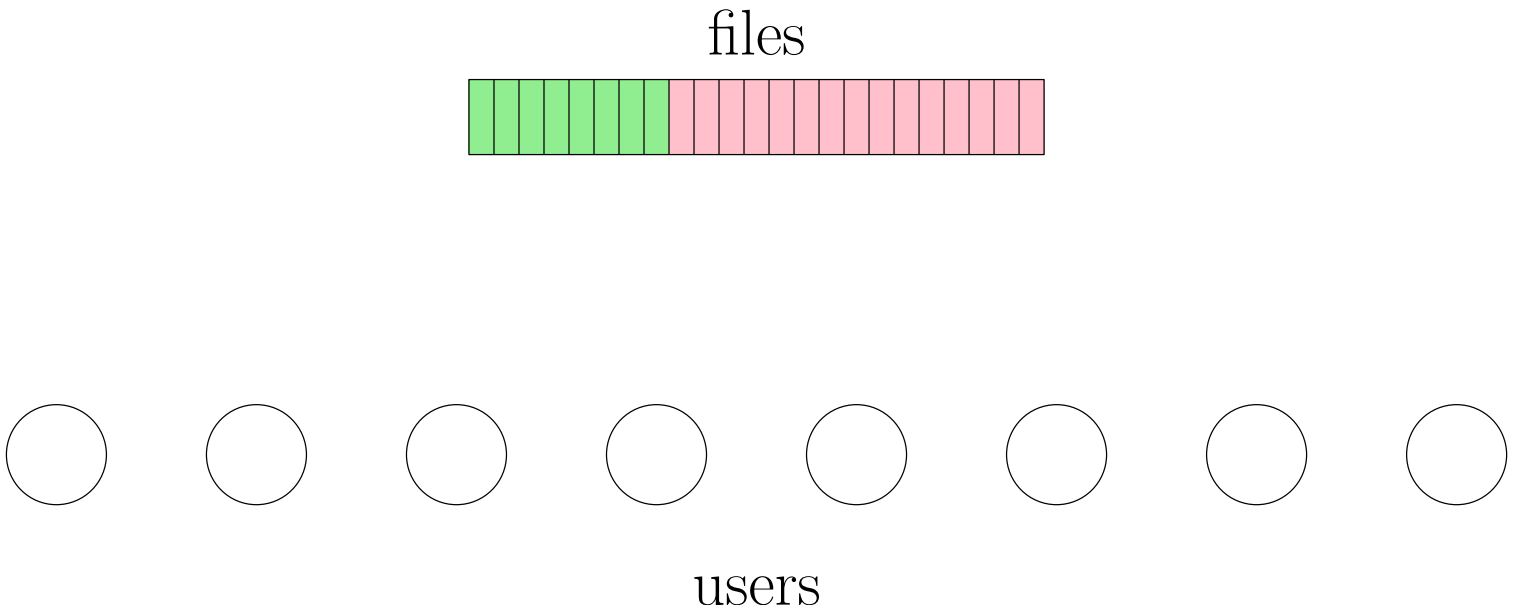
Effect of number of users in multi-level coded caching

General

- Files divided into popularity levels (classes)

Example

- 2 levels



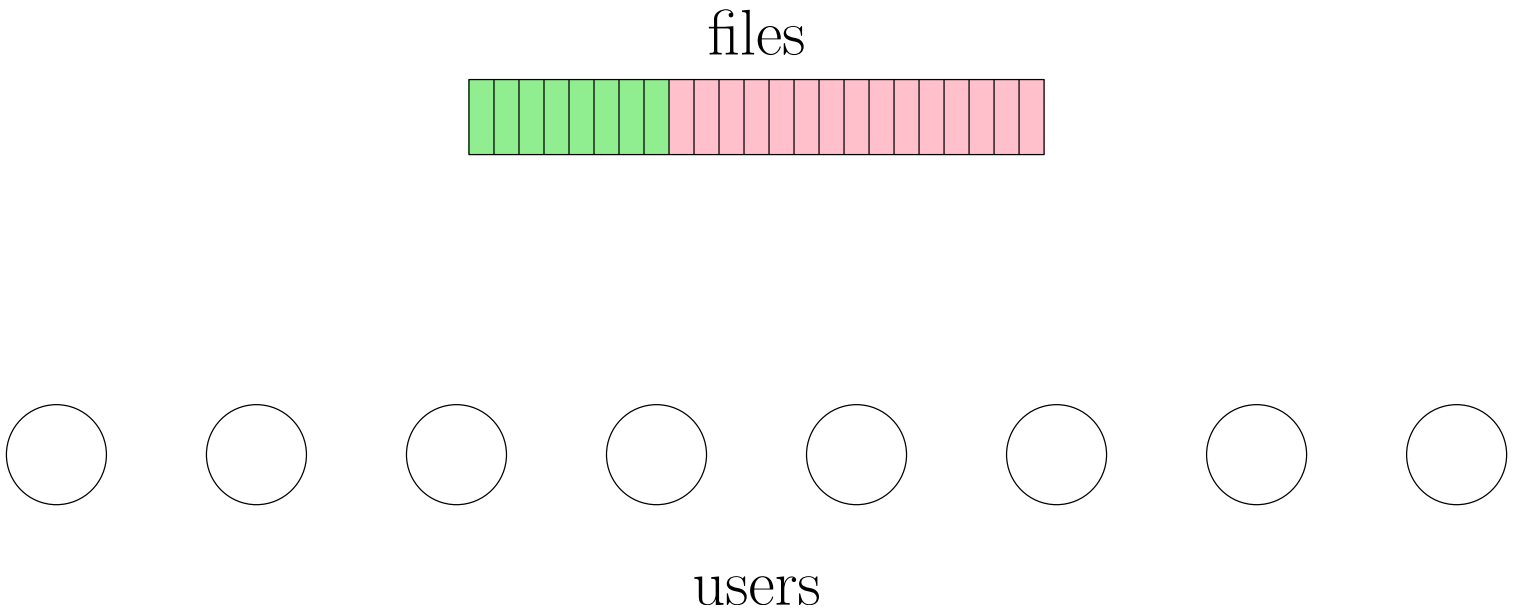
Effect of number of users in multi-level coded caching

General

- Files divided into popularity levels (classes)
  - Uniform popularity within each level

Example

- 2 levels



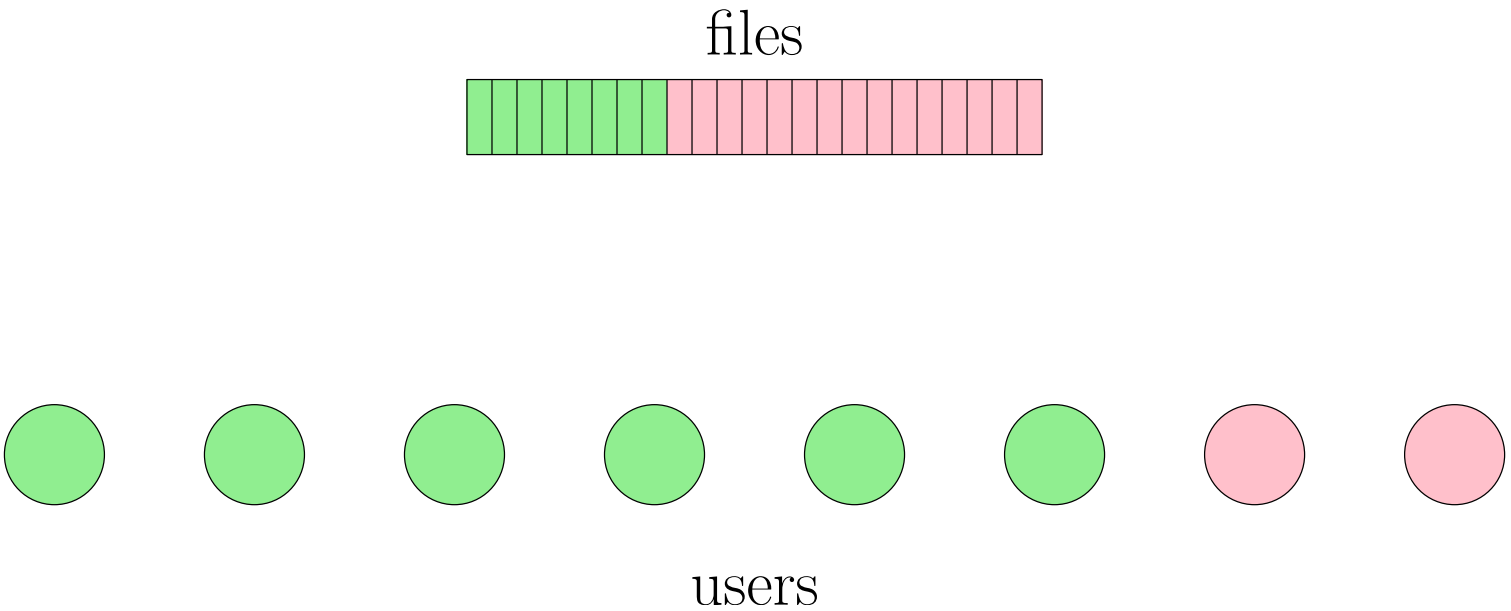
Effect of number of users in multi-level coded caching

General

- Files divided into popularity levels (classes)
  - Uniform popularity within each level
- Fixed fraction of users per level

Example

- 2 levels
- 75%–25%

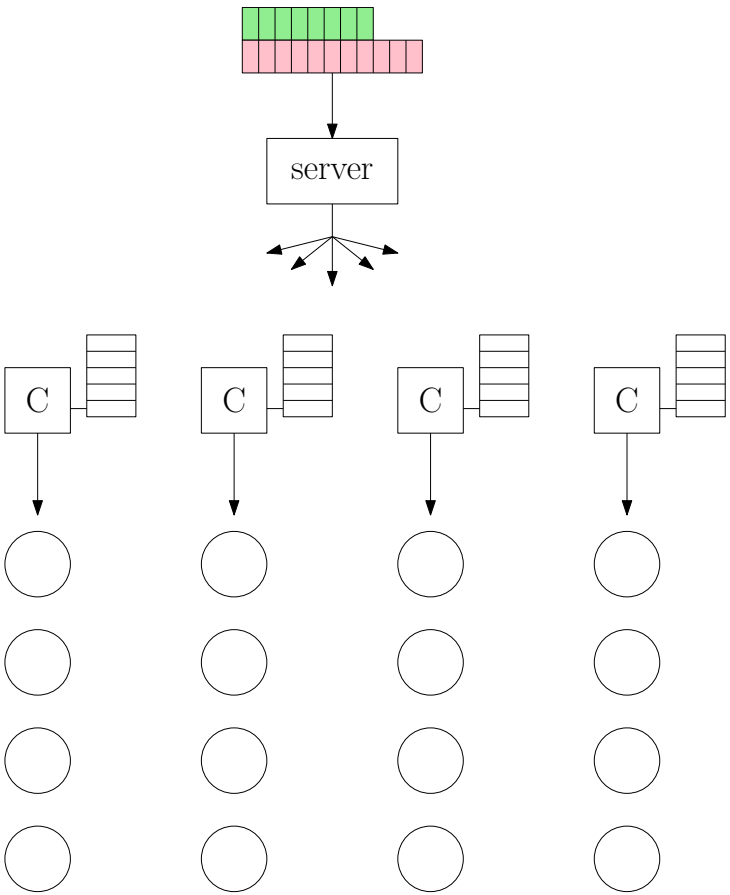


Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%

Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%



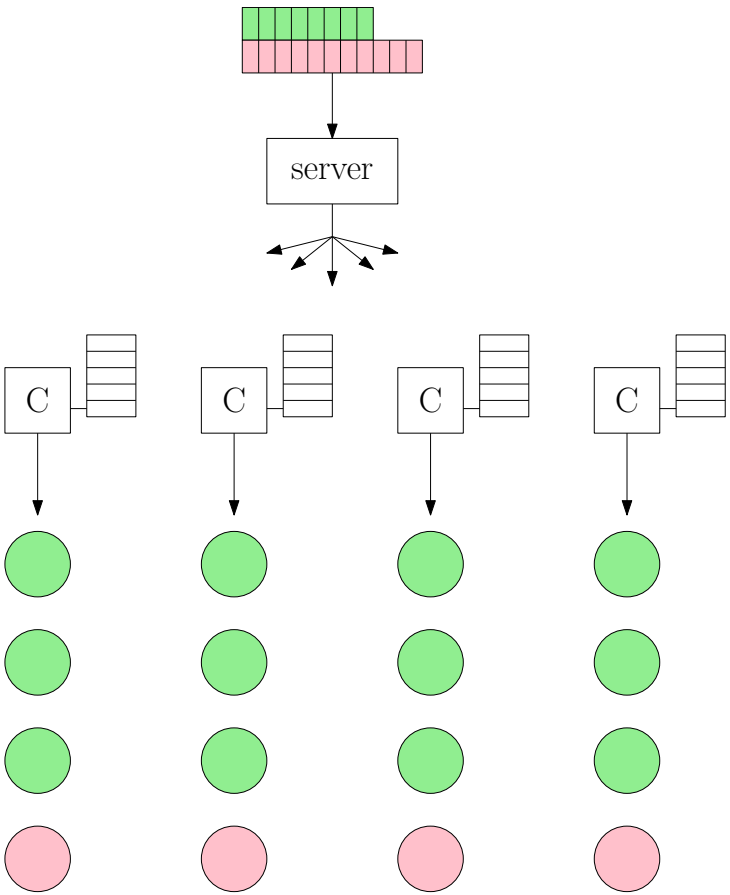
"Multi-user setup"

[H., K., D., 2014]

- 4 users per cache

Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%



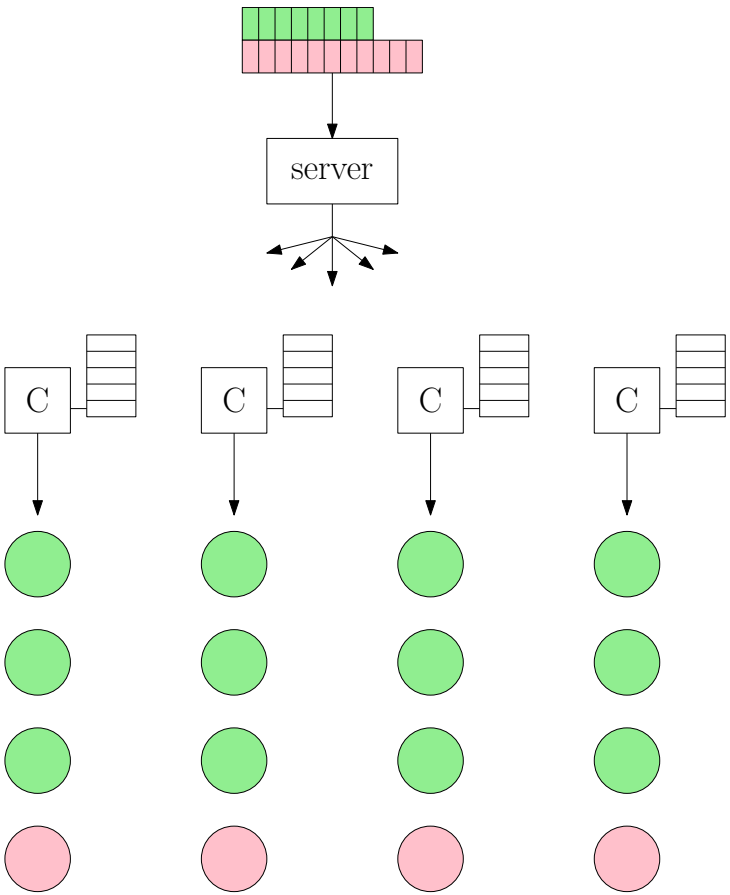
"Multi-user setup"

[H., K., D., 2014]

- 4 users per cache
- Level 1: 3 users per cache;
  Level 2: 1 user per cache

Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%



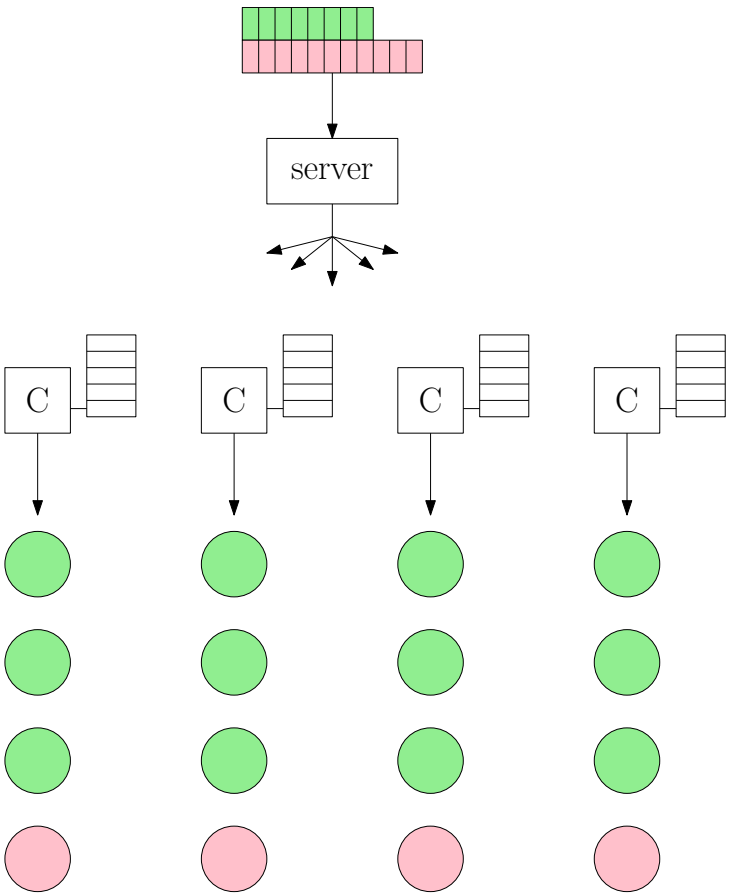
"Multi-user setup"

[H., K., D., 2014]

- 4 users per cache
- Level 1: 3 users per cache;
  Level 2: 1 user per cache
- Proportion maintained at each cache

Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%



"Multi-user setup"

[H., K., D., 2014]

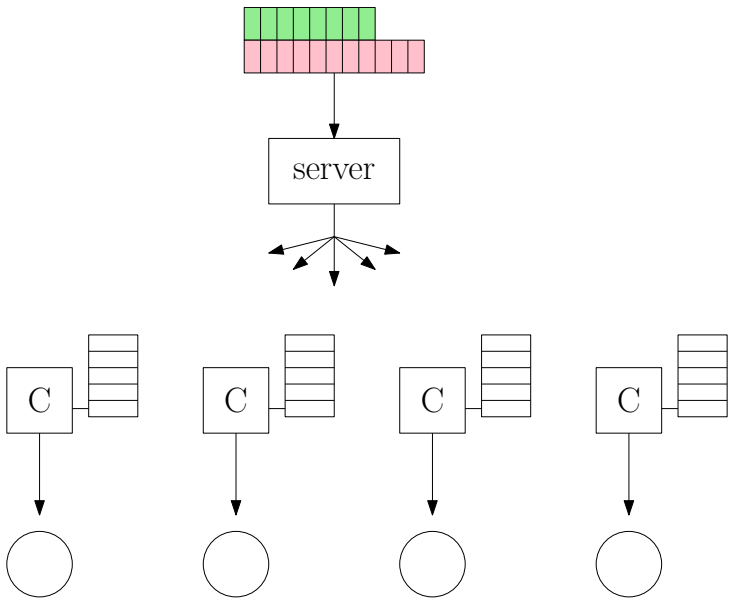
- 4 users per cache
- Level 1: 3 users per cache; Level 2: 1 user per cache
- Proportion maintained at each cache
- Expected when number of users per cache is large

Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%

"Single-user setup"

- One user per cache

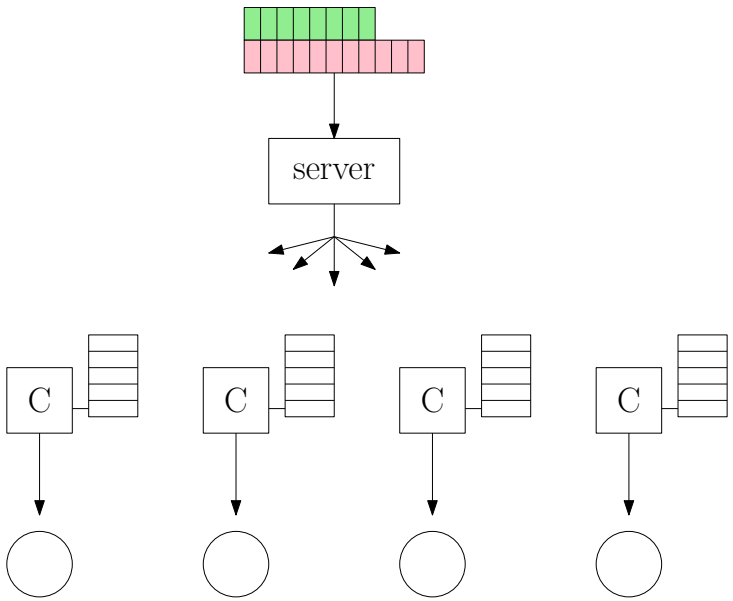


Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%

"Single-user setup"

- One user per cache
- 4 users total

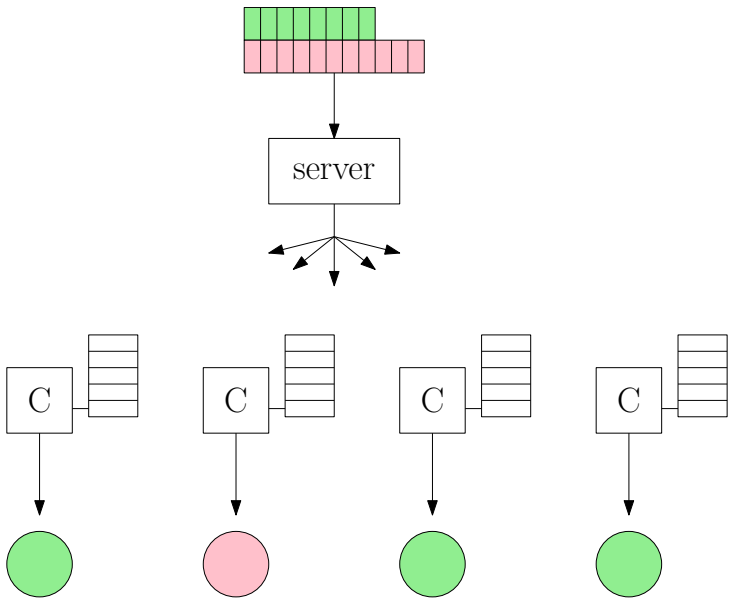


Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%

"Single-user setup"

- One user per cache
- 4 users total
- Level 1: 3 users;
  Level 2: 1 user

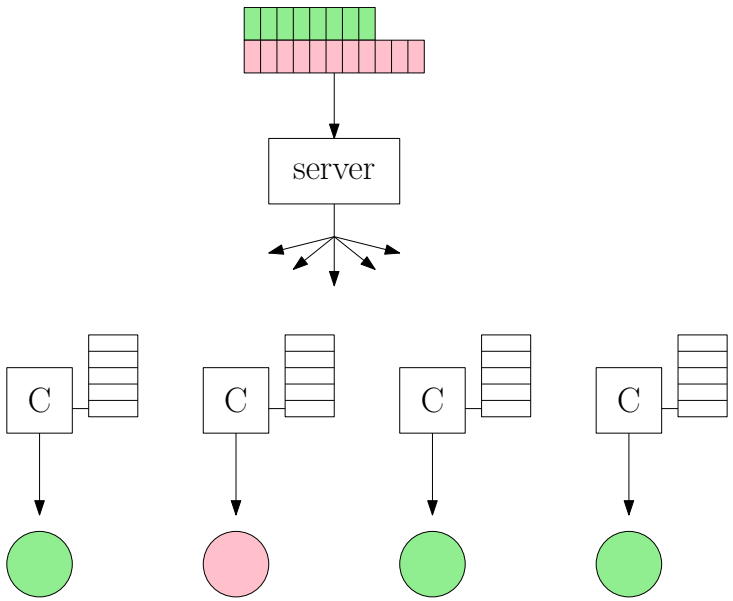


Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%

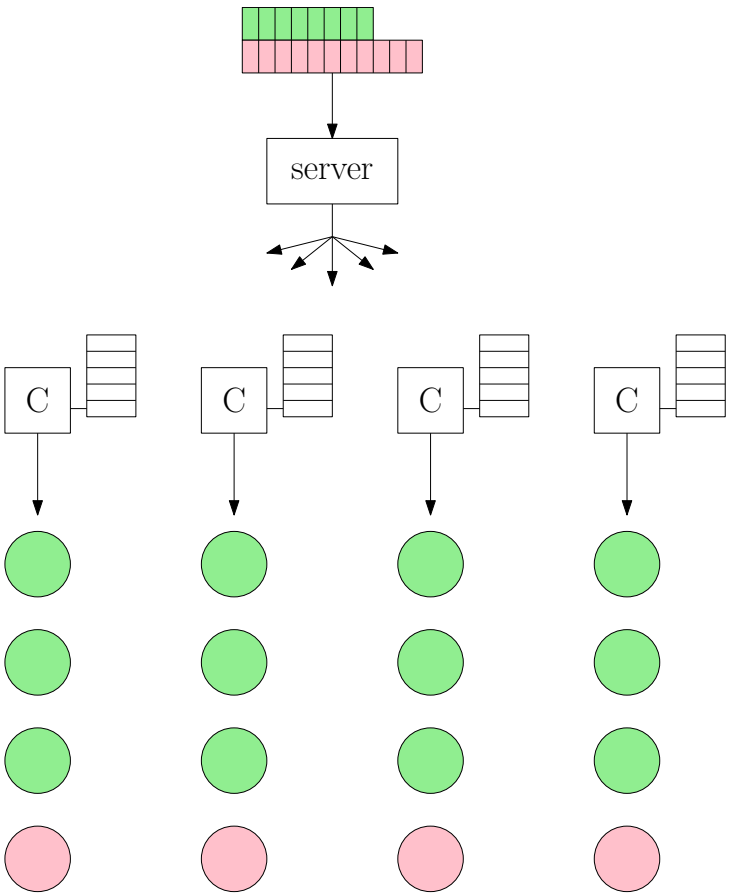
"Single-user setup"

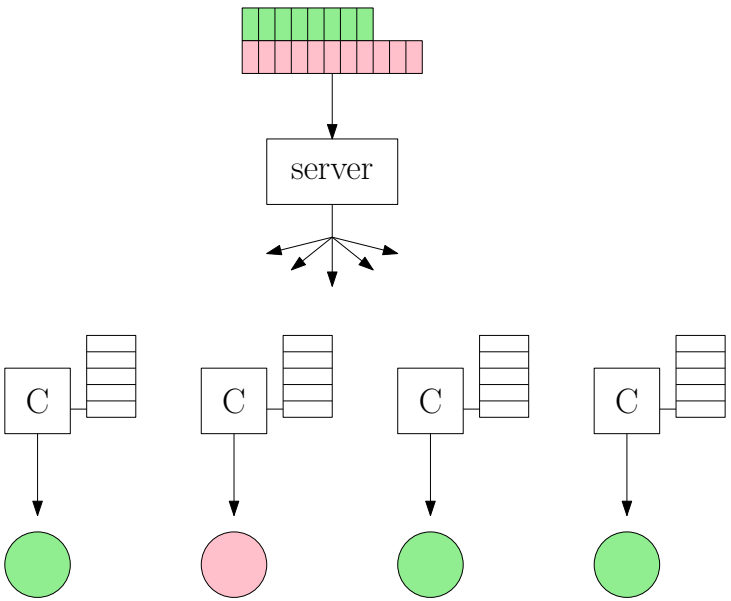
- One user per cache
- 4 users total
- Level 1: 3 users;
  Level 2: 1 user
- Proportion maintained across all caches



Effect of number of users in multi-level coded caching

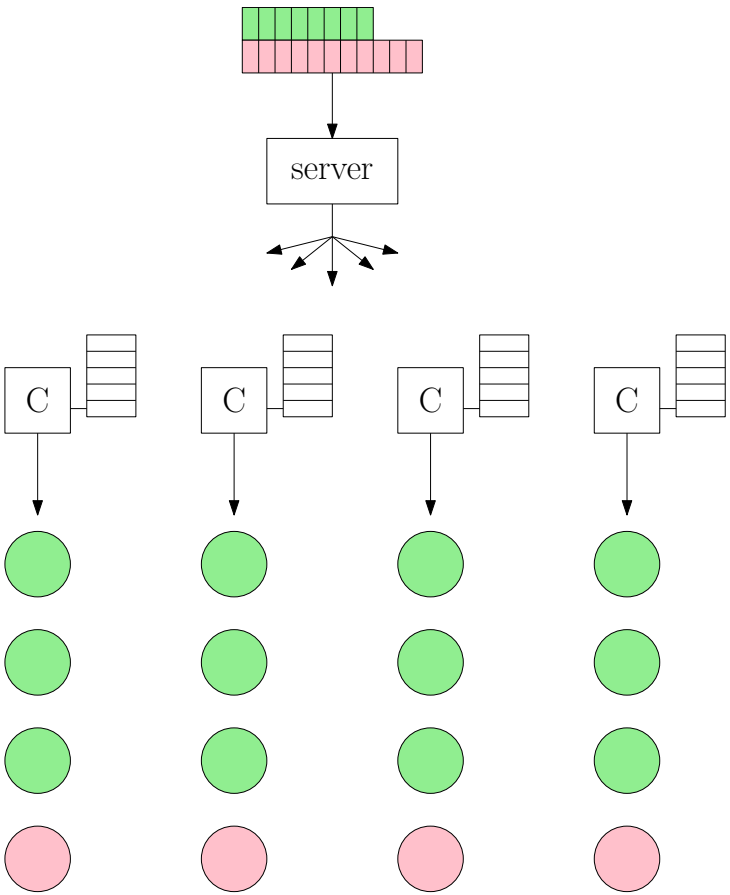
Example: 2 levels of files, dividing users into 75%–25%



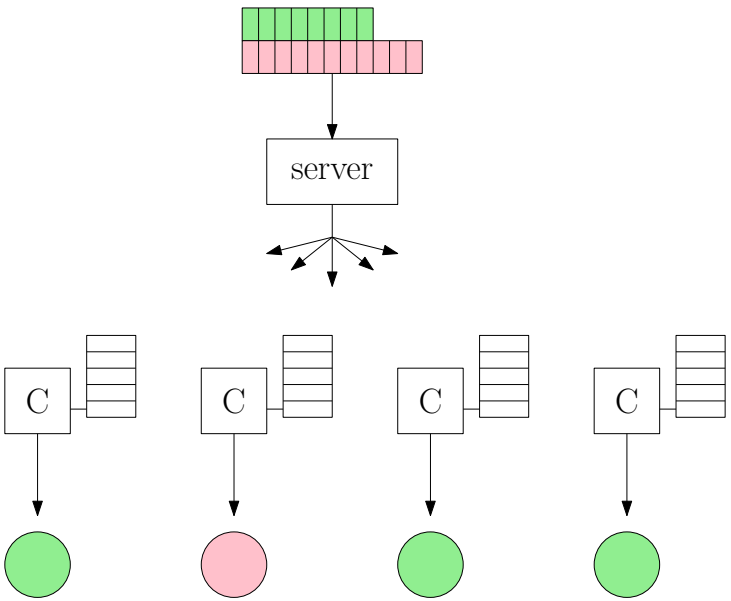


Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%



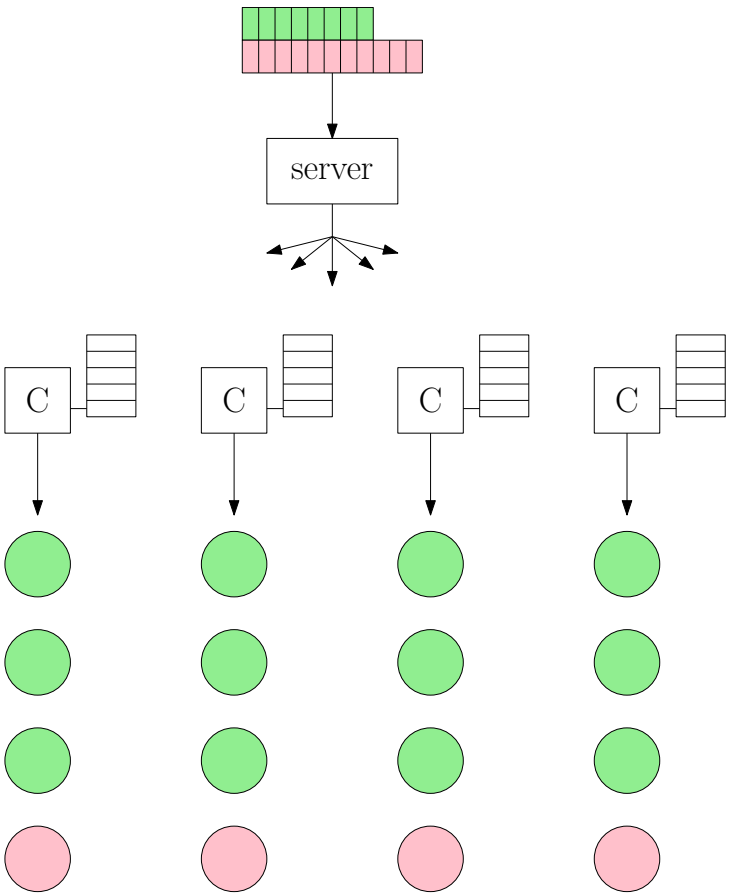
Same user profile at each cache



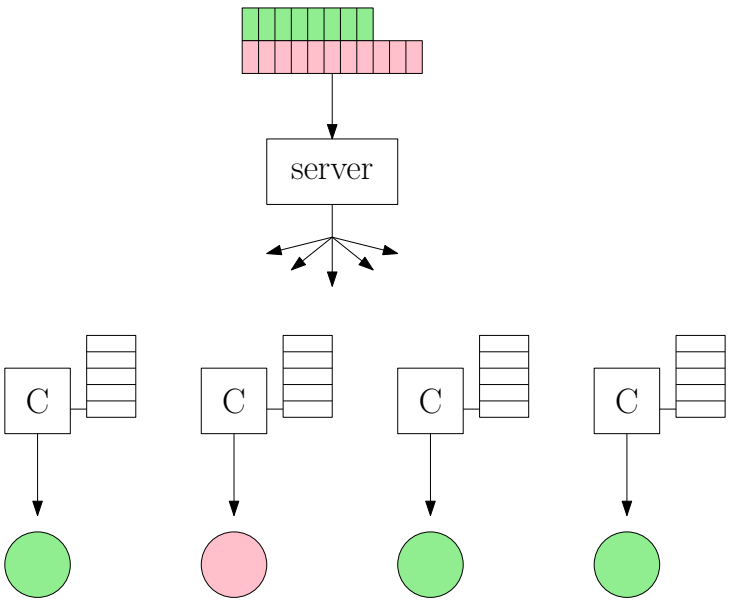
Different user profile at each cache

Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%



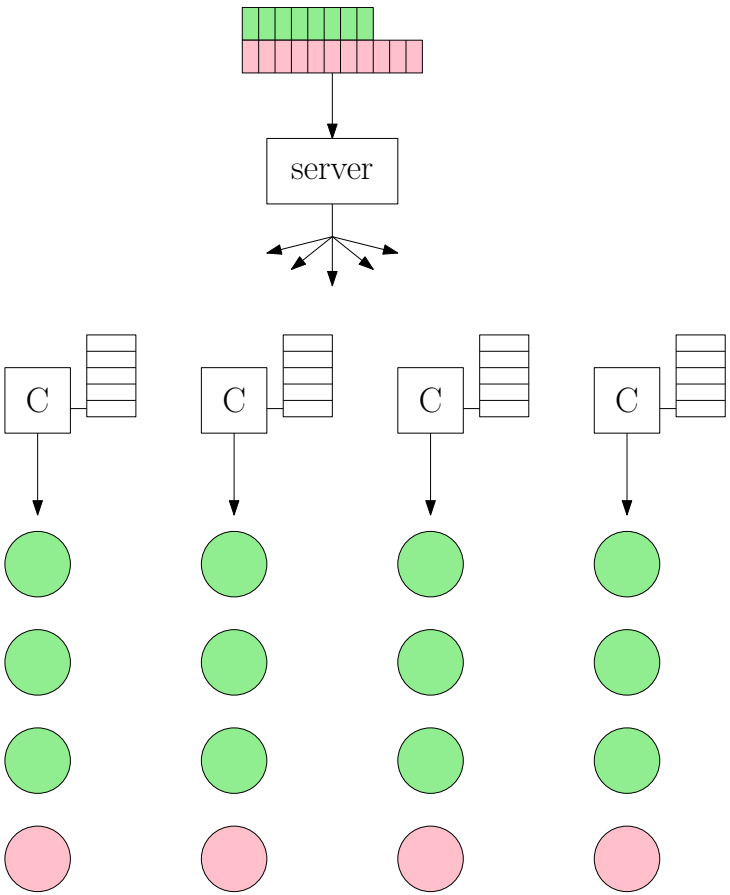
Same user profile at each cache
Predictability per cache



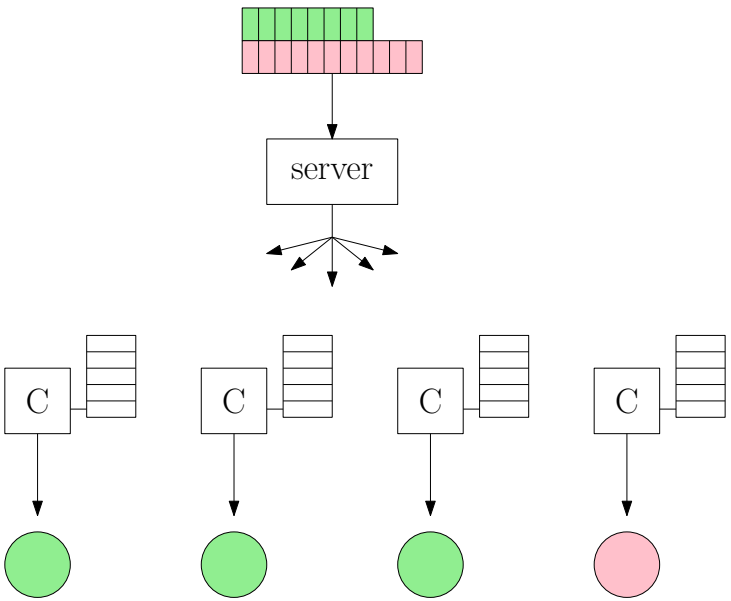
Different user profile at each cache
Predictability over all caches

Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%



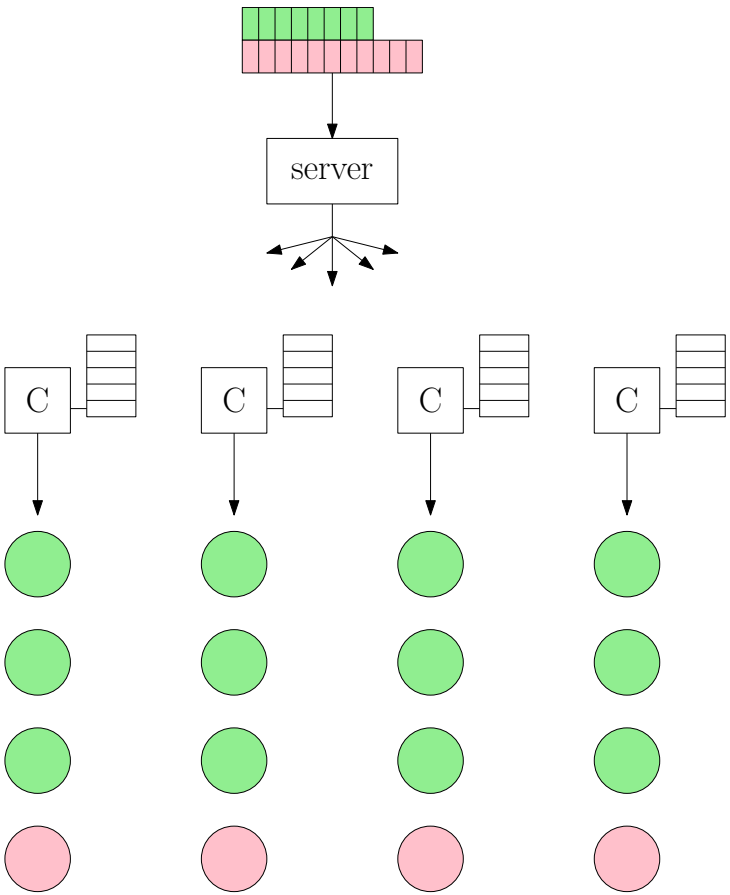
Same user profile at each cache
Predictability per cache



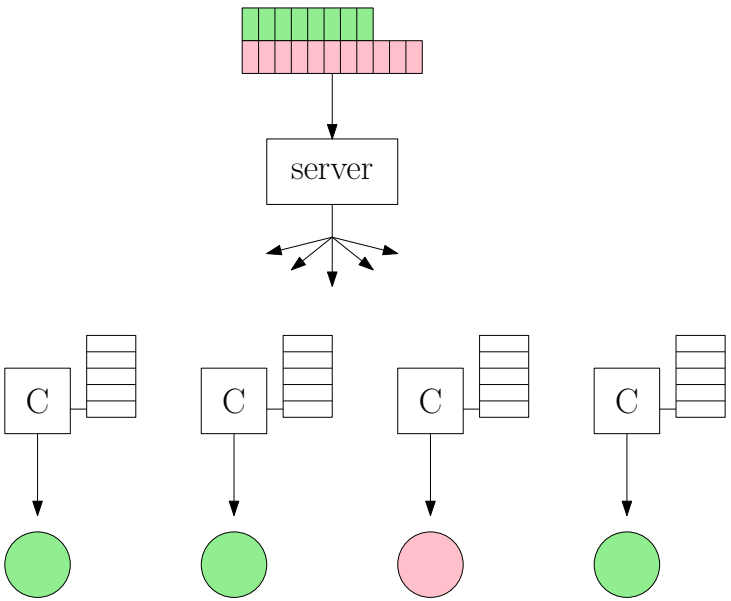
Different user profile at each cache
Predictability over all caches

Effect of number of users in multi-level coded caching

Example: 2 levels of files, dividing users into 75%–25%



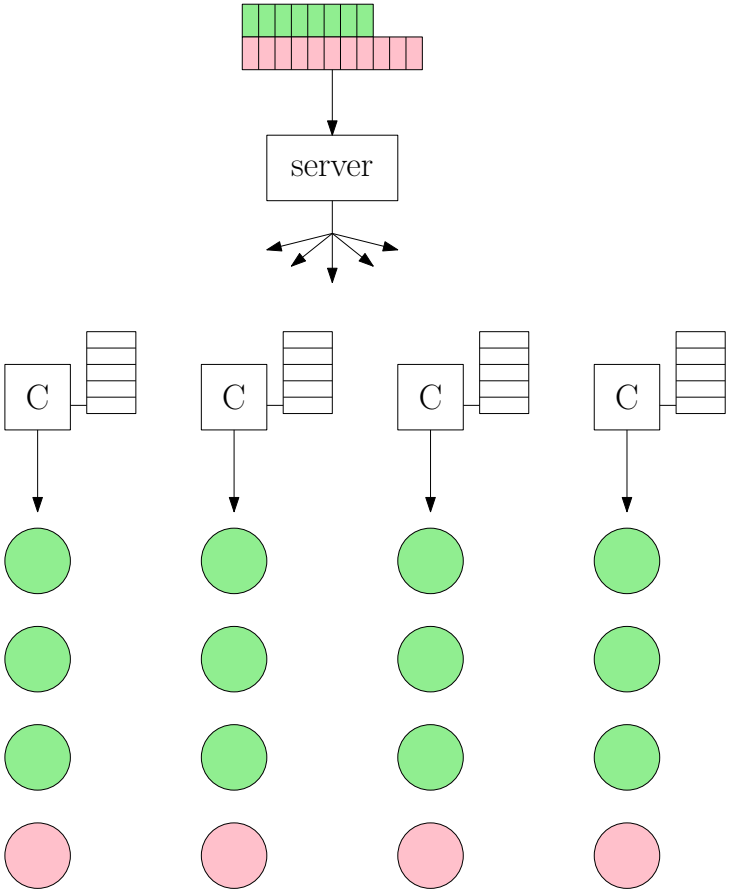
Same user profile at each cache
Predictability per cache

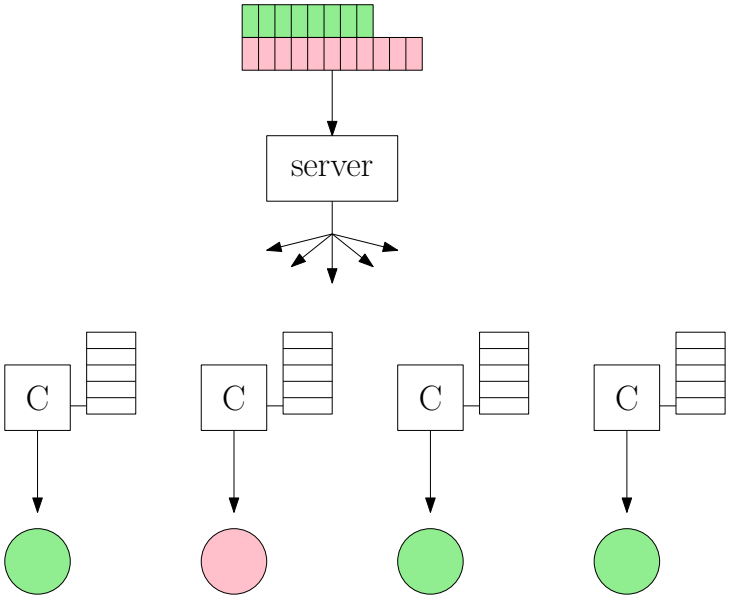


Different user profile at each cache
Predictability over all caches

Effect of number of users in multi-level coded caching

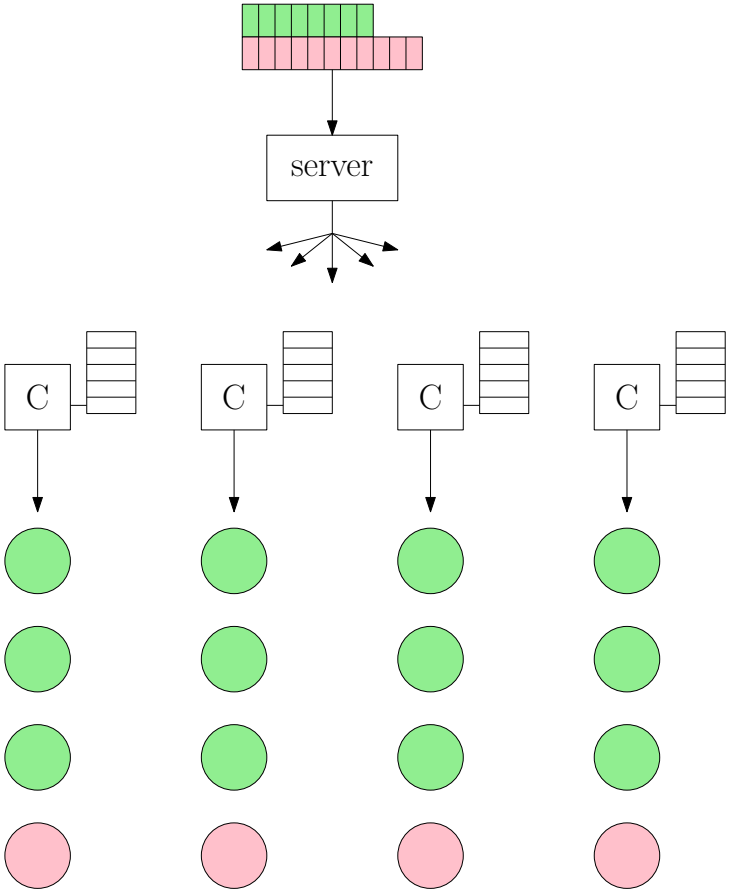
Main Question: Are these two setups fundamentally different?

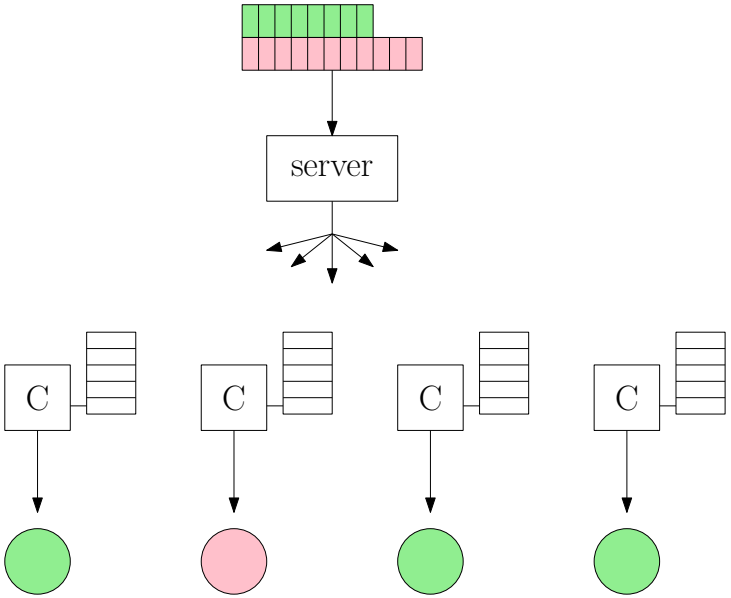




Effect of number of users in multi-level coded caching

Main Question: Are these two setups fundamentally different?





Yes! They require different strategies.

Main results

Multi-user setup

Single-user setup

Main results

Multi-user setup

- Memory-sharing strategy
  - *Separation* of levels

Single-user setup

Main results

Multi-user setup

- Memory-sharing strategy
  - *Separation* of levels

Single-user setup

- Threshold-and-cluster strategy
  - *Merging* of levels

Main results

Multi-user setup

- Memory-sharing strategy
  - *Separation* of levels
- Order-optimal

Single-user setup

- Threshold-and-cluster strategy
  - *Merging* of levels
- Order-optimal

Main results

Multi-user setup

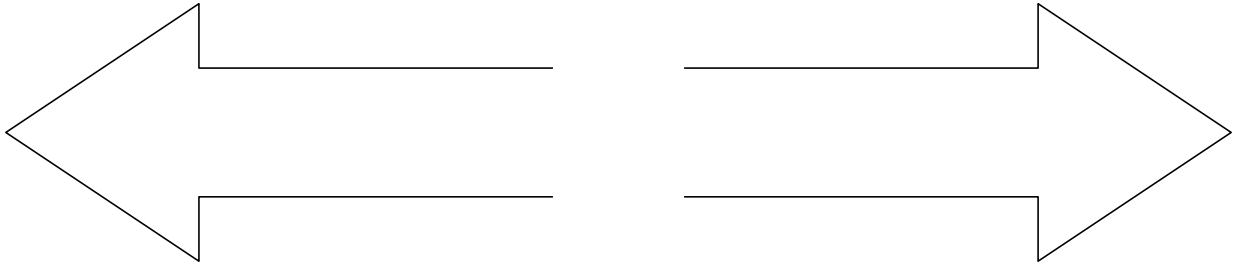
- Memory-sharing strategy
  - *Separation* of levels
- Order-optimal
- Threshold-and-cluster is inefficient

Single-user setup

- Threshold-and-cluster strategy
  - *Merging* of levels
- Order-optimal
- Memory-sharing is inefficient

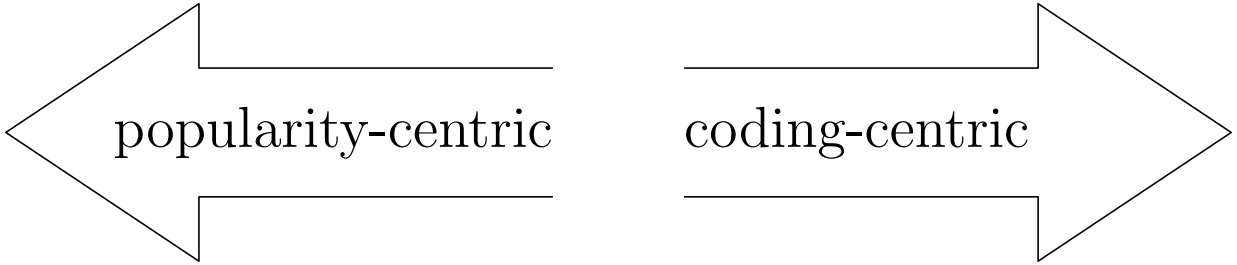
Why different strategies?

Two main forces drive any strategy:



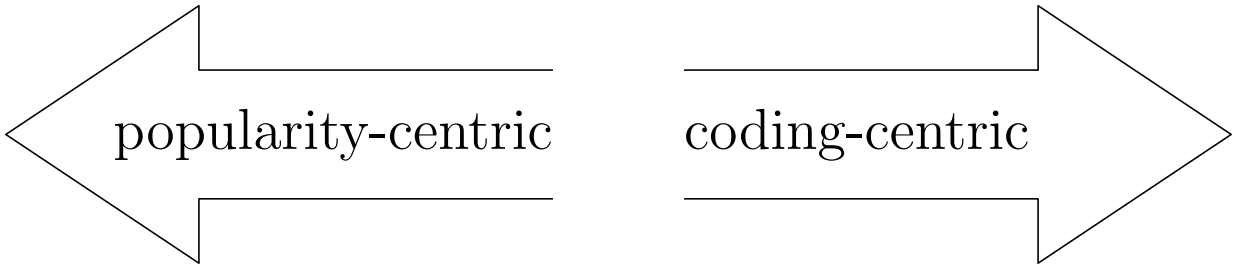
Why different strategies?

Two main forces drive any strategy:



Why different strategies?

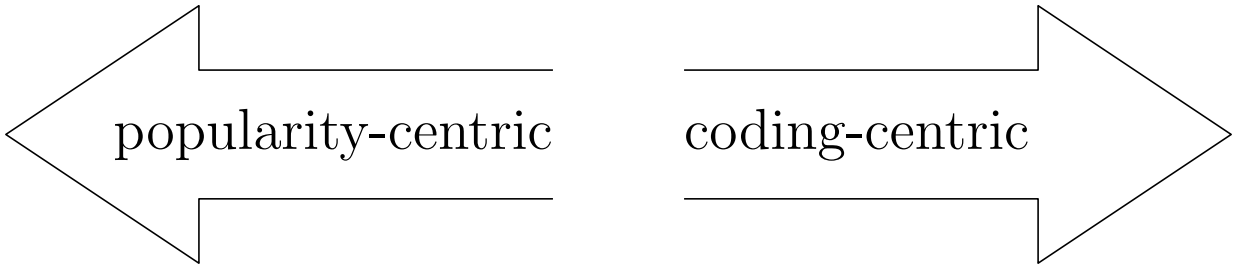
Two main forces drive any strategy:



- More popular files get more memory
- Coding opportunities maximized when files get same memory

Why different strategies?

Two main forces drive any strategy:



- More popular files get more memory
- No coding across popularity levels

- Coding opportunities maximized when files get same memory
- Code across popularity levels

Why different strategies?

Two main forces drive any strategy:

Separating levels ⟵ popularity-centric | coding-centric ⟶ Merging levels

- More popular files get more memory
- No coding across popularity levels

- Coding opportunities maximized when files get same memory
- Code across popularity levels

Why different strategies?

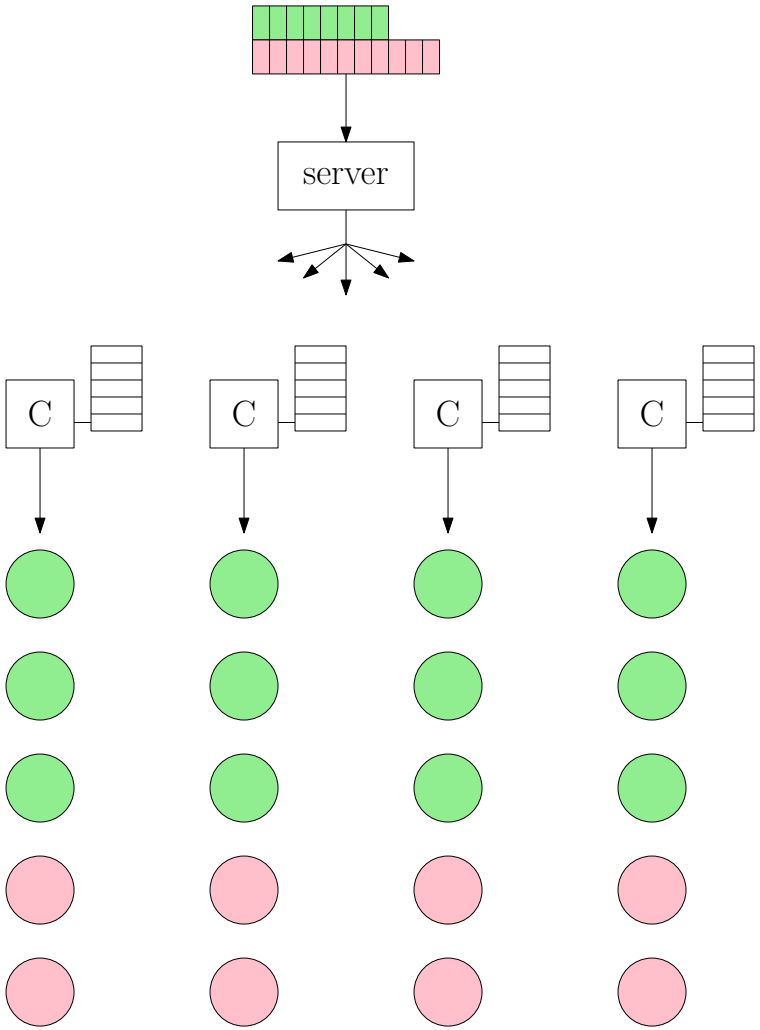
Two main forces drive any strategy:

Separating levels ⟵ popularity-centric | coding-centric ⟶ Merging levels

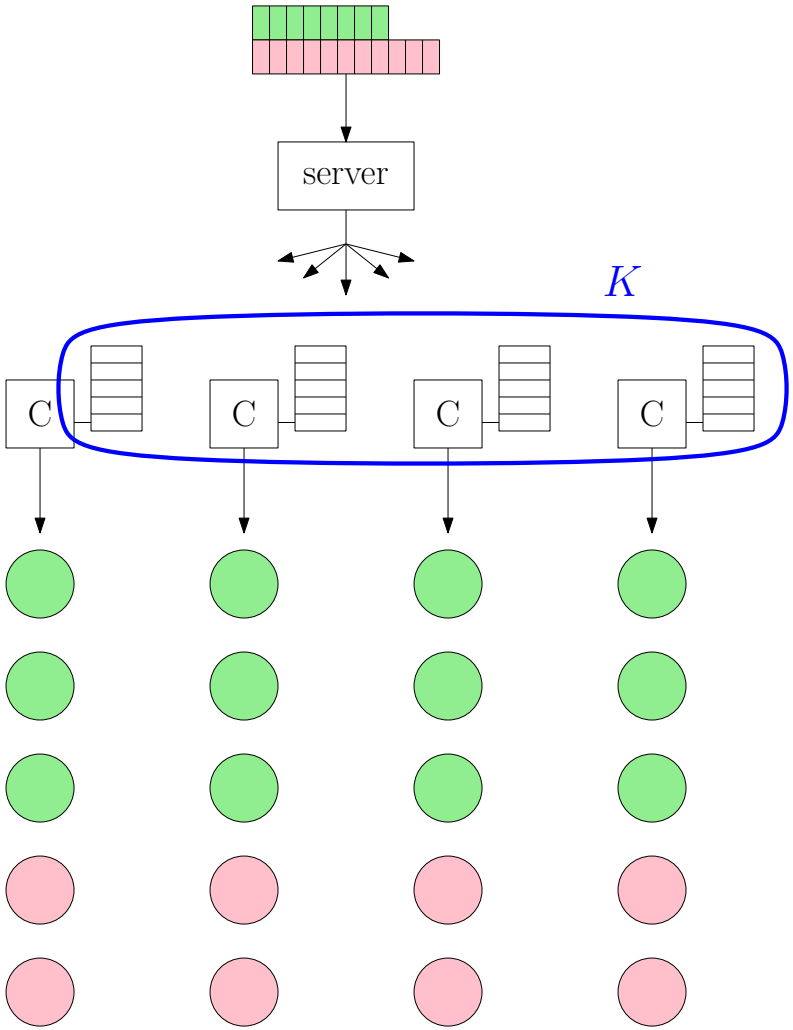
- More popular files get more memory
- No coding across popularity levels
- Favored by **Multi-user** setup

- Coding opportunities maximized when files get same memory
- Code across popularity levels
- Favored by **Single-user** setup

Multi-user setup (formal)

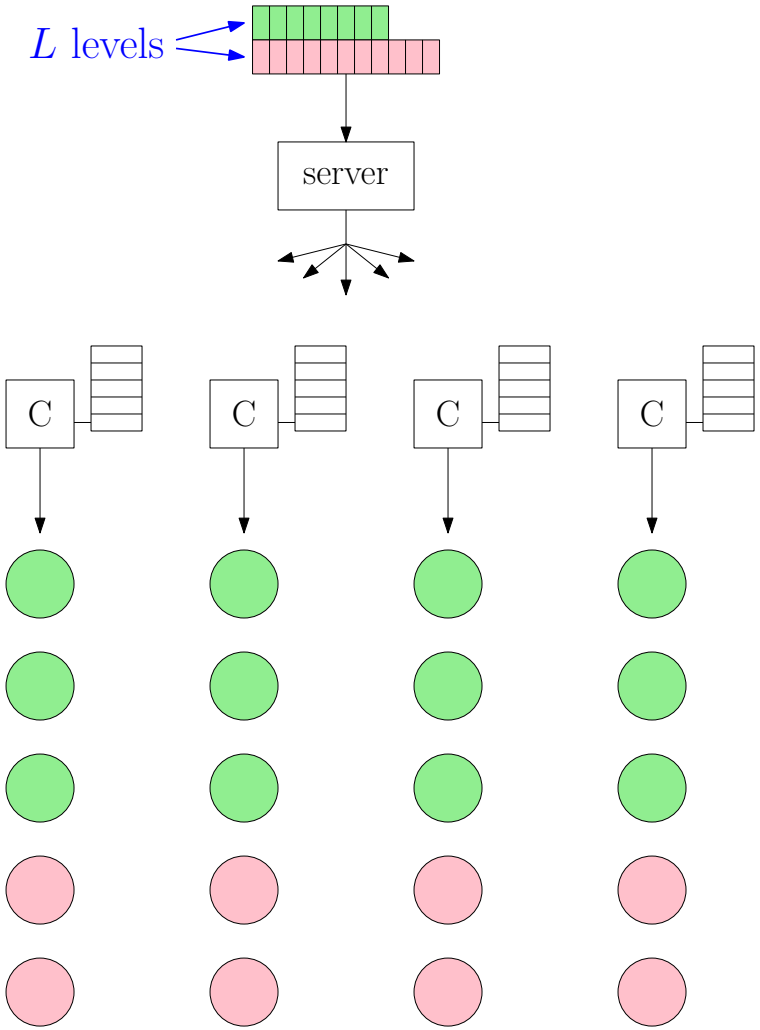


Multi-user setup (formal)



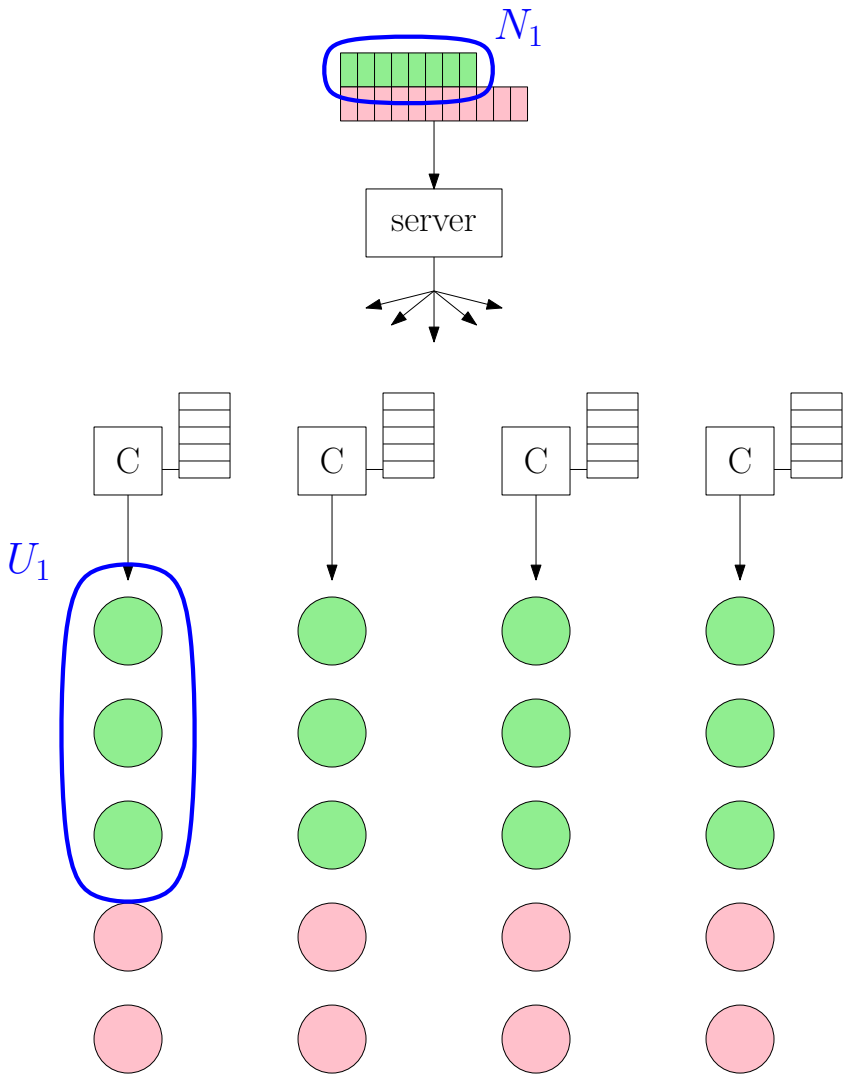
- # of caches

Multi-user setup (formal)



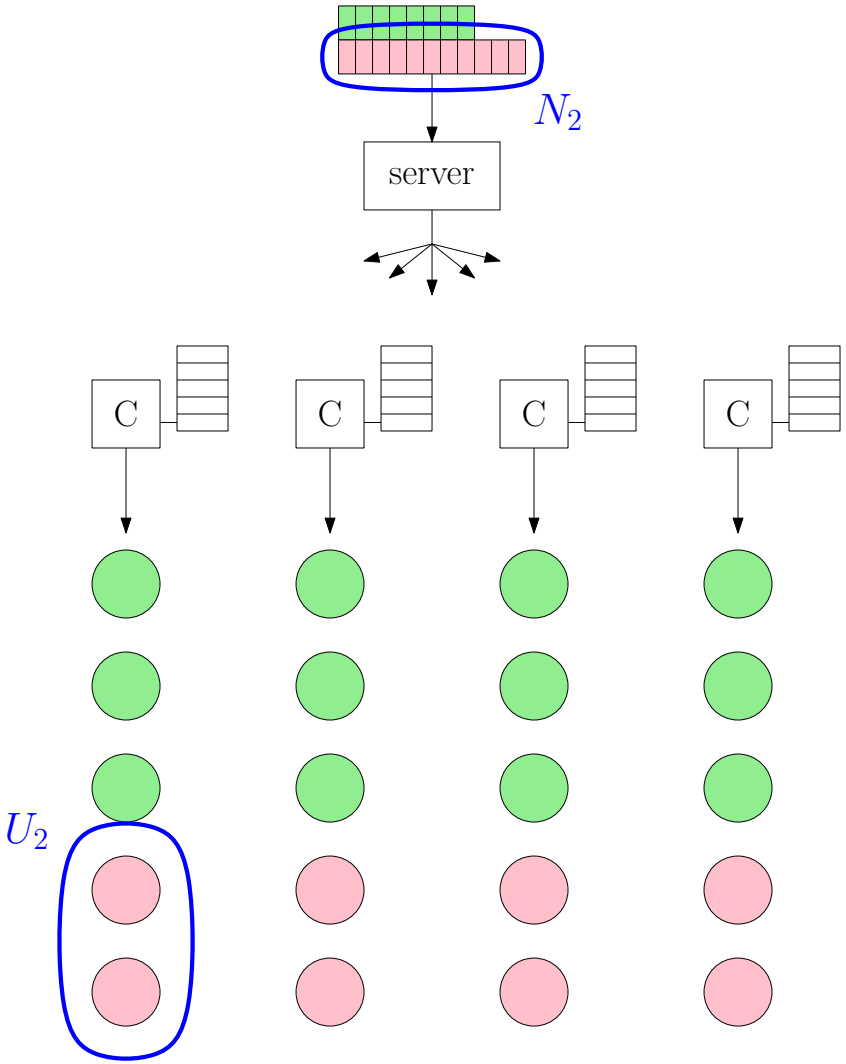
- # of caches
- # of levels

Multi-user setup (formal)



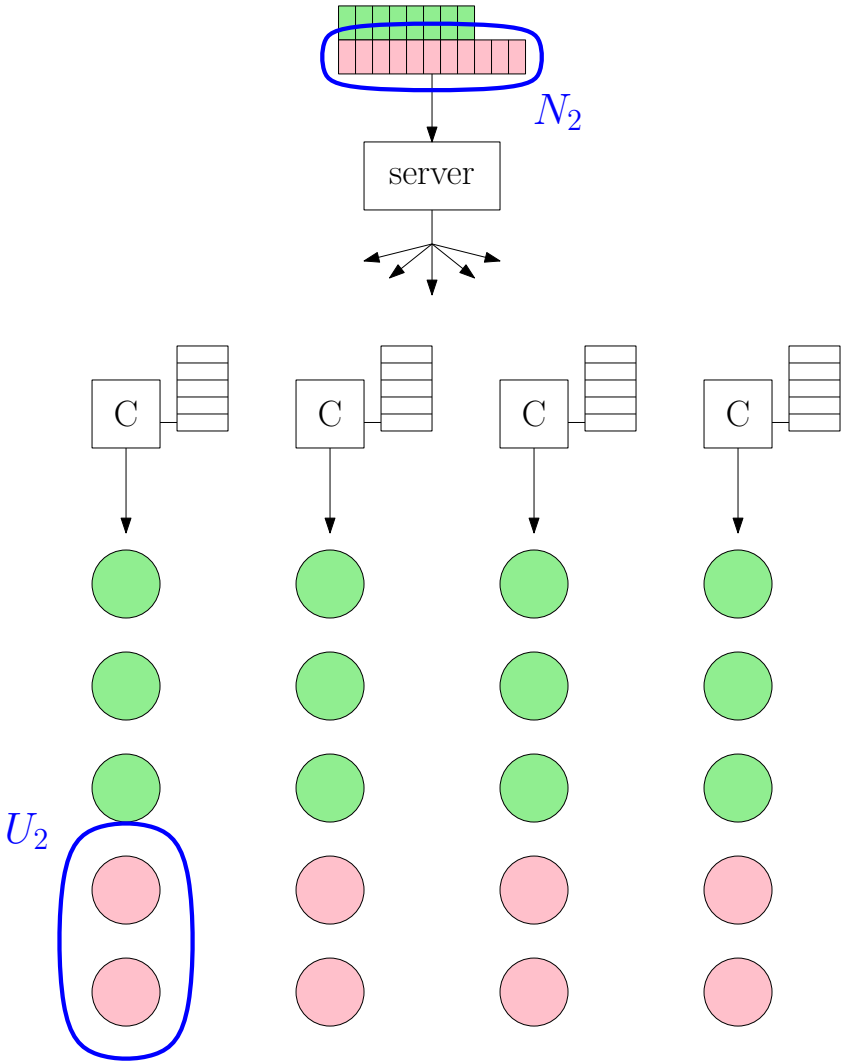
- # of caches
- # of levels
- For each level:
  - # of files
  - # of users per cache

Multi-user setup (formal)



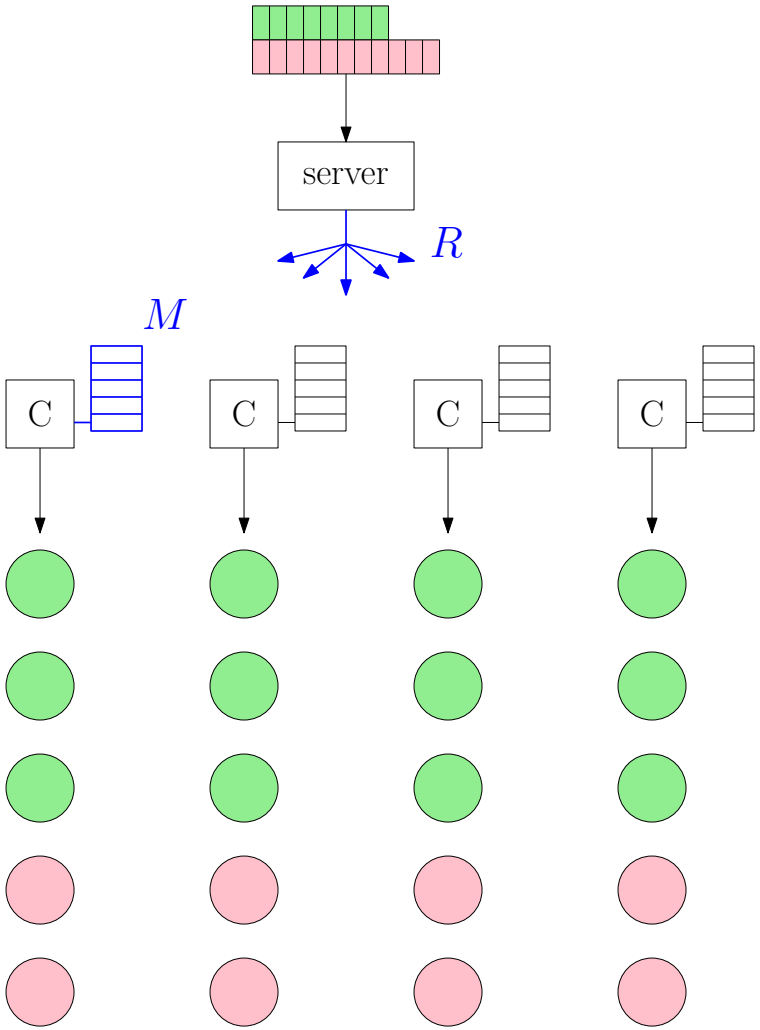
- # of caches
- # of levels
- For each level:
  - # of files
  - # of users per cache

Multi-user setup (formal)



- # of caches
- # of levels
- For each level:
  - # of files
  - # of users per cache
  - popularity $\propto$ users per file

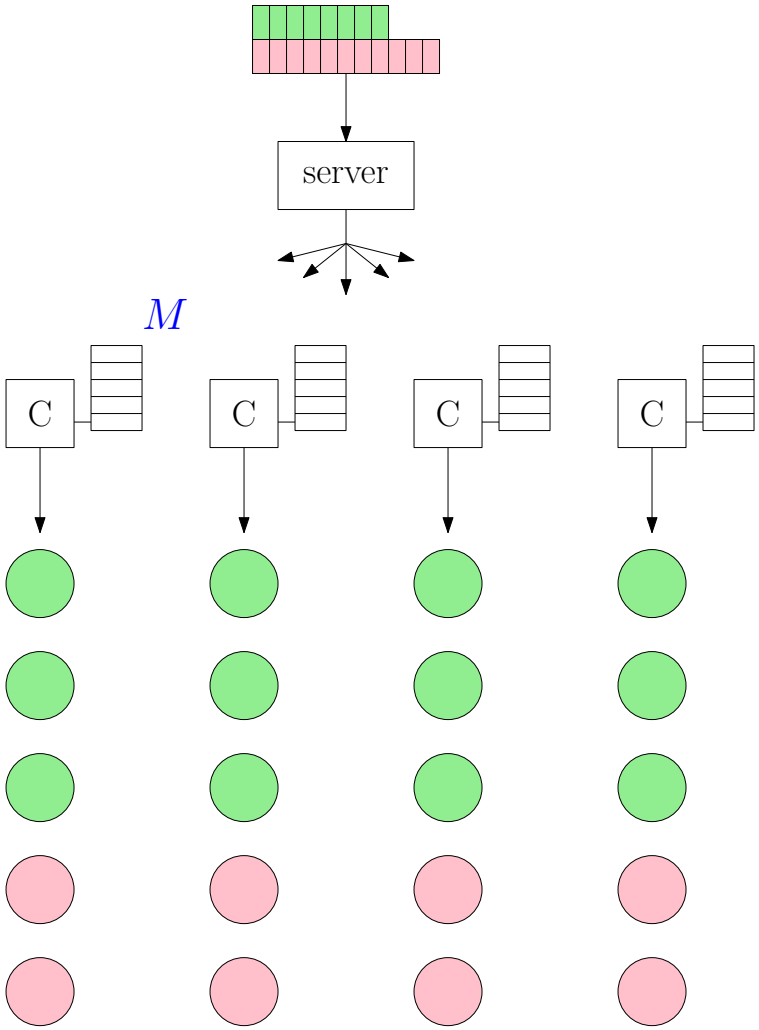
Multi-user setup (formal)



- # of caches
- # of levels
- For each level:
  - # of files
  - # of users per cache
  - popularity $\propto$ users per file
- Resources:
  - Cache memory M
  - Broadcast rate R

Strategy: memory-sharing

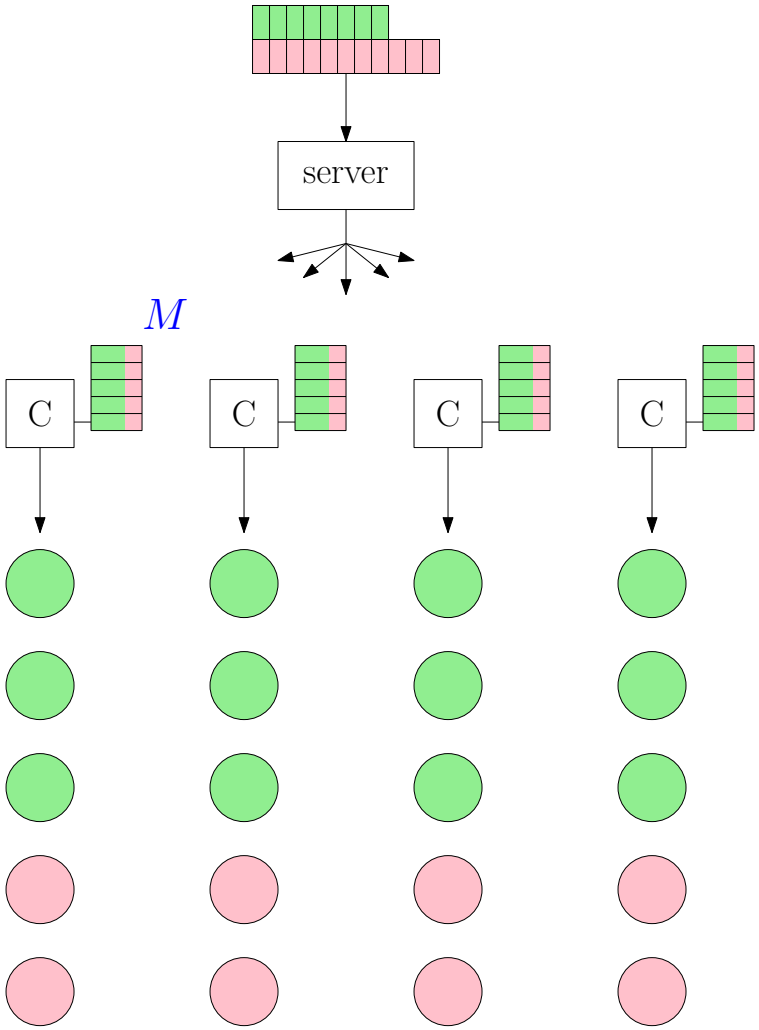
[H., K., D., 2014]



- Separate the popularity levels

Strategy: memory-sharing

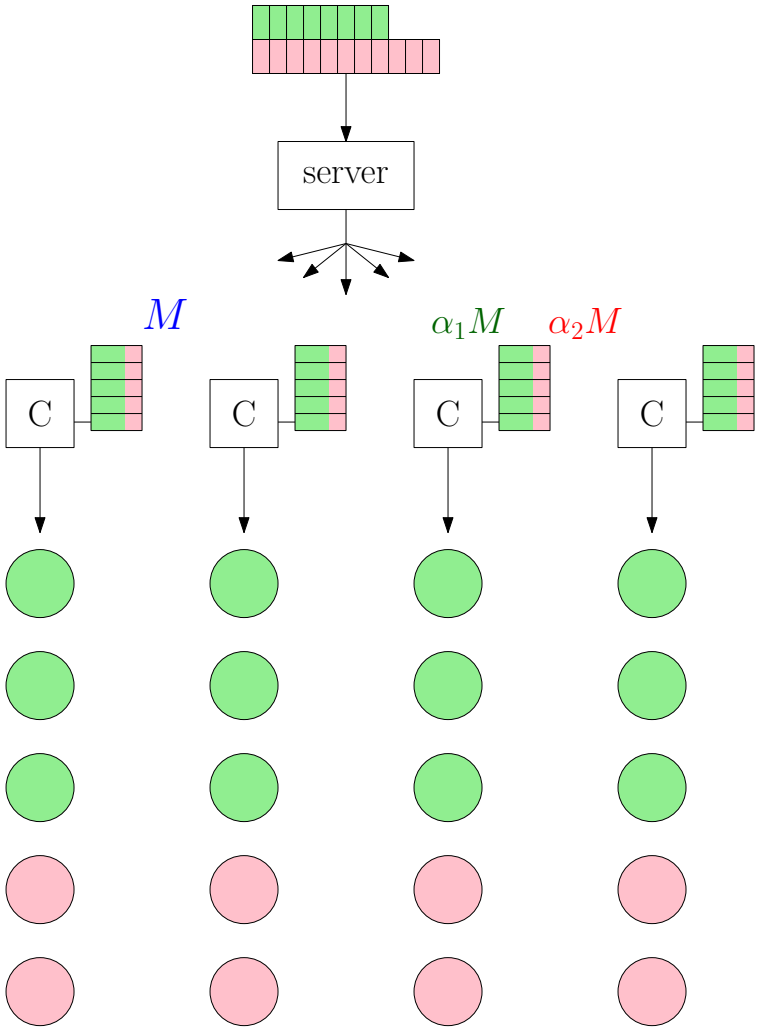
[H., K., D., 2014]



- Separate the popularity levels
- Share memory M between the levels

Strategy: memory-sharing

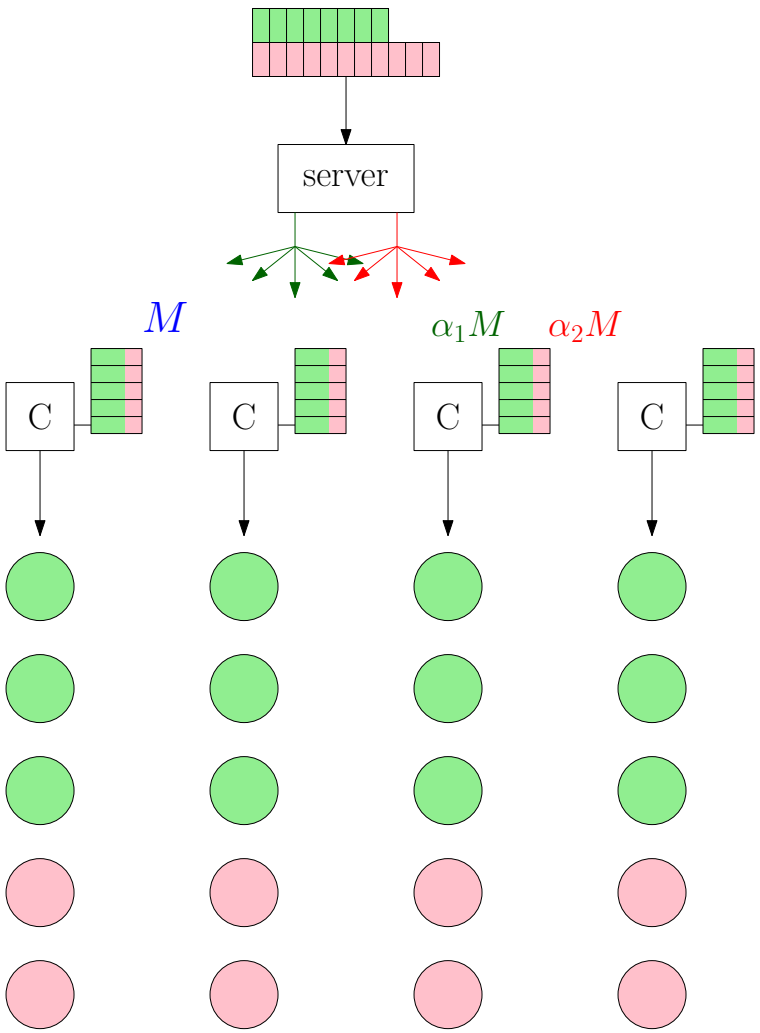
[H., K., D., 2014]



- Separate the popularity levels
- Share memory M between the levels
  - Level i gets memory $\alpha_i M$

Strategy: memory-sharing

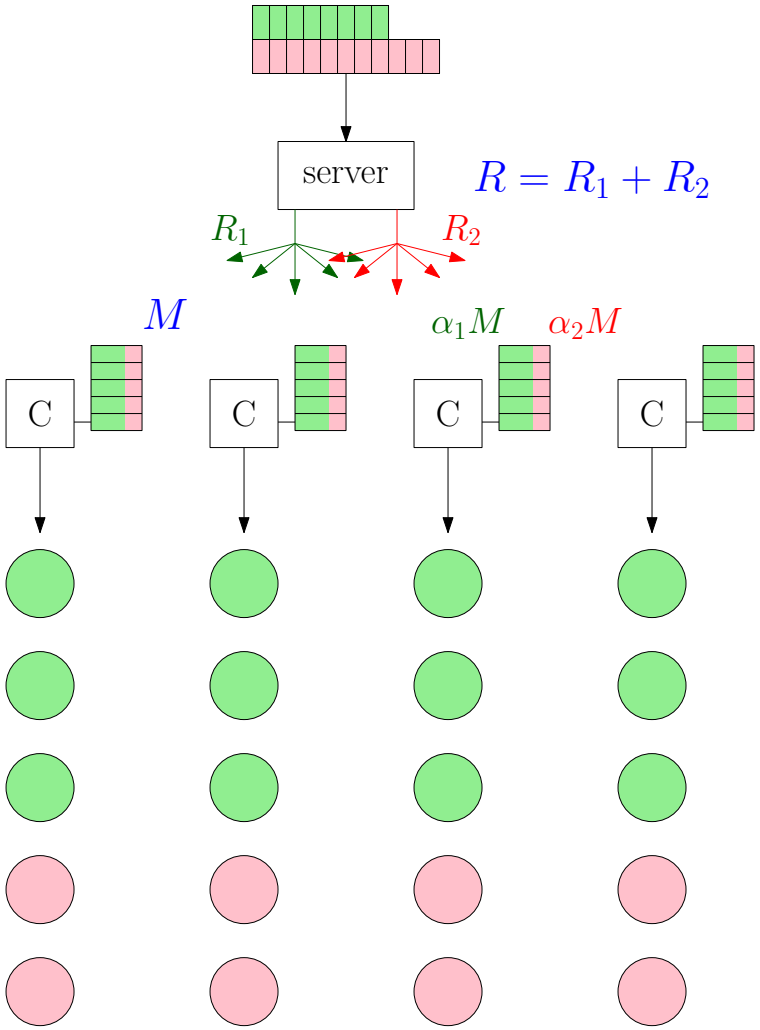
[H., K., D., 2014]



- Separate the popularity levels
- Share memory M between the levels
  - Level i gets memory $\alpha_i M$
- Perform separate placement and delivery for each level

Strategy: memory-sharing

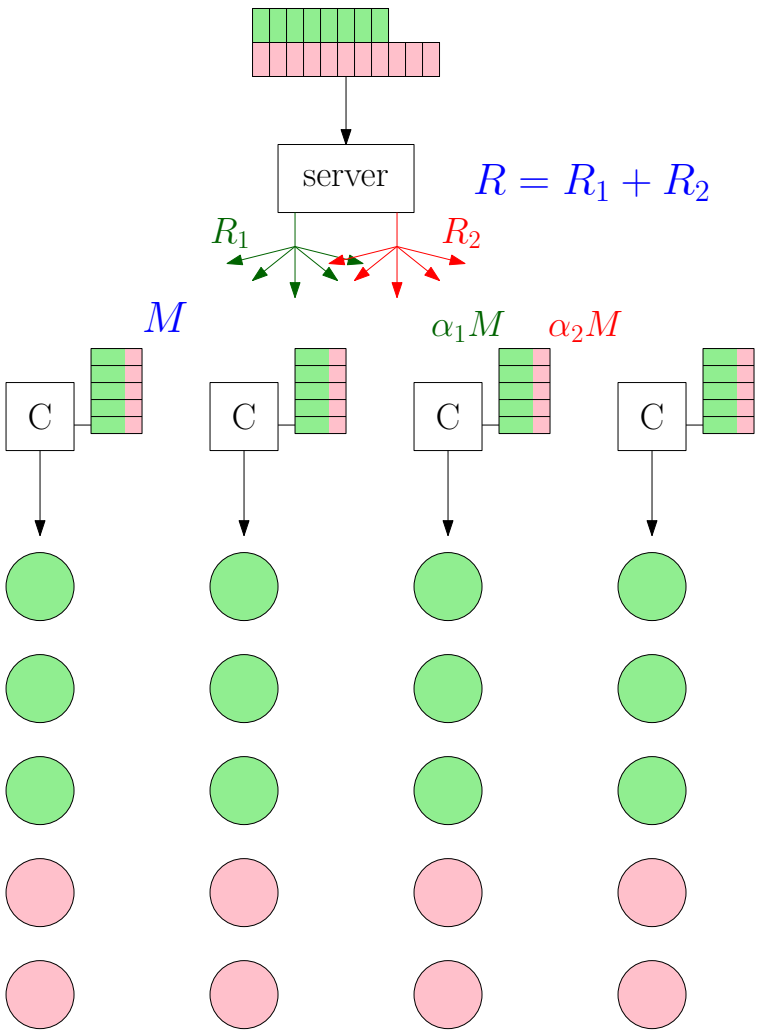
[H., K., D., 2014]



- Separate the popularity levels
- Share memory M between the levels
  - Level i gets memory $\alpha_i M$
- Perform separate placement and delivery for each level
- Total rate = sum of individual rates

Strategy: memory-sharing

[H., K., D., 2014]



- Separate the popularity levels
- Share memory M between the levels
  - Level i gets memory $\alpha_i M$
- Perform separate placement and delivery for each level
- Total rate = sum of individual rates
- Optimize over α_i's

Strategy: memory-sharing

[H., K., D., 2014]

3 sets of levels

Strategy: memory-sharing

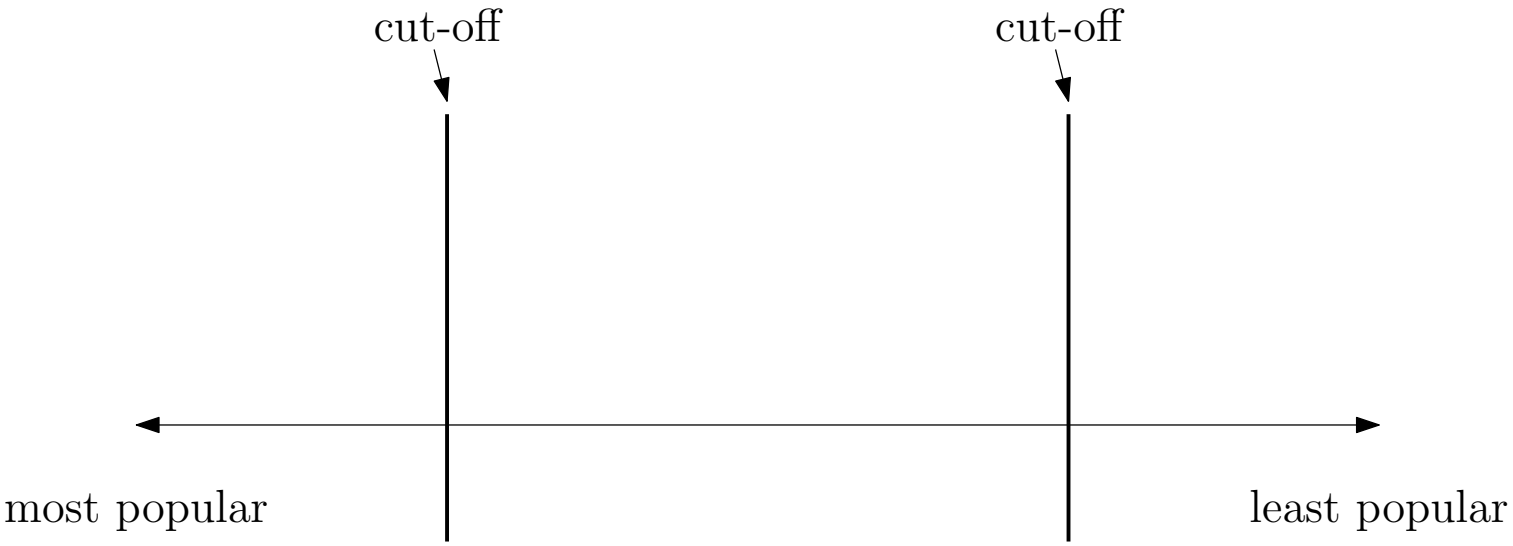
[H., K., D., 2014]

3 sets of levels

Strategy: memory-sharing

[H., K., D., 2014]

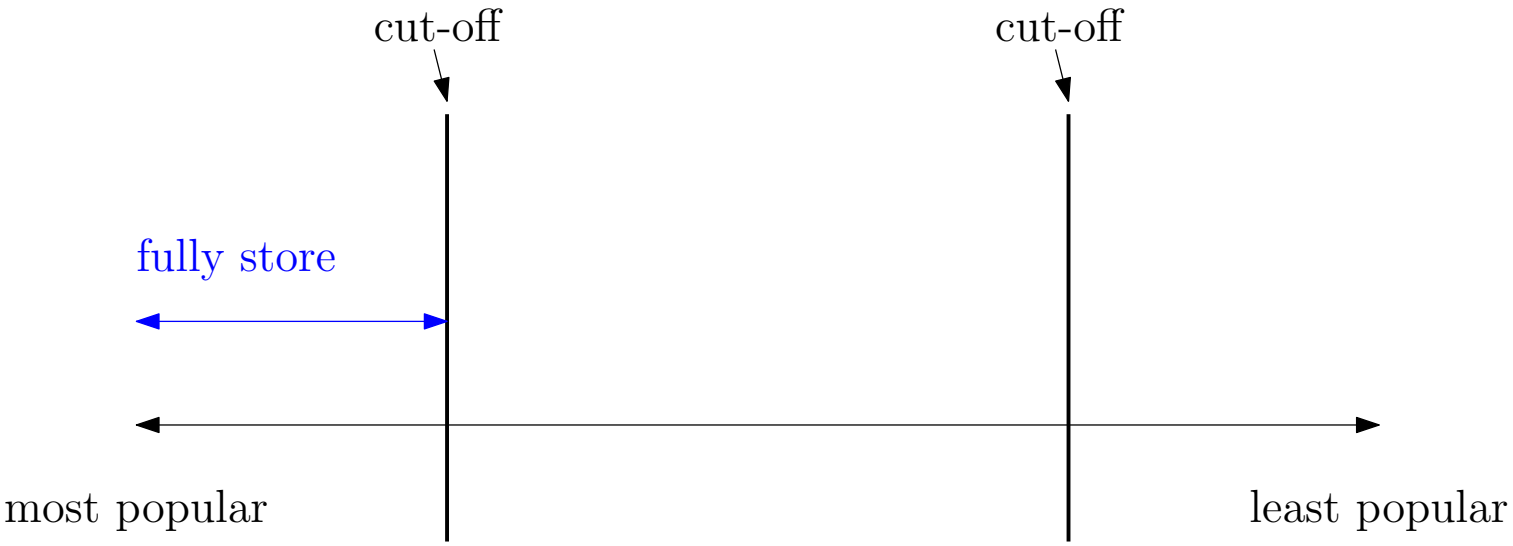
3 sets of levels



Strategy: memory-sharing

[H., K., D., 2014]

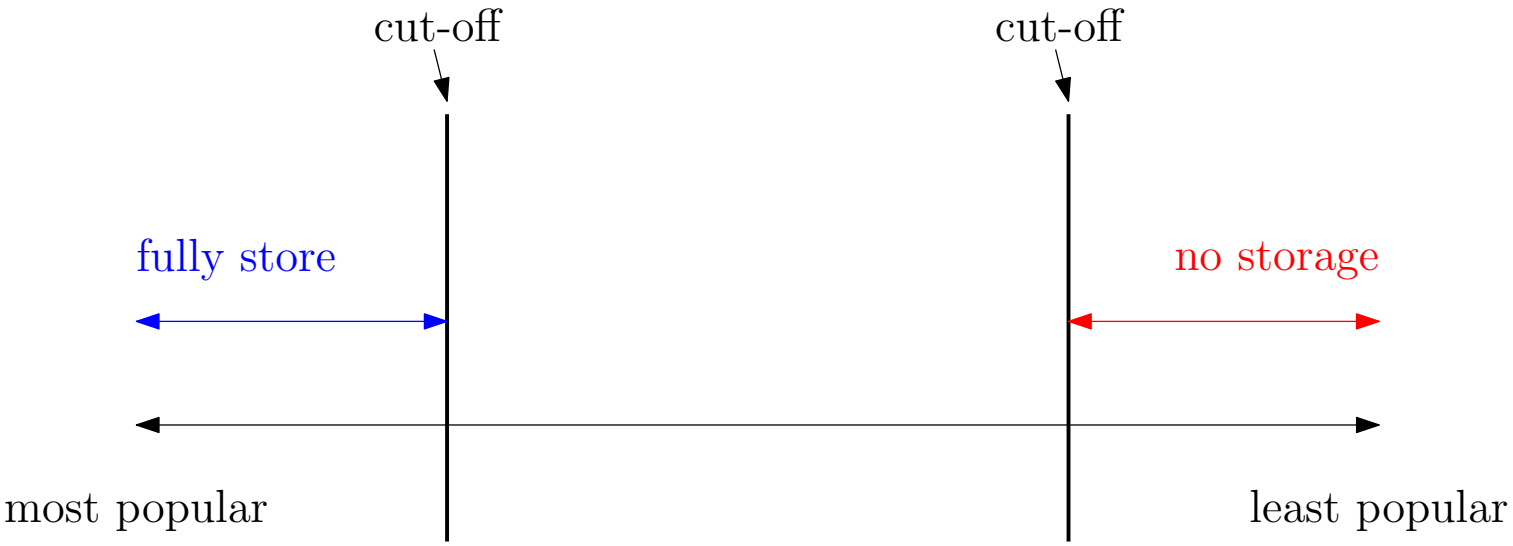
3 sets of levels



Strategy: memory-sharing

[H., K., D., 2014]

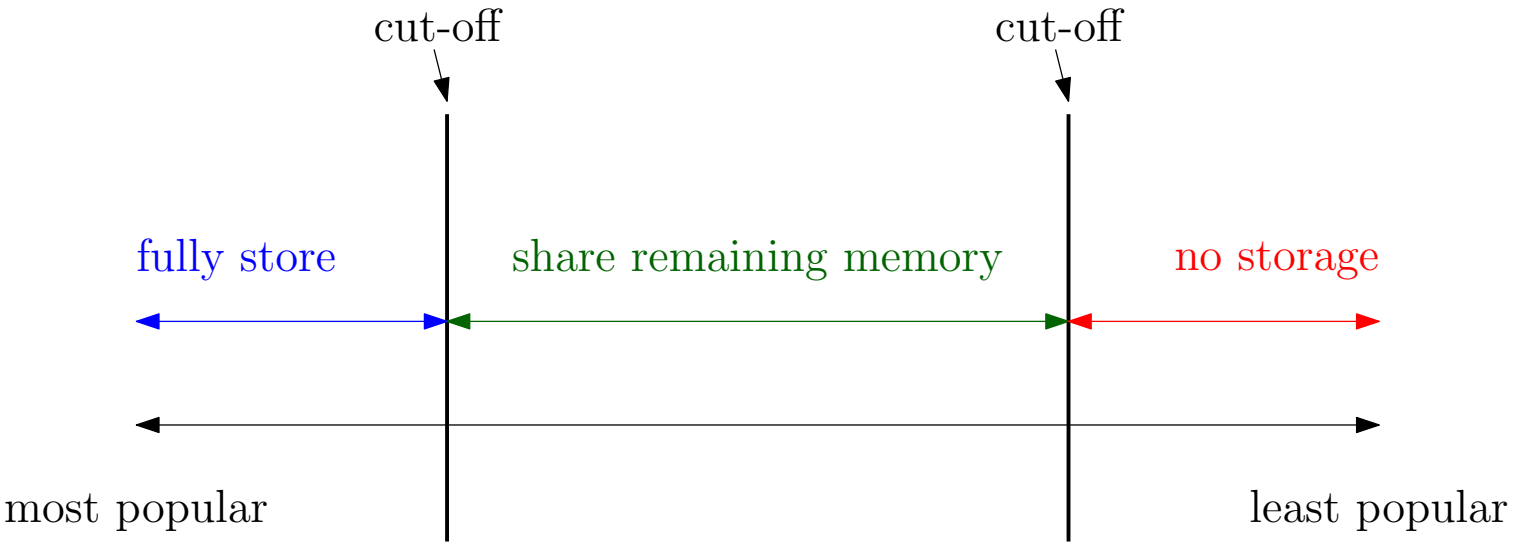
3 sets of levels



Strategy: memory-sharing

[H., K., D., 2014]

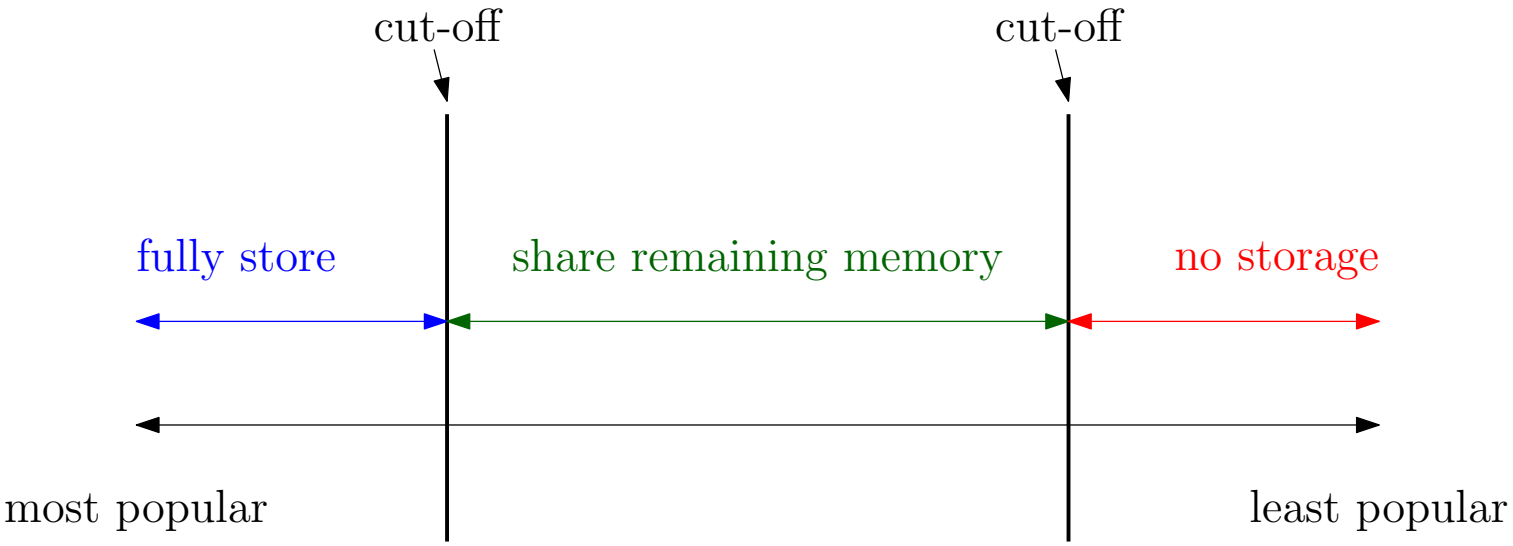
3 sets of levels



Strategy: memory-sharing

[H., K., D., 2014]

3 sets of levels



Achieved rate:

$$R \approx \sum_h KU_h + \frac{\left(\sum_i \sqrt{N_i U_i}\right)^2}{M - \sum_j N_j}$$

Order-optimality

Theorem (Order-optimality of memory-sharing for the multi-user setup)

In the **multi-user** *setup:*

$$\frac{\textit{rate achieved by } \textbf{memory-sharing}}{\textit{optimal rate } R^*} \leq c.$$

Order-optimality

Theorem (Order-optimality of memory-sharing for the multi-user setup)

In the **multi-user** *setup:*

$$\frac{\textit{rate achieved by } \textbf{memory-sharing}}{\textit{optimal rate } R^*} \leq c.$$

- c is independent of problem parameters

Order-optimality

Theorem (Order-optimality of memory-sharing for the multi-user setup)

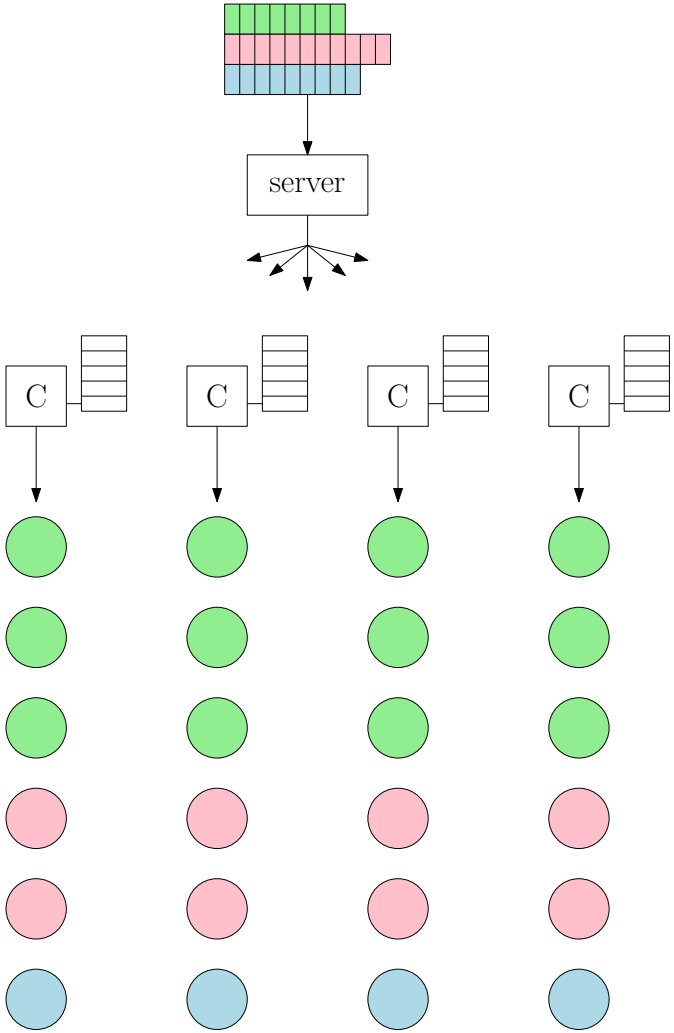
In the **multi-user** *setup:*

$$\frac{\textit{rate achieved by } \textbf{memory-sharing}}{\textit{optimal rate } R^*} \leq c.$$

- c is independent of problem parameters
- Proof: requires non-cut-set lower bounds on R^*

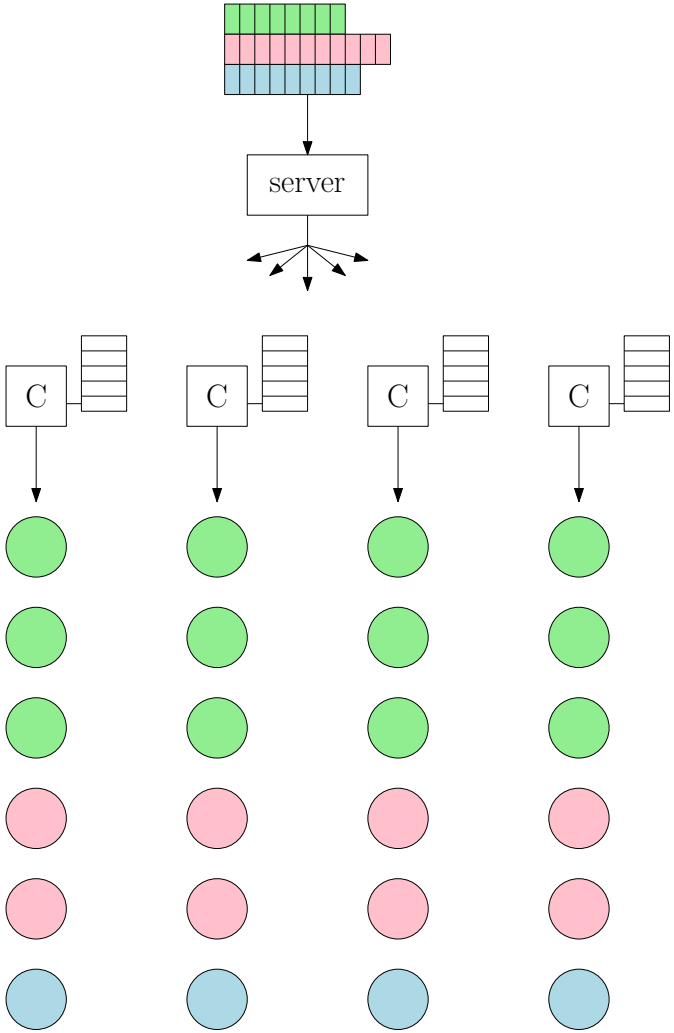
Lower bounds for multi-user setup

- Capture:
  - Necessity of level separation
  - All levels present at each cache



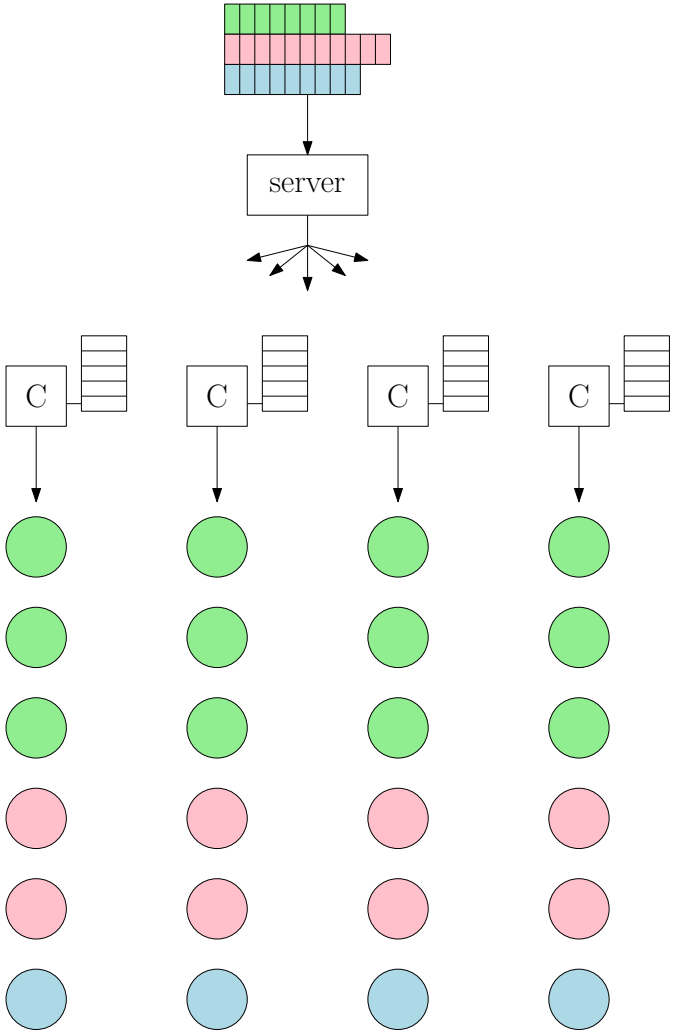
Lower bounds for multi-user setup

- Capture:
  - Necessity of level separation
  - All levels present at each cache
- Achieve: $R = R_1 + R_2 + R_3$



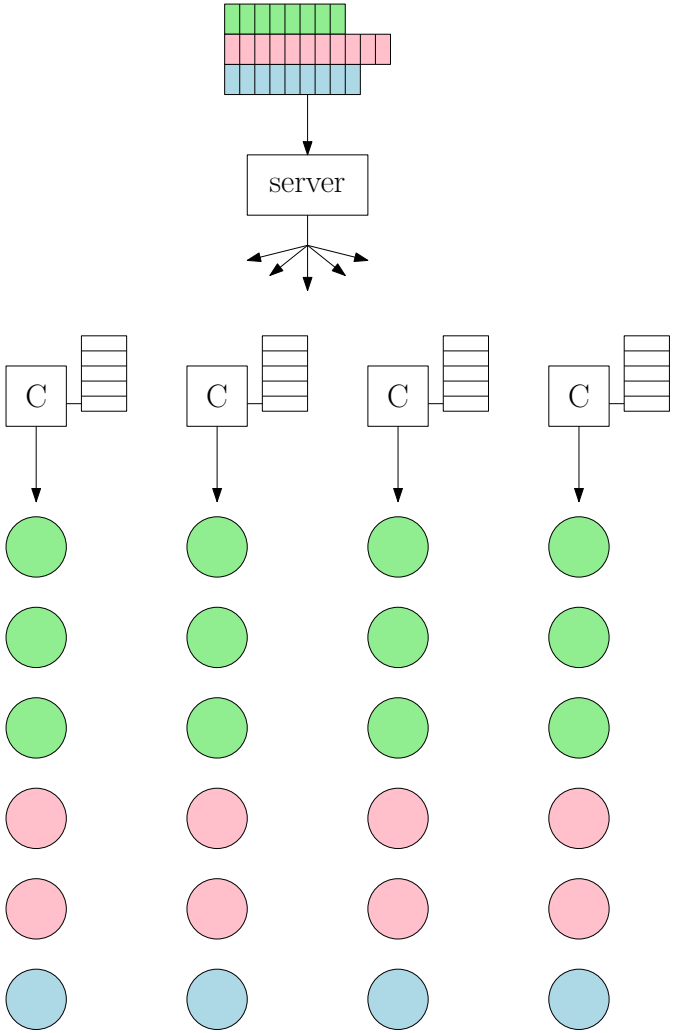
Lower bounds for multi-user setup

- Capture:
  - Necessity of level separation
  - All levels present at each cache
- Achieve: $R = R_1 + R_2 + R_3$
- Want: $R^* \geq aR_1 + bR_2 + cR_3$



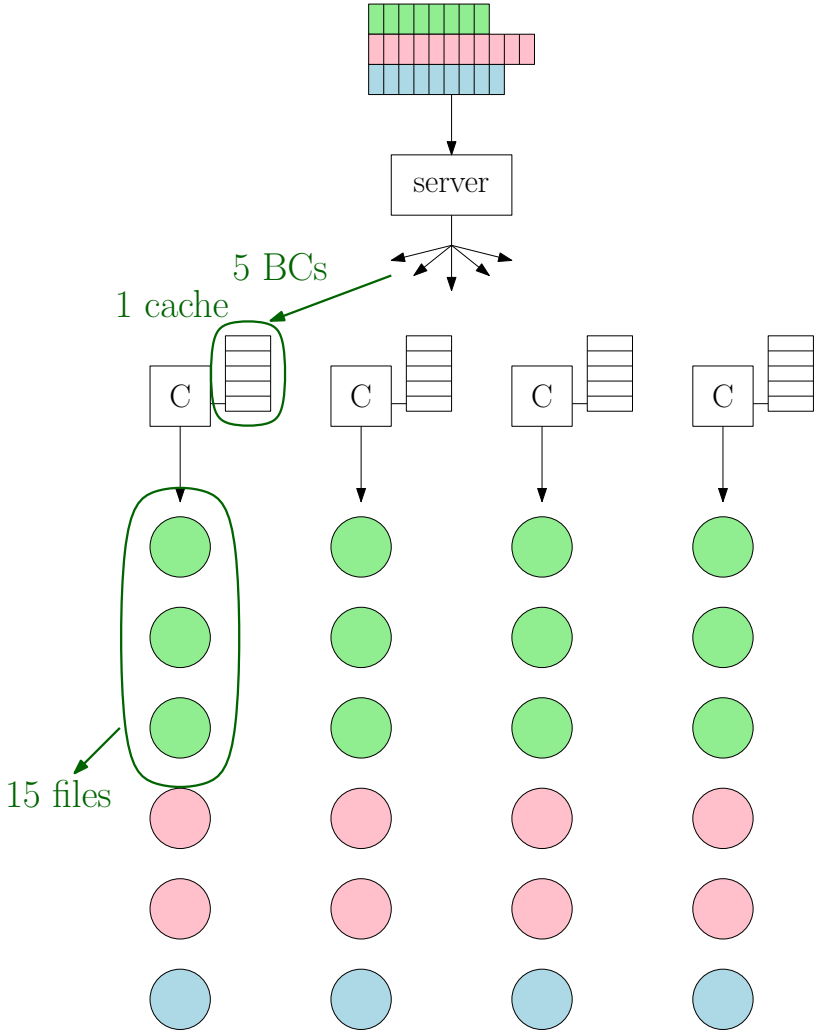
Lower bounds for multi-user setup

- Capture:
  - Necessity of level separation
  - All levels present at each cache
- Achieve: $R = R_1 + R_2 + R_3$
- Want: $R^* \geq aR_1 + bR_2 + cR_3$
- Individual terms can be derived using cut-set bounds



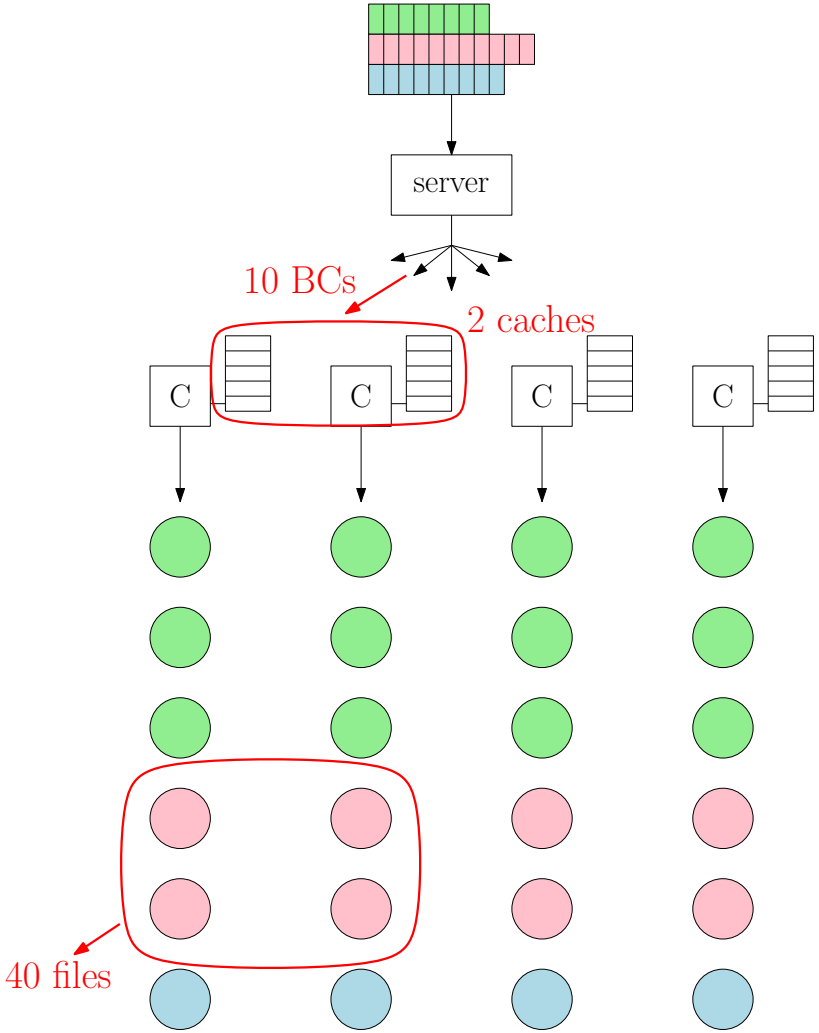
Lower bounds for multi-user setup

- Capture:
  - Necessity of level separation
  - All levels present at each cache
- Achieve: $R = R_1 + R_2 + R_3$
- Want: $R^* \geq aR_1 + bR_2 + cR_3$
- Individual terms can be derived using cut-set bounds



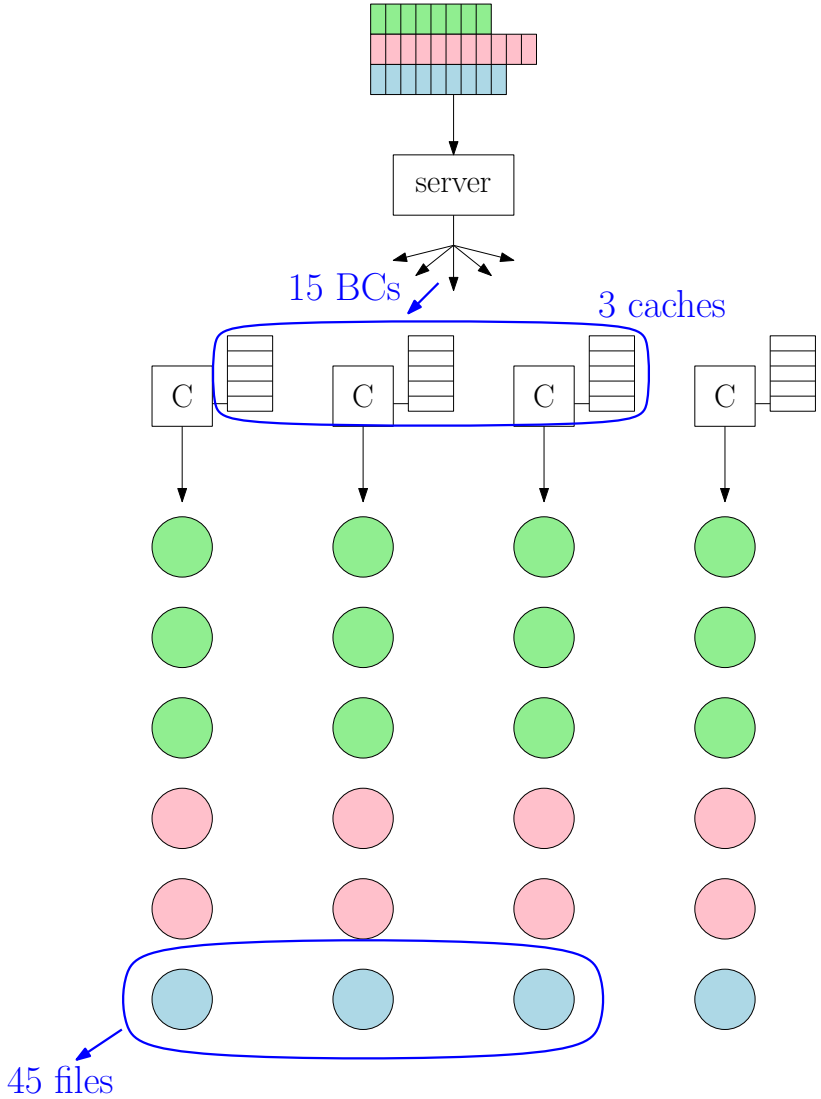
Lower bounds for multi-user setup

- Capture:
  - Necessity of level separation
  - All levels present at each cache
- Achieve: $R = R_1 + R_2 + R_3$
- Want: $R^* \geq aR_1 + bR_2 + cR_3$
- Individual terms can be derived using cut-set bounds



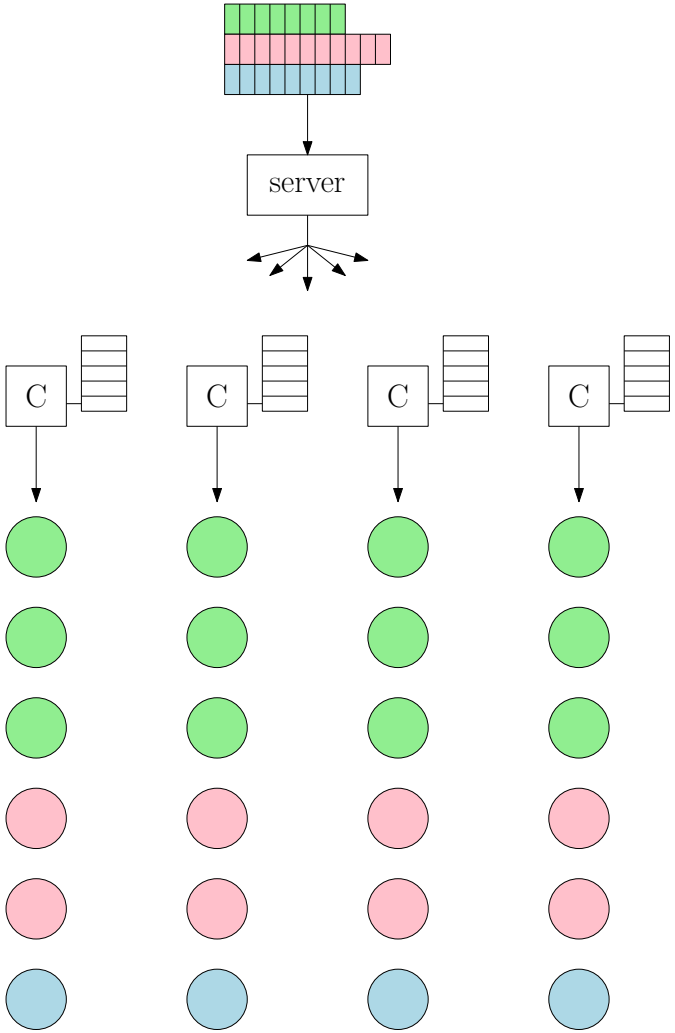
Lower bounds for multi-user setup

- Capture:
  - Necessity of level separation
  - All levels present at each cache
- Achieve: $R = R_1 + R_2 + R_3$
- Want: $R^* \geq aR_1 + bR_2 + cR_3$
- Individual terms can be derived using cut-set bounds



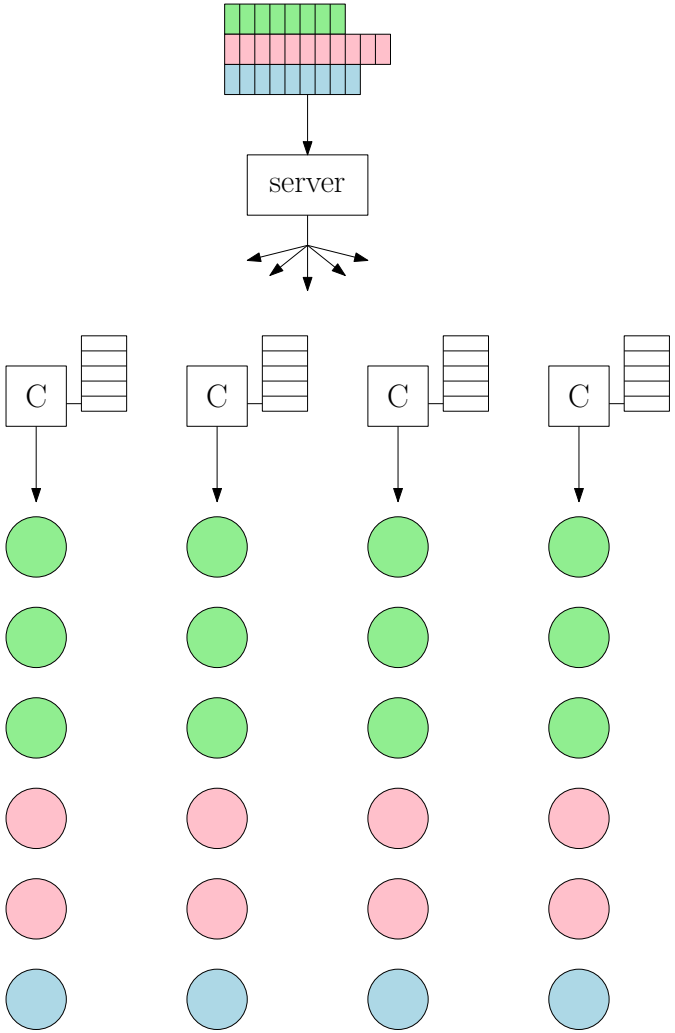
Lower bounds for multi-user setup

- Capture:
  - Necessity of level separation
  - All levels present at each cache
- Achieve: $R = R_1 + R_2 + R_3$
- Want: $R^* \geq aR_1 + bR_2 + cR_3$
- Individual terms can be derived using cut-set bounds
- Combine?



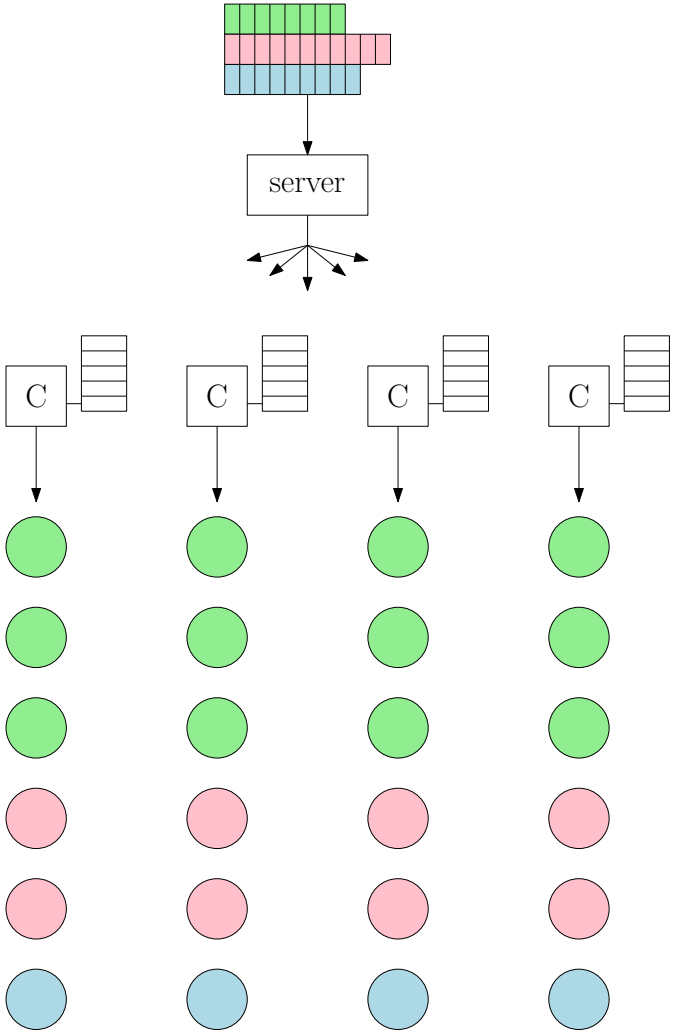
Lower bounds for multi-user setup

- Capture:
  - Necessity of level separation
  - All levels present at each cache
- Achieve: $R = R_1 + R_2 + R_3$
- Want: $R^* \geq aR_1 + bR_2 + cR_3$
- Individual terms can be derived using cut-set bounds
- Combine?
  - Sliding-window subset entropy inequality [Jiang *et al.*, 2011]

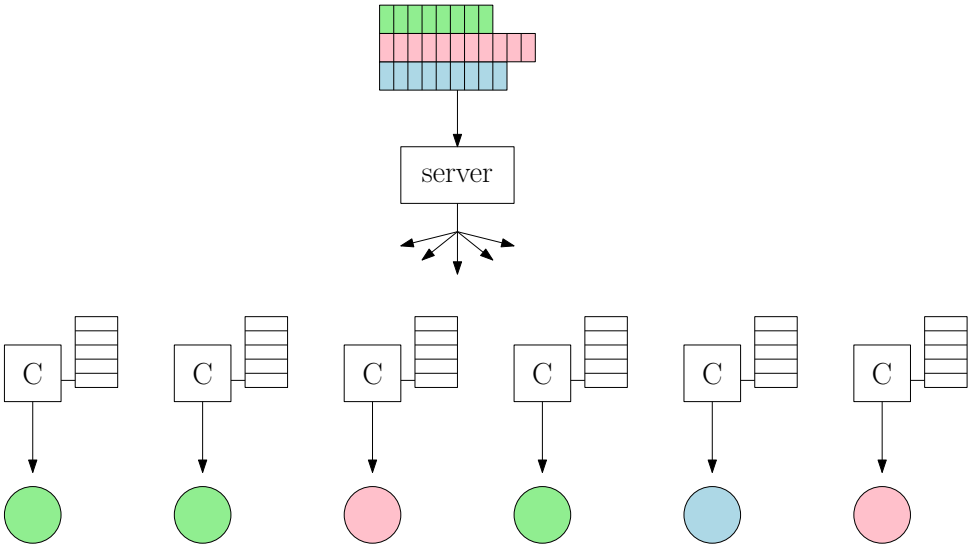


Lower bounds for multi-user setup

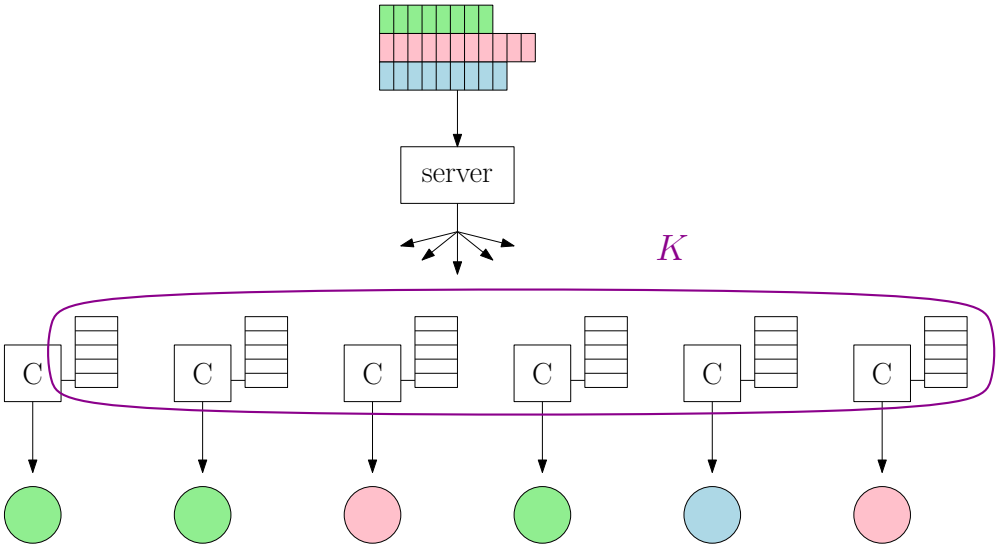
- Capture:
  - Necessity of level separation
  - All levels present at each cache
- Achieve: $R = R_1 + R_2 + R_3$
- Want: $R^* \geq aR_1 + bR_2 + cR_3$
- Individual terms can be derived using cut-set bounds
- Combine?
  - Sliding-window subset entropy inequality [Jiang *et al.*, 2011]
  - $\implies$ "non-cut-set" lower bounds that account for each level's allocated memory, **without any restriction on the scheme**



Single-user setup (formal)

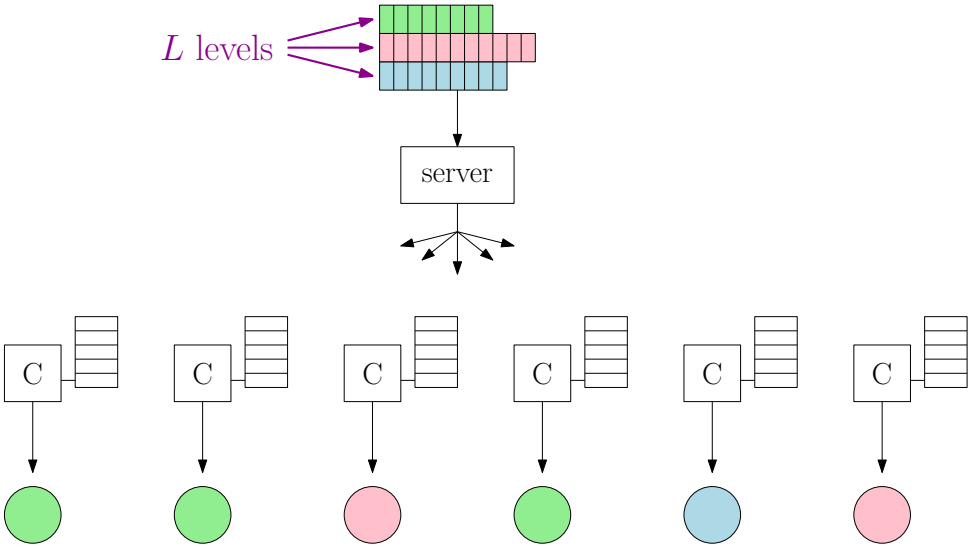


Single-user setup (formal)



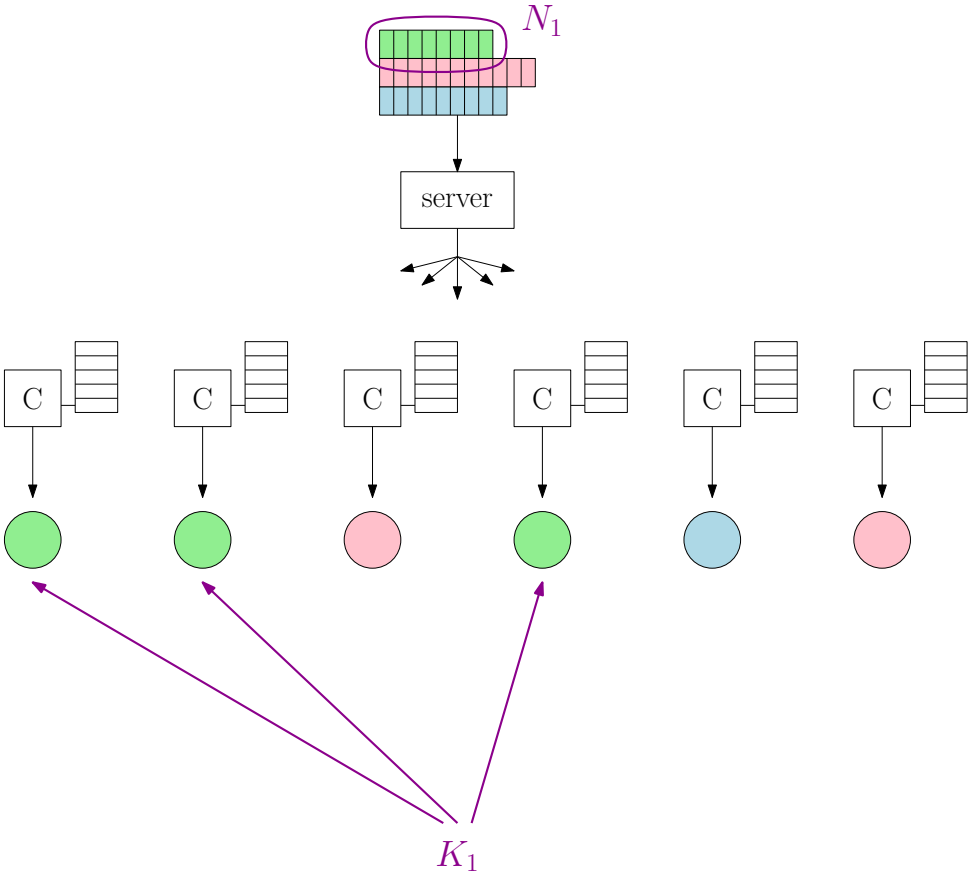
- # of caches
- one user per cache

Single-user setup (formal)



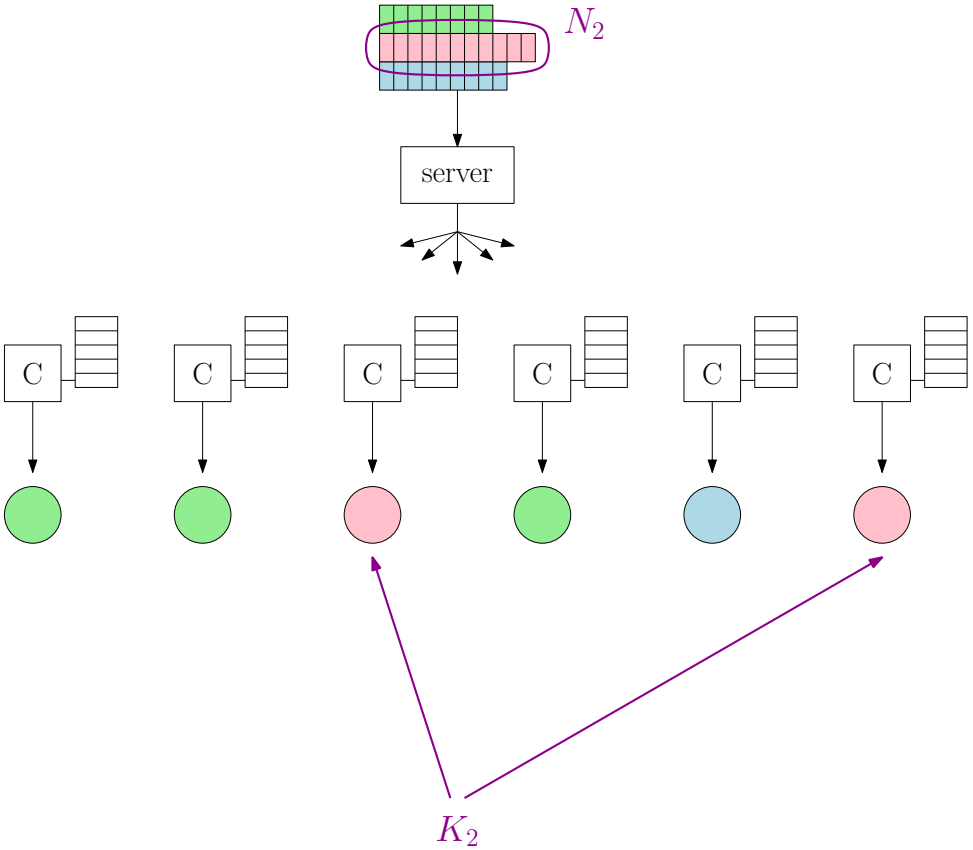
- # of caches
- one user per cache
- # of levels

Single-user setup (formal)



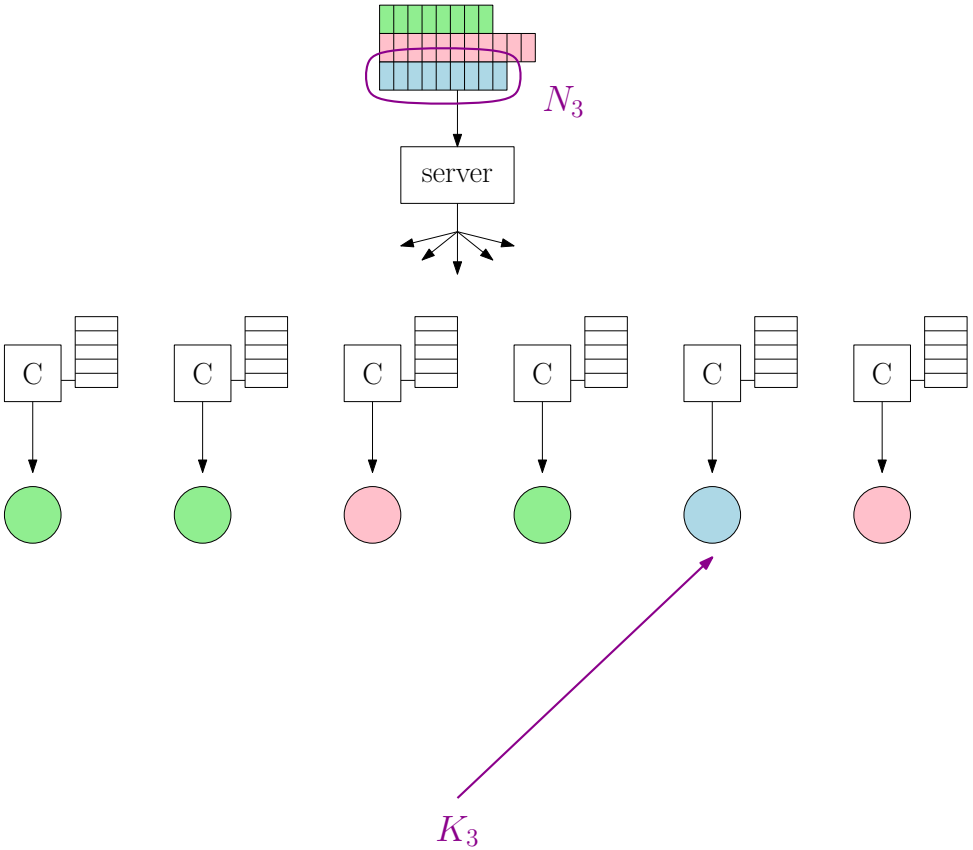
- # of caches
- one user per cache
- # of levels
- For each level:
  - # of files
  - # of users

Single-user setup (formal)



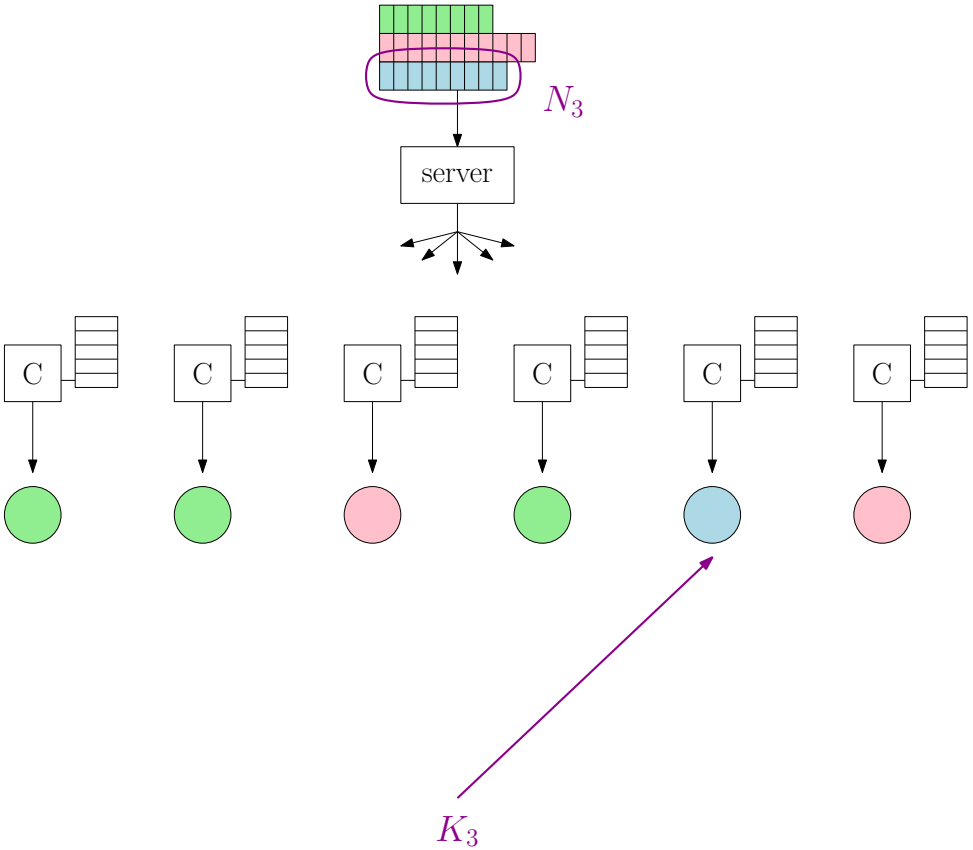
- # of caches
- one user per cache
- # of levels
- For each level:
  - # of files
  - # of users

Single-user setup (formal)



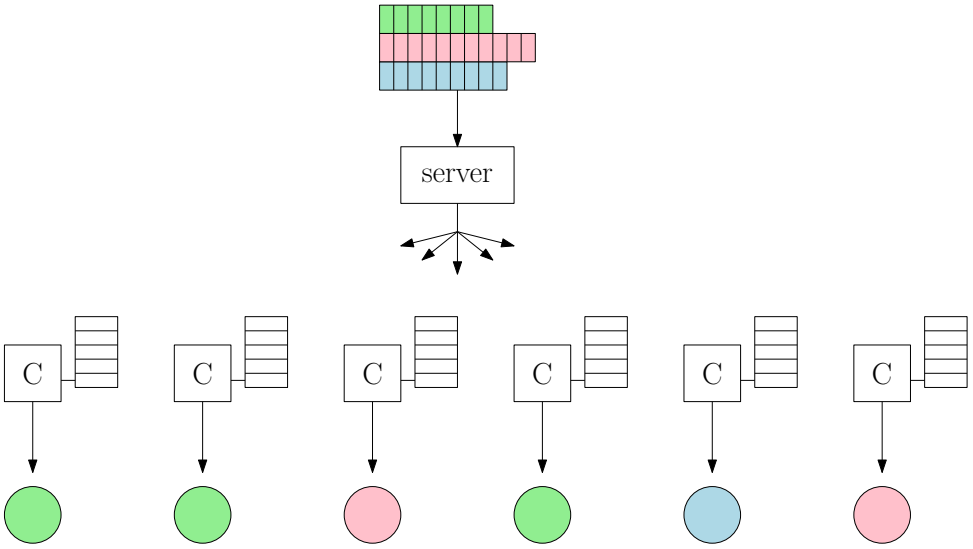
- # of caches
- one user per cache
- # of levels
- For each level:
  - # of files
  - # of users

Single-user setup (formal)



- # of caches
- one user per cache
- # of levels
- For each level:
  - # of files
  - # of users
  - popularity $\propto$ users per file

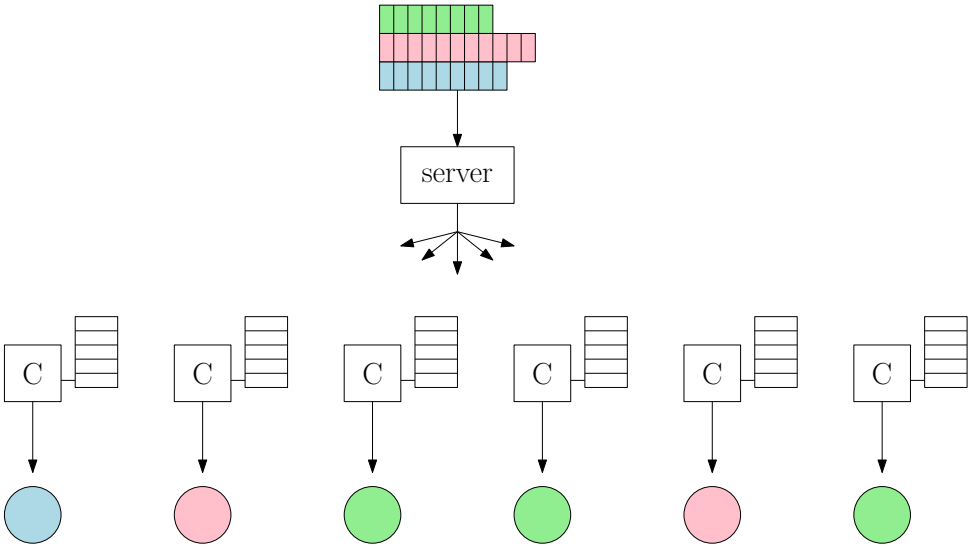
Single-user setup (formal)



unknown arrangement

- # of caches
- one user per cache
- # of levels
- For each level:
  - # of files
  - # of users
  - popularity $\propto$ users per file
- Different possible arrangements

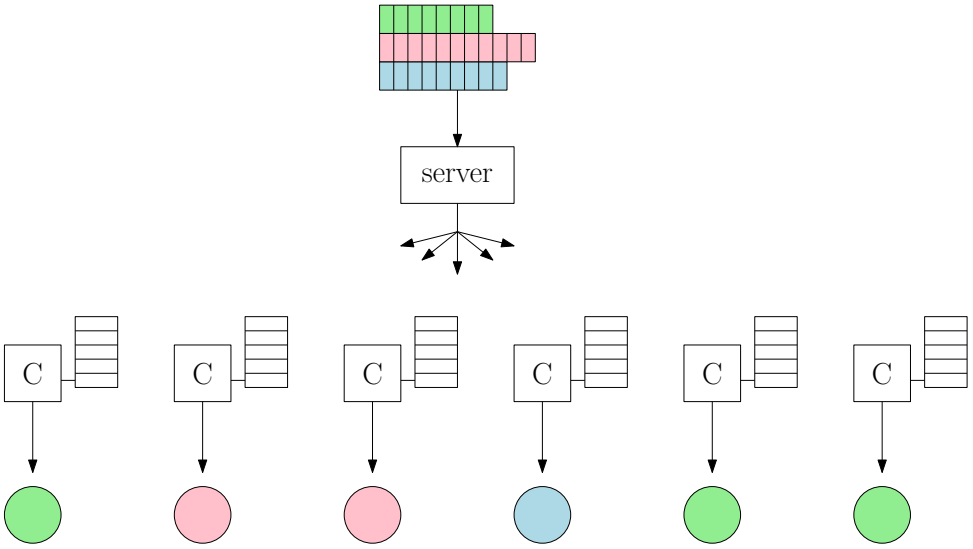
Single-user setup (formal)



unknown arrangement

- # of caches
- one user per cache
- # of levels
- For each level:
  - # of files
  - # of users
  - popularity $\propto$ users per file
- Different possible arrangements

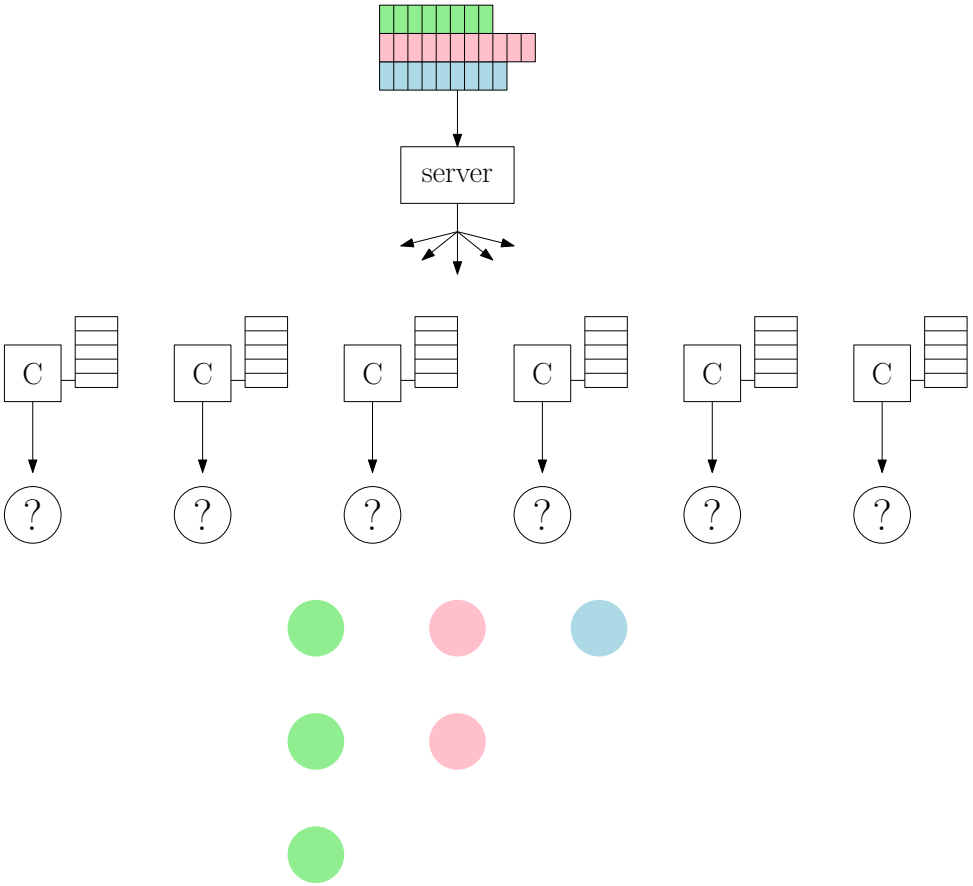
Single-user setup (formal)



unknown arrangement

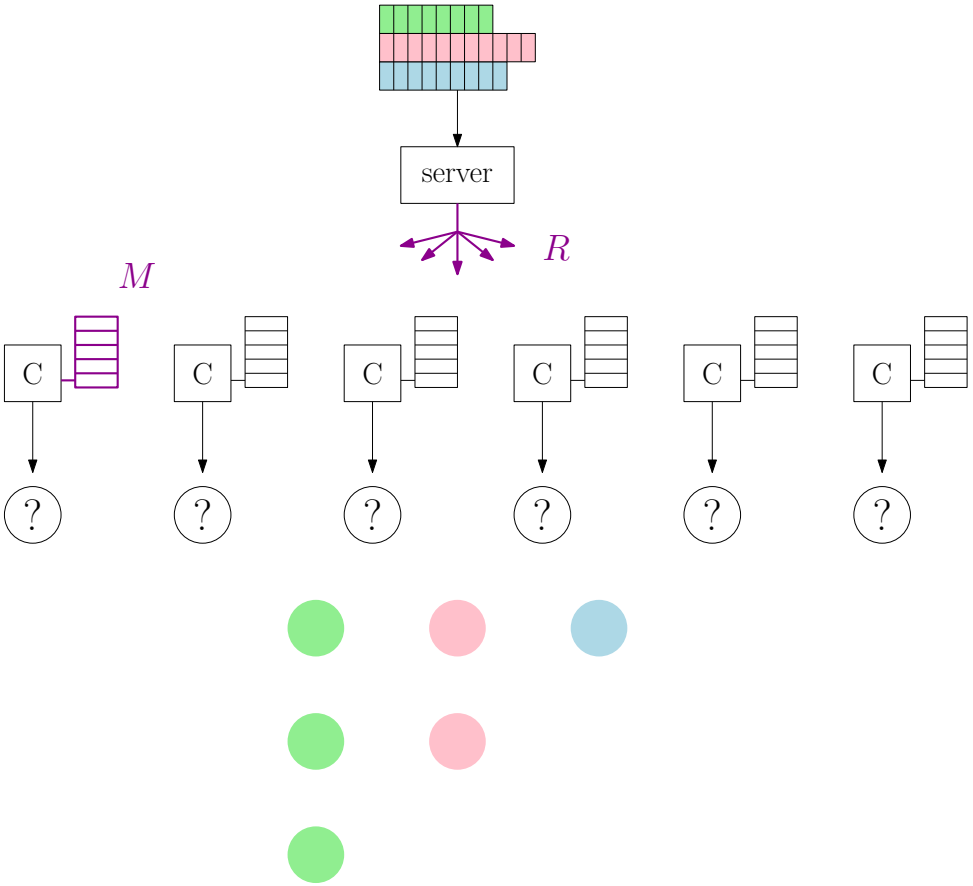
- # of caches
- one user per cache
- # of levels
- For each level:
  - # of files
  - # of users
  - popularity $\propto$ users per file
- Different possible arrangements

Single-user setup (formal)



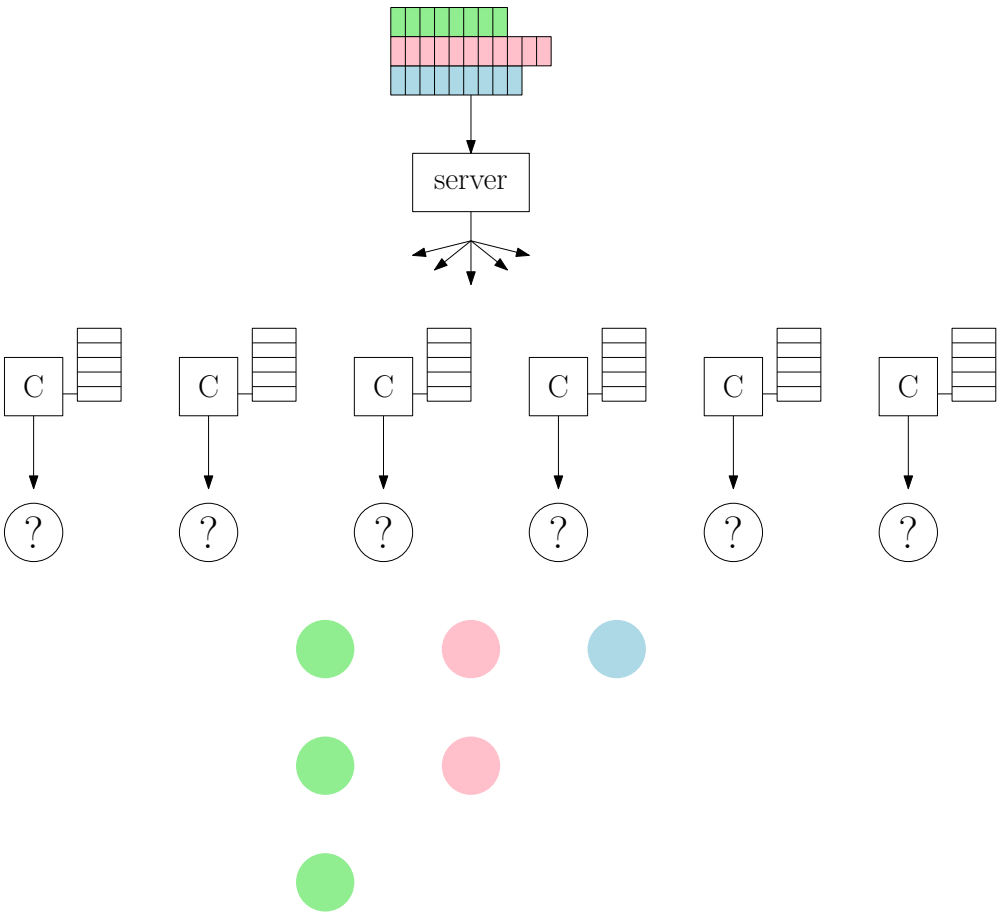
- # of caches
- one user per cache
- # of levels
- For each level:
  - # of files
  - # of users
  - popularity $\propto$ users per file
- Different possible arrangements

Single-user setup (formal)



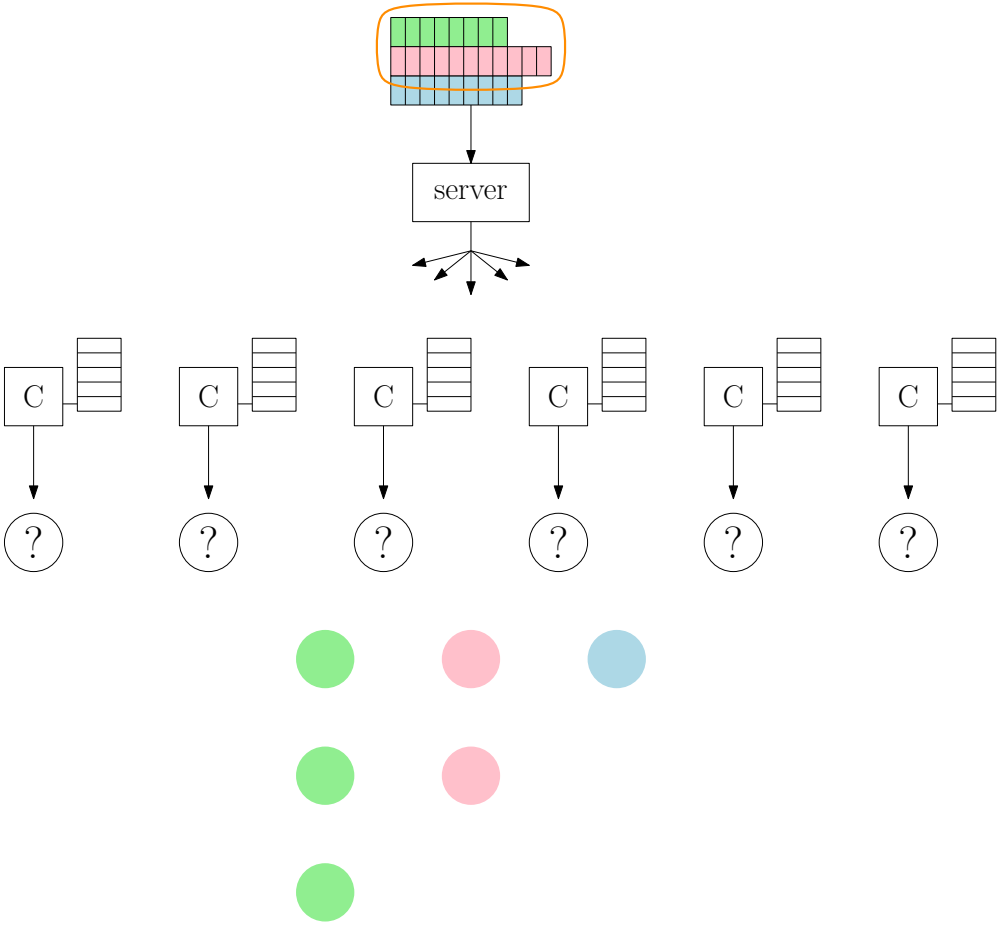
- # of caches
- one user per cache
- # of levels
- For each level:
  - # of files
  - # of users
  - popularity $\propto$ users per file
- Different possible arrangements
- Resources:
  - Cache memory M
  - Broadcast rate R

Strategy: threshold-and-cluster



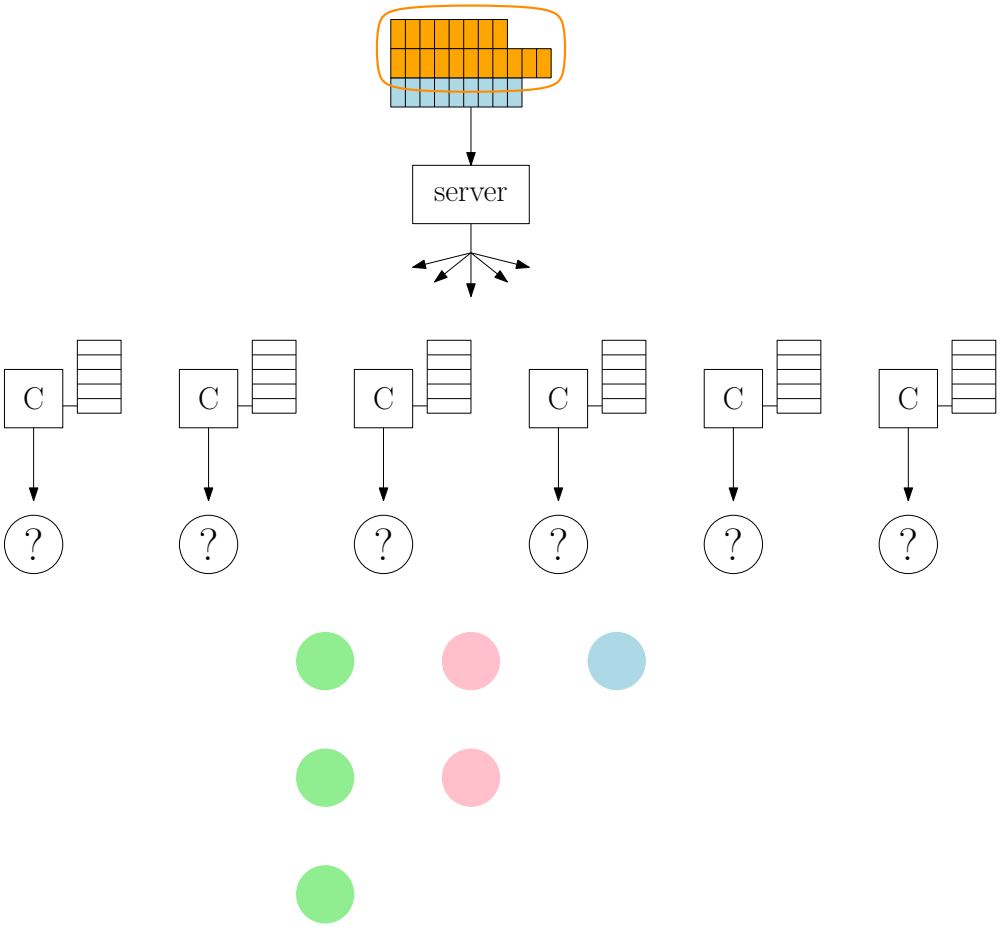
- Idea: merge some levels; ignore the rest

Strategy: threshold-and-cluster



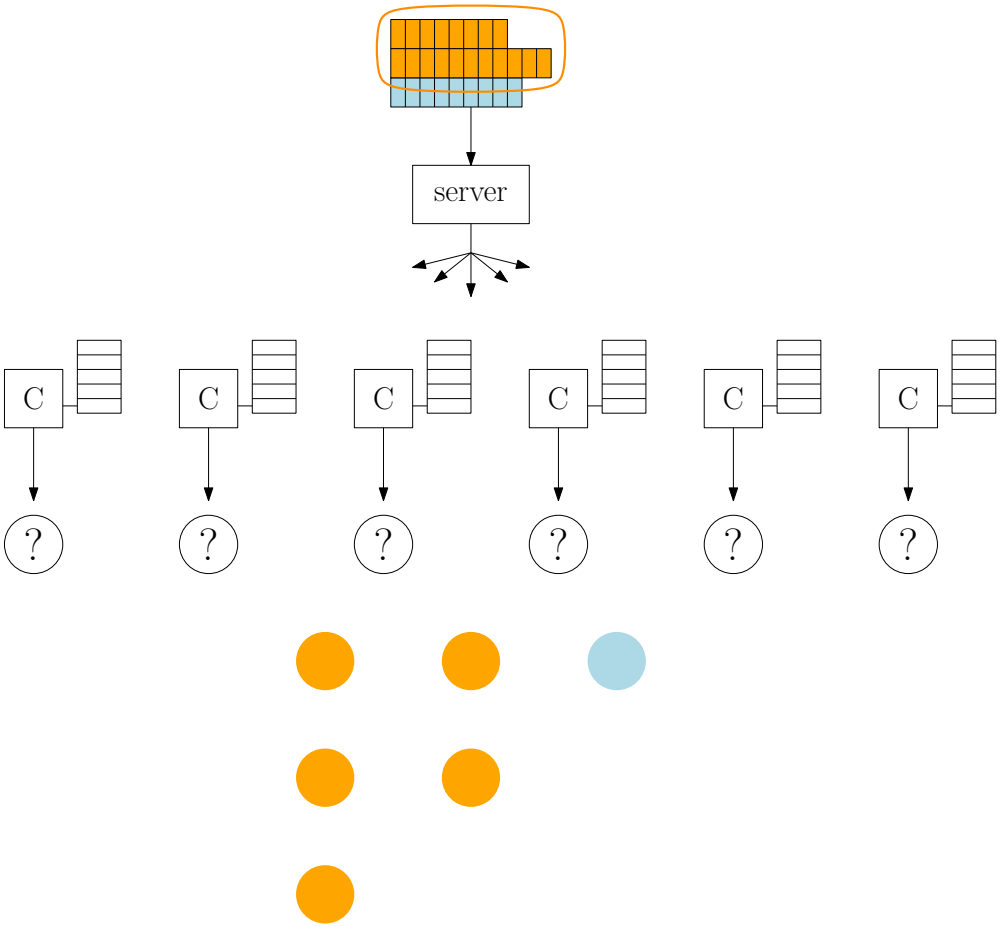
- Idea: merge some levels; ignore the rest
- Choose levels to merge

Strategy: threshold-and-cluster



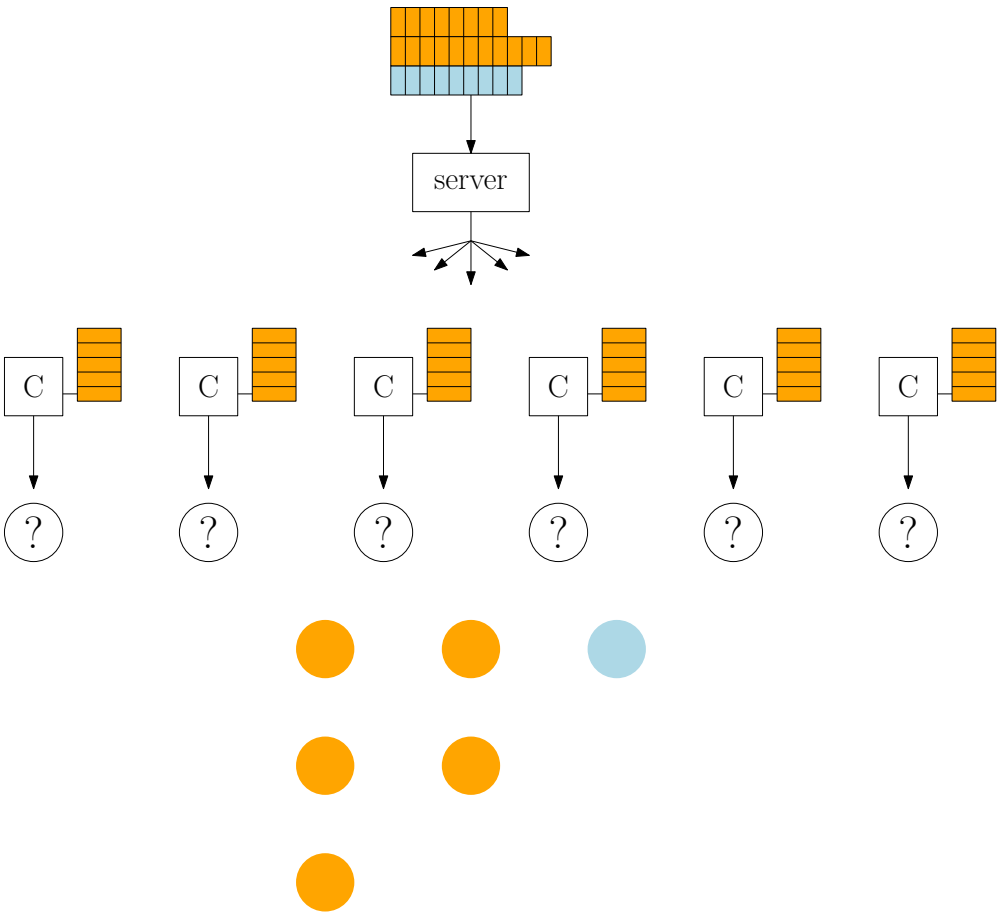
- Idea: merge some levels; ignore the rest
- Choose levels to merge
  - Files

Strategy: threshold-and-cluster



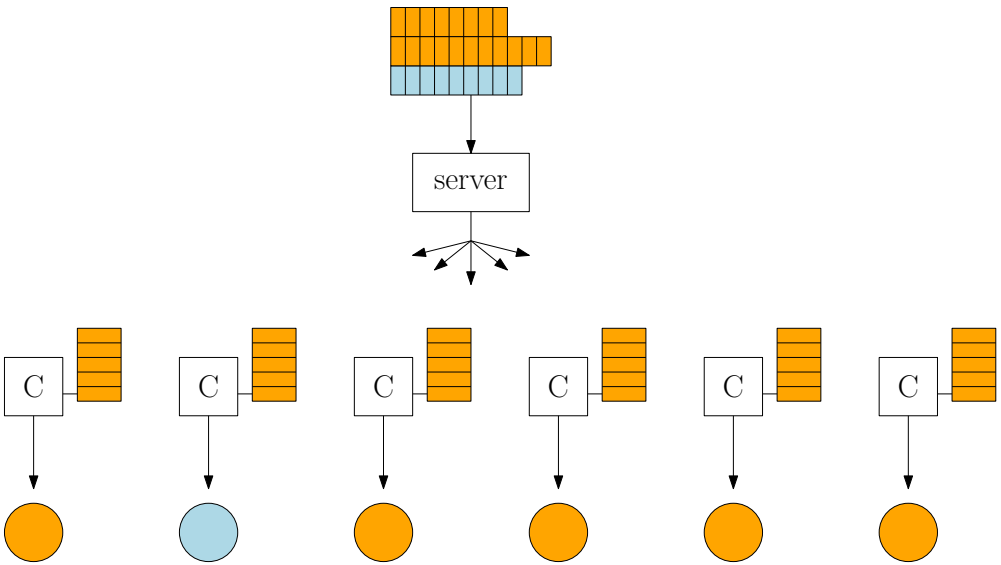
- Idea: merge some levels; ignore the rest
- Choose levels to merge
  - Files
  - User requests

Strategy: threshold-and-cluster



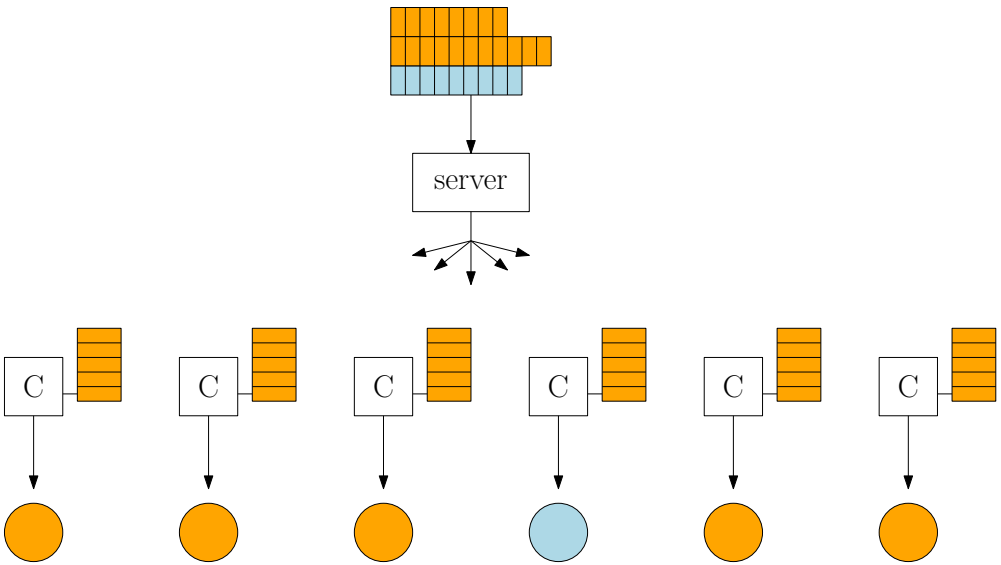
- Idea: merge some levels; ignore the rest
- Choose levels to merge
  - Files
  - User requests
- Give all memory to merged levels
- Ignore remaining levels

Strategy: threshold-and-cluster



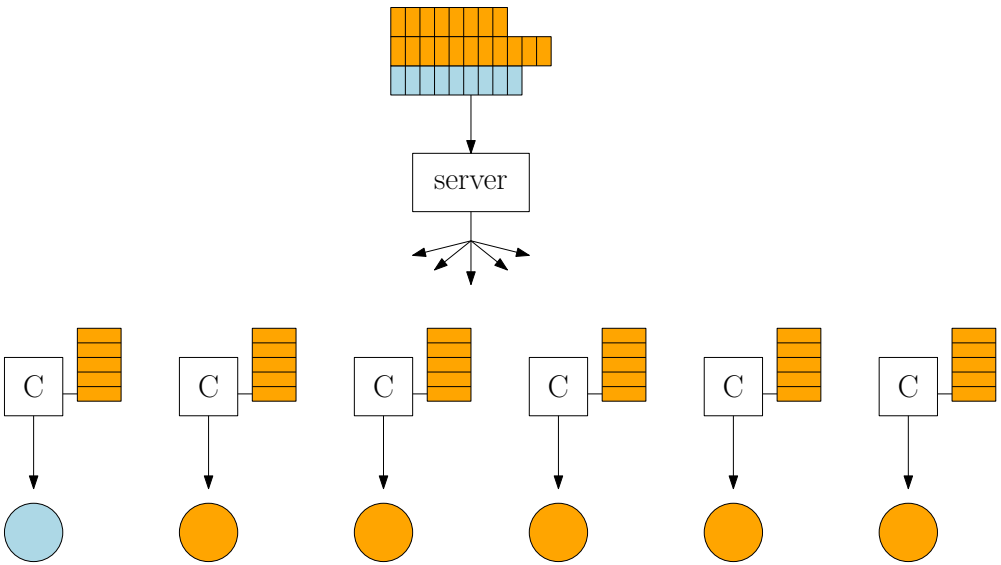
- Idea: merge some levels; ignore the rest
- Choose levels to merge
  - Files
  - User requests
- Give all memory to merged levels
- Ignore remaining levels
- Delivery phase

Strategy: threshold-and-cluster



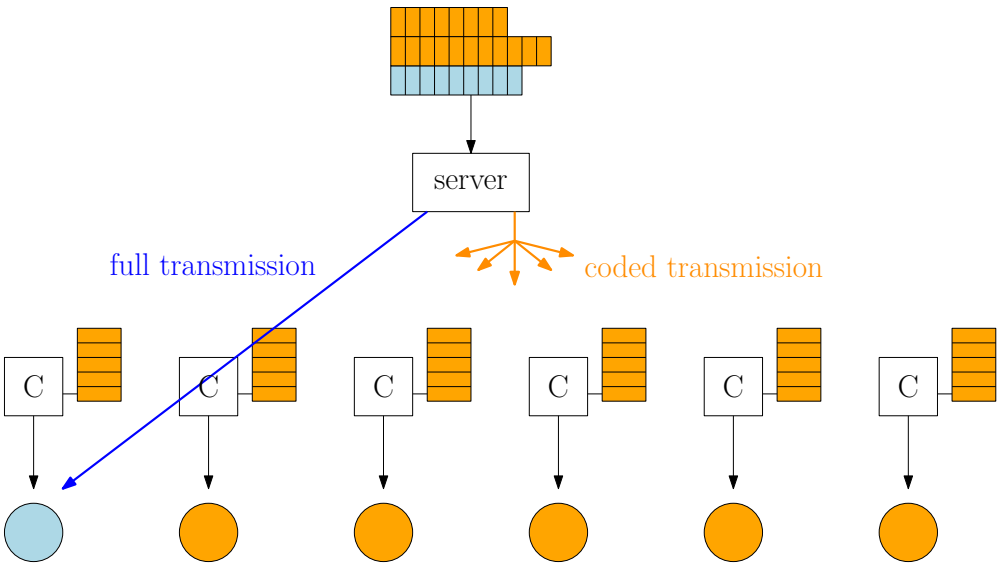
- Idea: merge some levels; ignore the rest
- Choose levels to merge
  - Files
  - User requests
- Give all memory to merged levels
- Ignore remaining levels
- Delivery phase
  - Different possible user profiles

Strategy: threshold-and-cluster



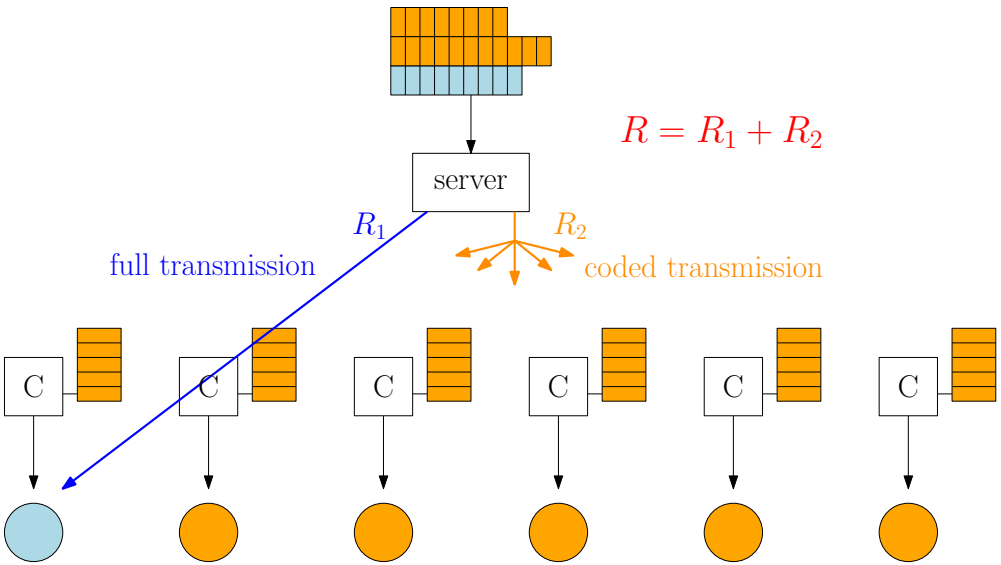
- Idea: merge some levels; ignore the rest
- Choose levels to merge
  - Files
  - User requests
- Give all memory to merged levels
- Ignore remaining levels
- Delivery phase
  - Different possible user profiles

Strategy: threshold-and-cluster



- Idea: merge some levels; ignore the rest
- Choose levels to merge
  - Files
  - User requests
- Give all memory to merged levels
- Ignore remaining levels
- Delivery phase
  - Different possible user profiles
  - Send BC transmissions

Strategy: threshold-and-cluster



- Idea: merge some levels; ignore the rest
- Choose levels to merge
  - Files
  - User requests
- Give all memory to merged levels
- Ignore remaining levels
- Delivery phase
  - Different possible user profiles
  - Send BC transmissions
  - Total rate = sum of rates

Strategy: threshold-and-cluster

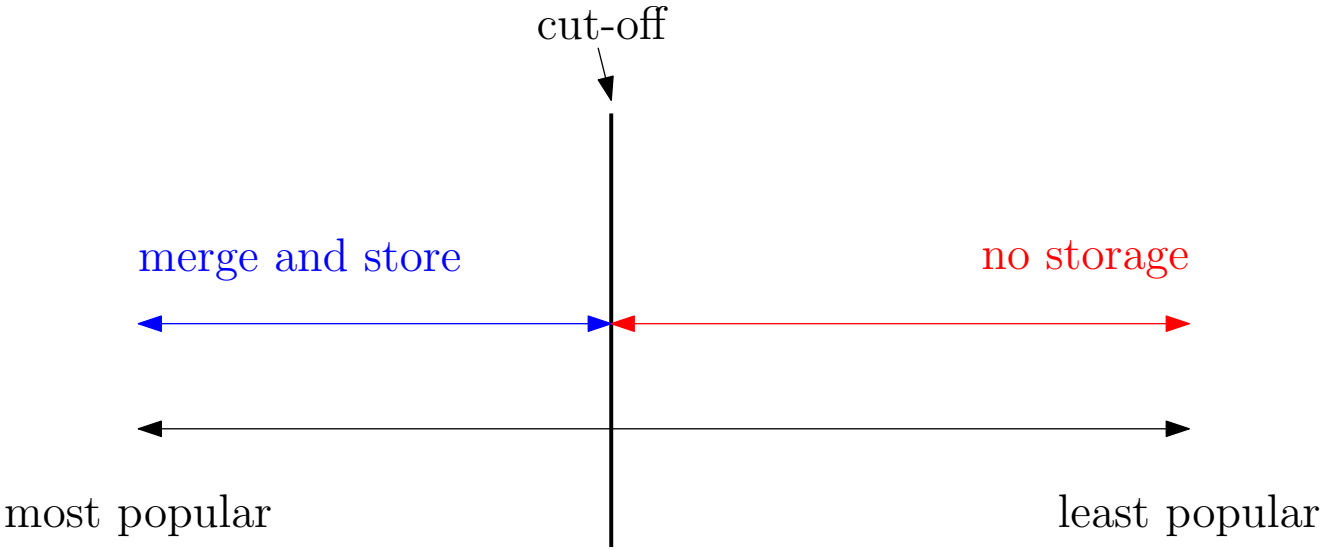
Which levels to merge?

Strategy: threshold-and-cluster

Which levels to merge?

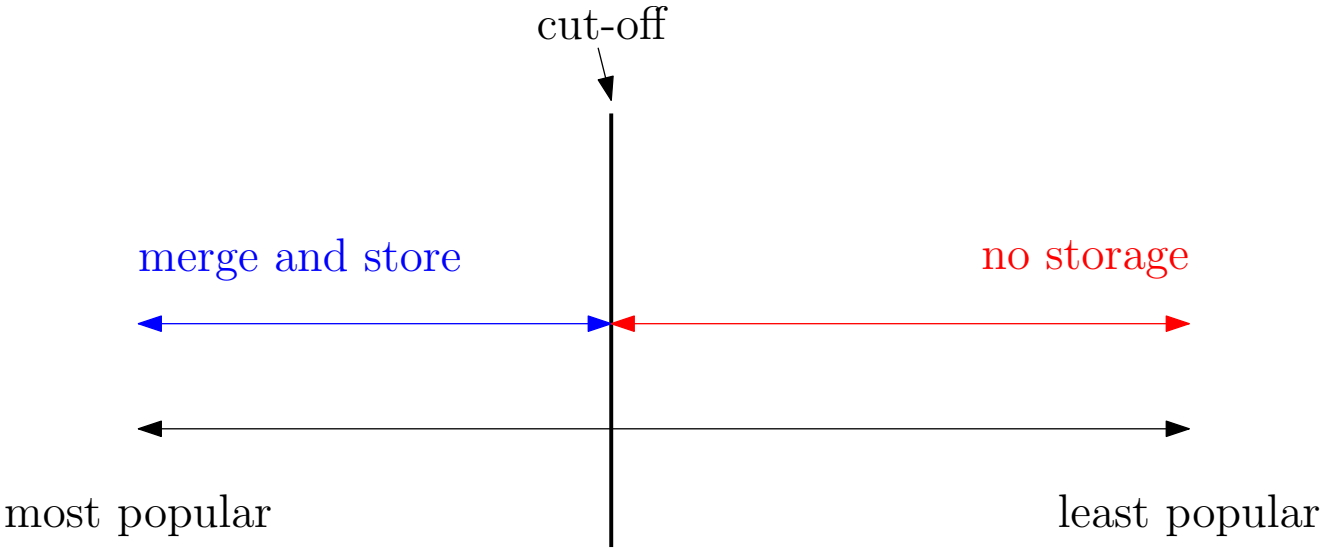
Strategy: threshold-and-cluster

Which levels to merge?



Strategy: threshold-and-cluster

Which levels to merge?



Achieved rate:

$$R \approx \sum_h K_h + \frac{\sum_i N_i}{M}$$

Order-optimality

Theorem (Order-optimality of threshold-and-cluster for the single-user setup)

In the **single-user** *setup:*

$$\frac{\textit{rate achieved by } \textbf{threshold-and-cluster}}{\textit{optimal rate } R^*} \leq c.$$

Order-optimality

Theorem (Order-optimality of threshold-and-cluster for the single-user setup)

In the **single-user** *setup:*

$$\frac{\textit{rate achieved by } \textbf{threshold-and-cluster}}{\textit{optimal rate } R^*} \leq c.$$

- c is independent of problem parameters

Order-optimality

Theorem (Order-optimality of threshold-and-cluster for the single-user setup)

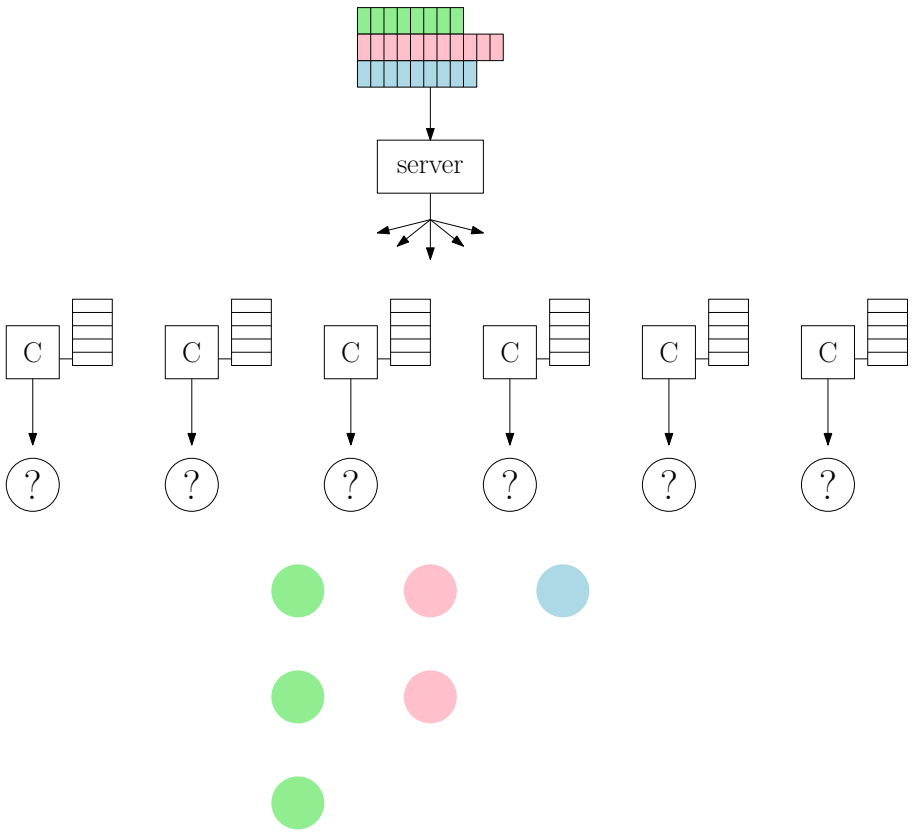
In the **single-user** *setup:*

$$\frac{\textit{rate achieved by } \textbf{threshold-and-cluster}}{\textit{optimal rate } R^*} \leq c.$$

- c is independent of problem parameters
- Proof: cut-set lower bounds on R^* are sufficient

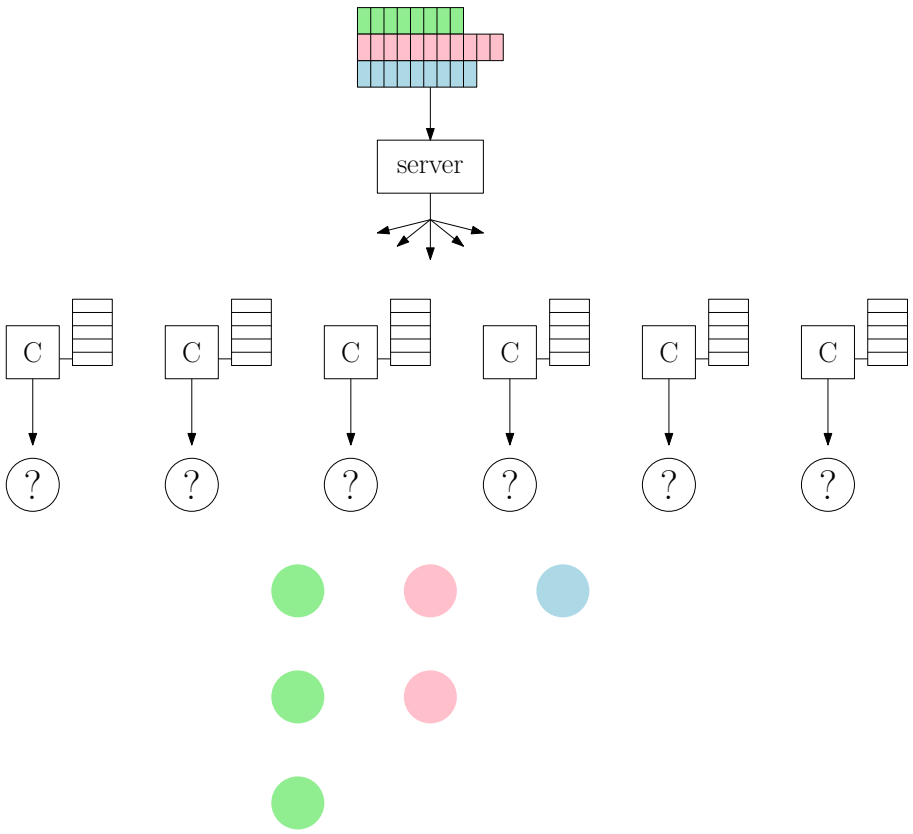
Lower bounds for single-user setup

- Capture:
  - Necessity of level merging
  - Uncertainty of level at caches



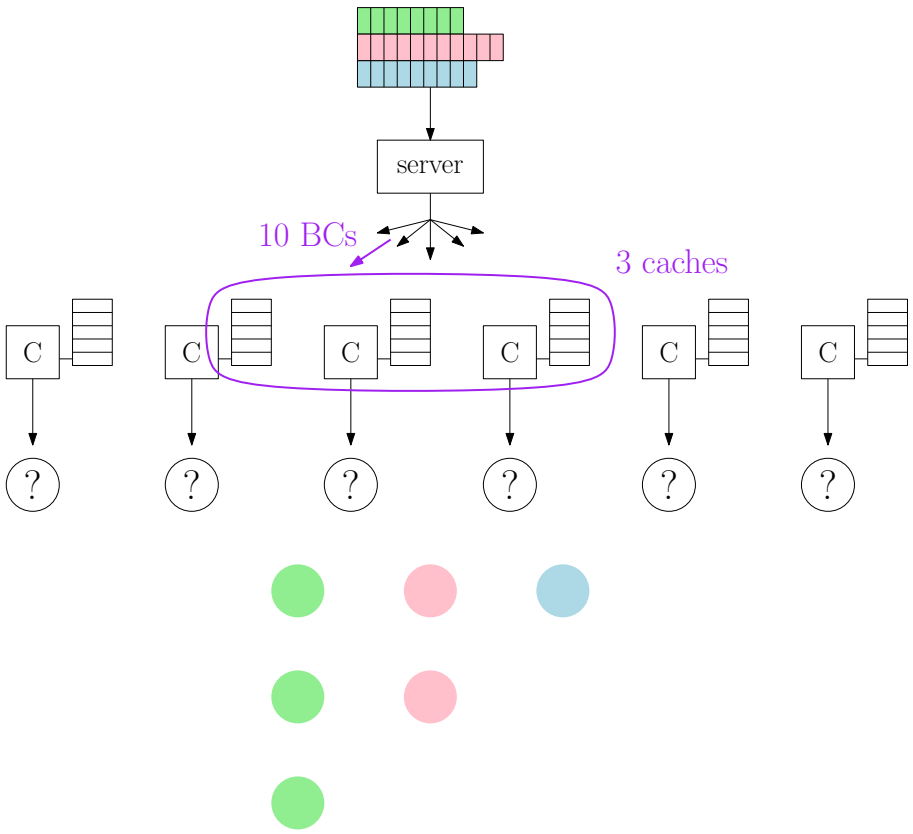
Lower bounds for single-user setup

- Capture:
  - Necessity of level merging
  - Uncertainty of level at caches
- One cut-set bound:
  - Serve multiple levels together
  - Different BC messages assume different request profiles



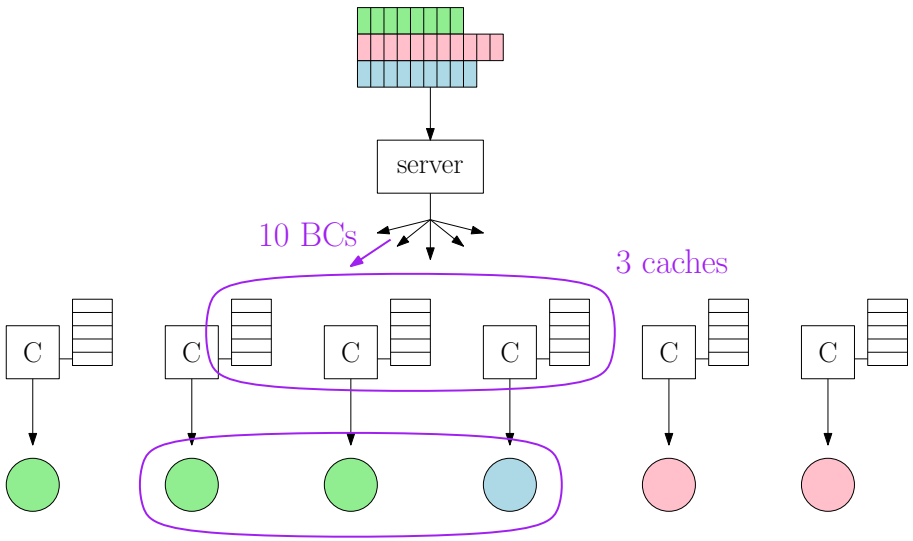
Lower bounds for single-user setup

- Capture:
  - Necessity of level merging
  - Uncertainty of level at caches
- One cut-set bound:
  - Serve multiple levels together
  - Different BC messages assume different request profiles



Lower bounds for single-user setup

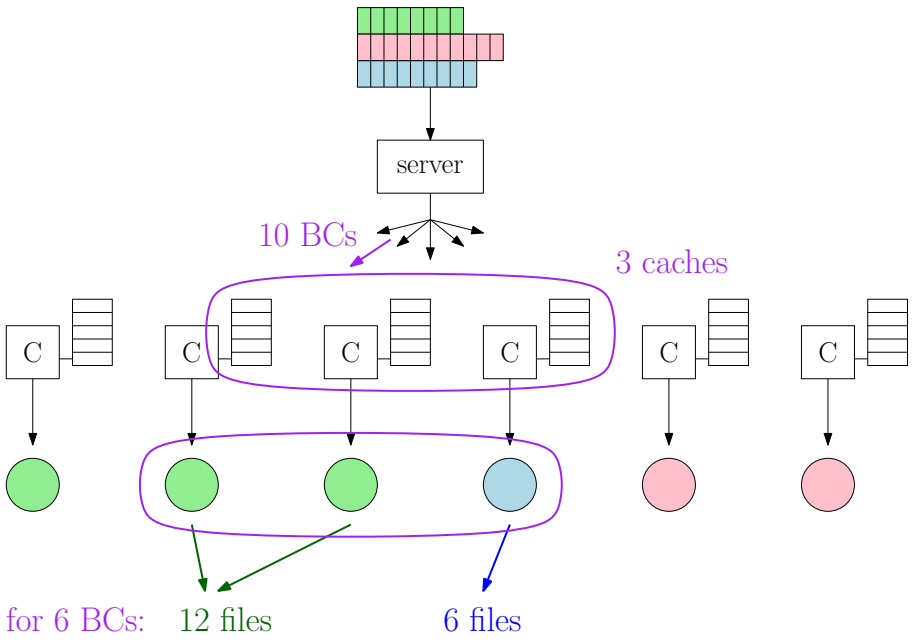
- Capture:
  - Necessity of level merging
  - Uncertainty of level at caches
- One cut-set bound:
  - Serve multiple levels together
  - Different BC messages assume different request profiles



for 6 BCs:

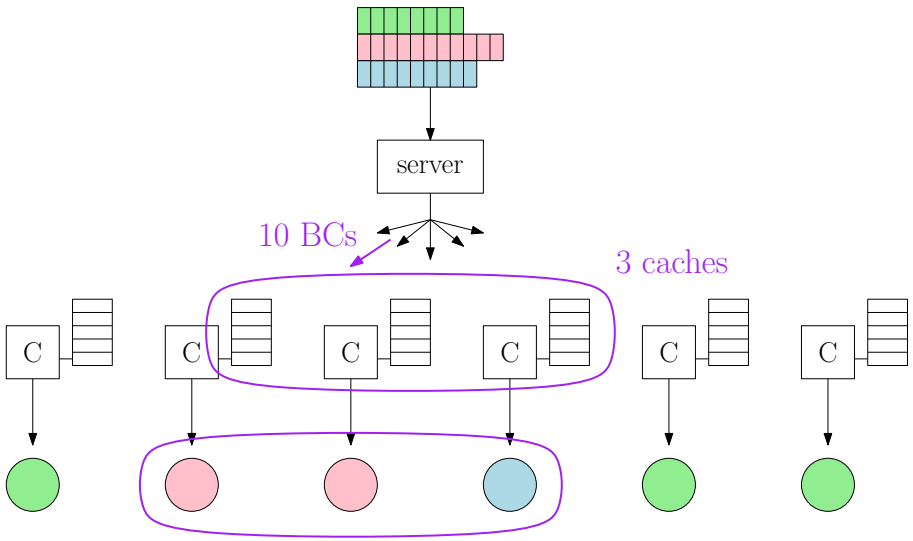
Lower bounds for single-user setup

- Capture:
  - Necessity of level merging
  - Uncertainty of level at caches
- One cut-set bound:
  - Serve multiple levels together
  - Different BC messages assume different request profiles



Lower bounds for single-user setup

- Capture:
  - Necessity of level merging
  - Uncertainty of level at caches
- One cut-set bound:
  - Serve multiple levels together
  - Different BC messages assume different request profiles

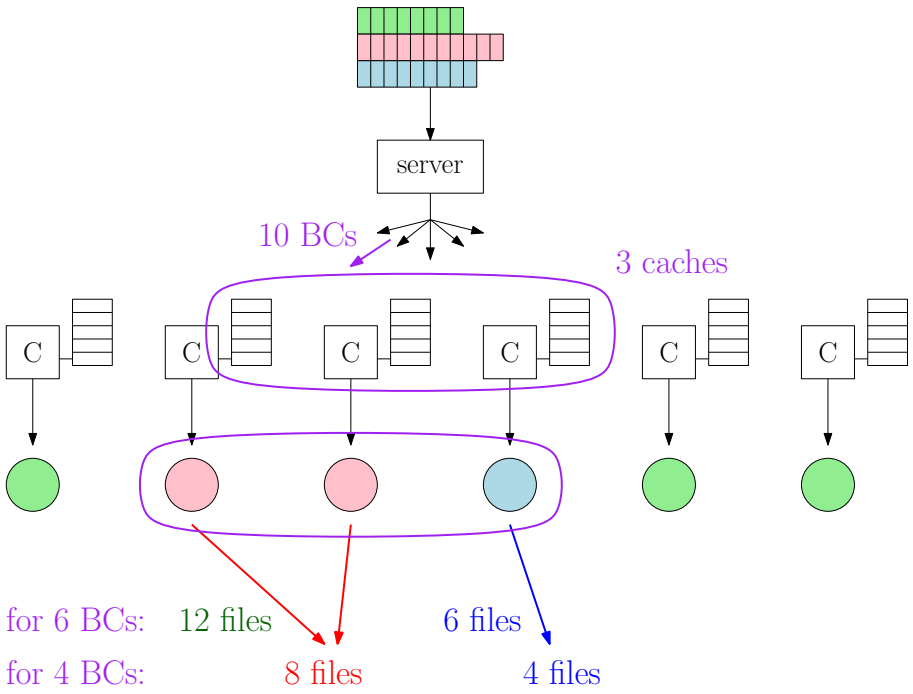


for 6 BCs: 12 files 6 files

for 4 BCs:

Lower bounds for single-user setup

- Capture:
  - Necessity of level merging
  - Uncertainty of level at caches
- One cut-set bound:
  - Serve multiple levels together
  - Different BC messages assume different request profiles



One strategy to rule them all?

Question: Could one (or both) strategy be order-optimal in both cases?

One strategy to rule them all?

Question: Could one (or both) strategy be order-optimal in both cases?

No!

One strategy to rule them all?

Question: Could one (or both) strategy be order-optimal in both cases?

No!

- Memory-sharing in single-user setup?

One strategy to rule them all?

Question: Could one (or both) strategy be order-optimal in both cases?

No!

- Memory-sharing in single-user setup?
  - Wastes coding opportunities

One strategy to rule them all?

Question: Could one (or both) strategy be order-optimal in both cases?

No!

- Memory-sharing in single-user setup?
  - Wastes coding opportunities
  - **Factor-**L increase in R $\implies$ not order-optimal!
    (L is the number of levels)

One strategy to rule them all?

Question: Could one (or both) strategy be order-optimal in both cases?

No!

- Memory-sharing in single-user setup?
  - Wastes coding opportunities
  - **Factor-**L increase in R $\Longrightarrow$ not order-optimal!
    (L is the number of levels)
- Threshold-and-cluster in multi-user setup?

One strategy to rule them all?

Question: Could one (or both) strategy be order-optimal in both cases?

No!

- Memory-sharing in single-user setup?
  - Wastes coding opportunities
  - **Factor-**L increase in R $\implies$ not order-optimal!
    (L is the number of levels)
- Threshold-and-cluster in multi-user setup?
  - Inefficiently distributes memory

One strategy to rule them all?

Question: Could one (or both) strategy be order-optimal in both cases?

No!

- Memory-sharing in single-user setup?
  - Wastes coding opportunities
  - **Factor-**L increase in R $\implies$ not order-optimal!
    (L is the number of levels)
- Threshold-and-cluster in multi-user setup?
  - Inefficiently distributes memory
  - No coding opportunities gained

One strategy to rule them all?

Question: Could one (or both) strategy be order-optimal in both cases?

No!

- Memory-sharing in single-user setup?
  - Wastes coding opportunities
  - **Factor-**L increase in R $\implies$ not order-optimal!
    (L is the number of levels)
- Threshold-and-cluster in multi-user setup?
  - Inefficiently distributes memory
  - No coding opportunities gained
  - **Arbitrarily large** increase in R $\implies$ not order-optimal!

Summary

- Coded caching

Summary

- Coded caching
- Multi-level popularities

Summary

- Coded caching
- Multi-level popularities
- Large vs small number of users per cache $\implies$ Determinism vs uncertainty in per-cache profiles

Summary

- Coded caching
- Multi-level popularities
- Large vs small number of users per cache $\implies$ Determinism vs uncertainty in per-cache profiles

	Multi-user	Single-user
Memory-sharing	✓	✗
Threshold-and-cluster	✗	✓

Summary

- Coded caching
- Multi-level popularities
- Large vs small number of users per cache $\implies$ Determinism vs uncertainty in per-cache profiles

	Multi-user	Single-user
Memory-sharing	✓	✗
Threshold-and-cluster	✗	✓

Thank you!