Scalable and private media consumption with Popcorn

Trinabh Gupta

The University of Texas at Austin

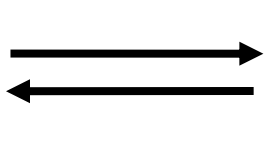
90 minutes/day

give me The Godfather

The Godfather

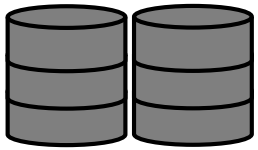
NETFLIX

User media consumption has increased …



NETFLIX

database of request trace, movie ratings, etc.



… leading to large centralized datasets …

… subject to risks such as server hacks, accidental disclosures, etc.

anonymized dataset of movie ratings

researcher

de-anonymized dataset of movie ratings

NETFLIX SPILLED YOUR *BROKEBACK MOUNTAIN* SECRET, LAWSUIT CLAIMS

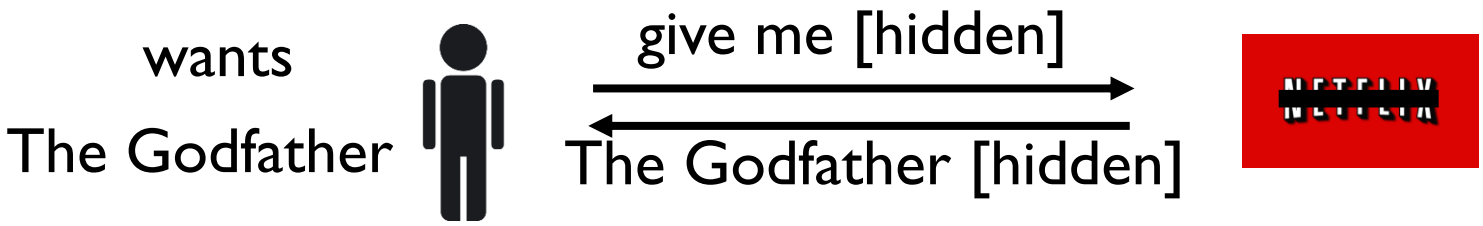
How can we build a Netflix-like system that

a) provably hides media diet,

b) has low dollar cost, and

c) is compatible with commercial media streaming?

Private Information Retrieval (PIR) provably hides requests but …



- Each request must touch the entire library.
- There is a tension between overhead and content protection.
- PIR assumes fixed-size objects, but media sizes vary.

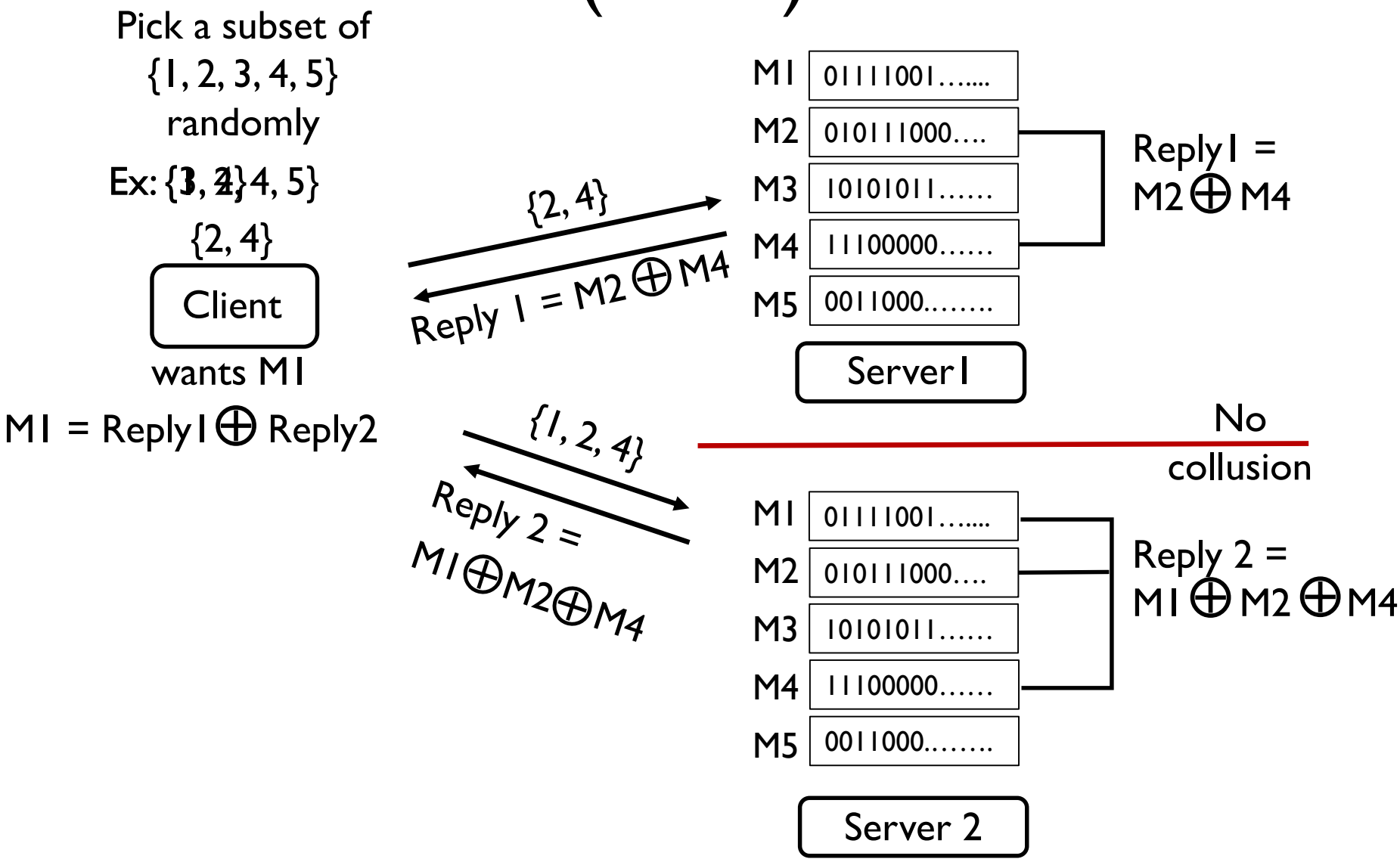
Popcorn tailors PIR for media to meet our three requirements.

Its per-request dollar cost is 3.87x times that of a non-private baseline.

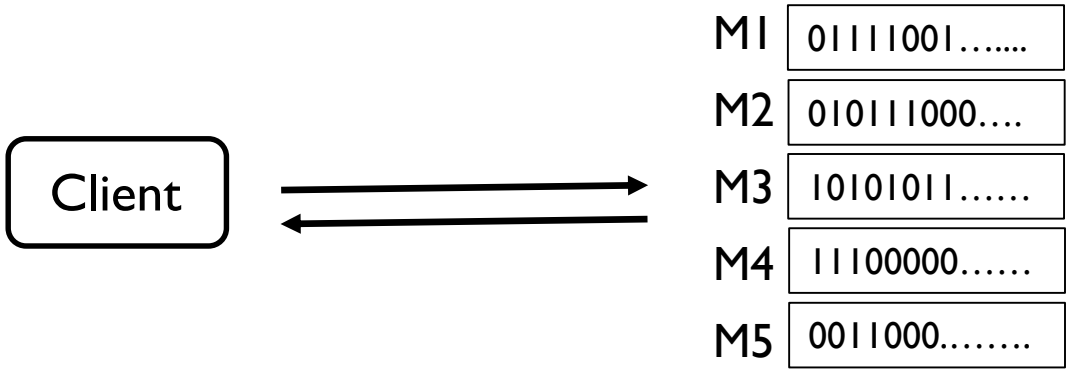
Rest of this talk

- Background on PIR.
- Challenges of using PIR (in detail).
- Design (tailoring of PIR) and evaluation of Popcorn.

Background on information-theoretic PIR (ITPIR)



Computational PIR (CPIR) from 10,000 feet



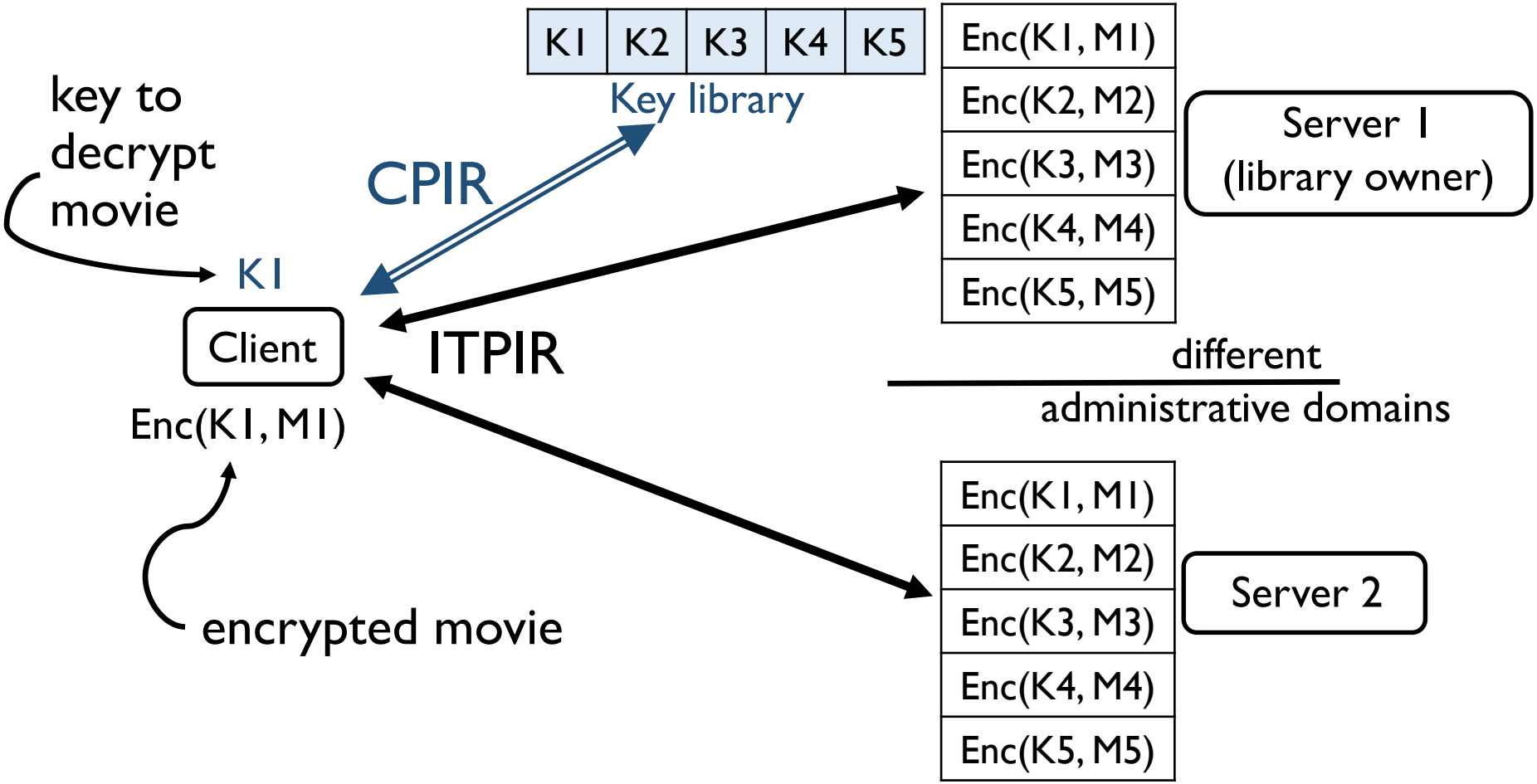
- one server
- instead of XORs, expensive server-side cryptographic operations

Challenges of using PIR

ITPIR	CPIR
content can disseminate in an uncontrolled manner	content disseminates in a controlled manner
cheap operations (XORs) but process entire library per request	expensive operations and process entire library per request
assumes fixed-size objects	assumes fixed-size objects

Given these, how can we build a system that controls content and is low cost?

Popcorn composes ITPIR and CPIR to get desirable properties from both

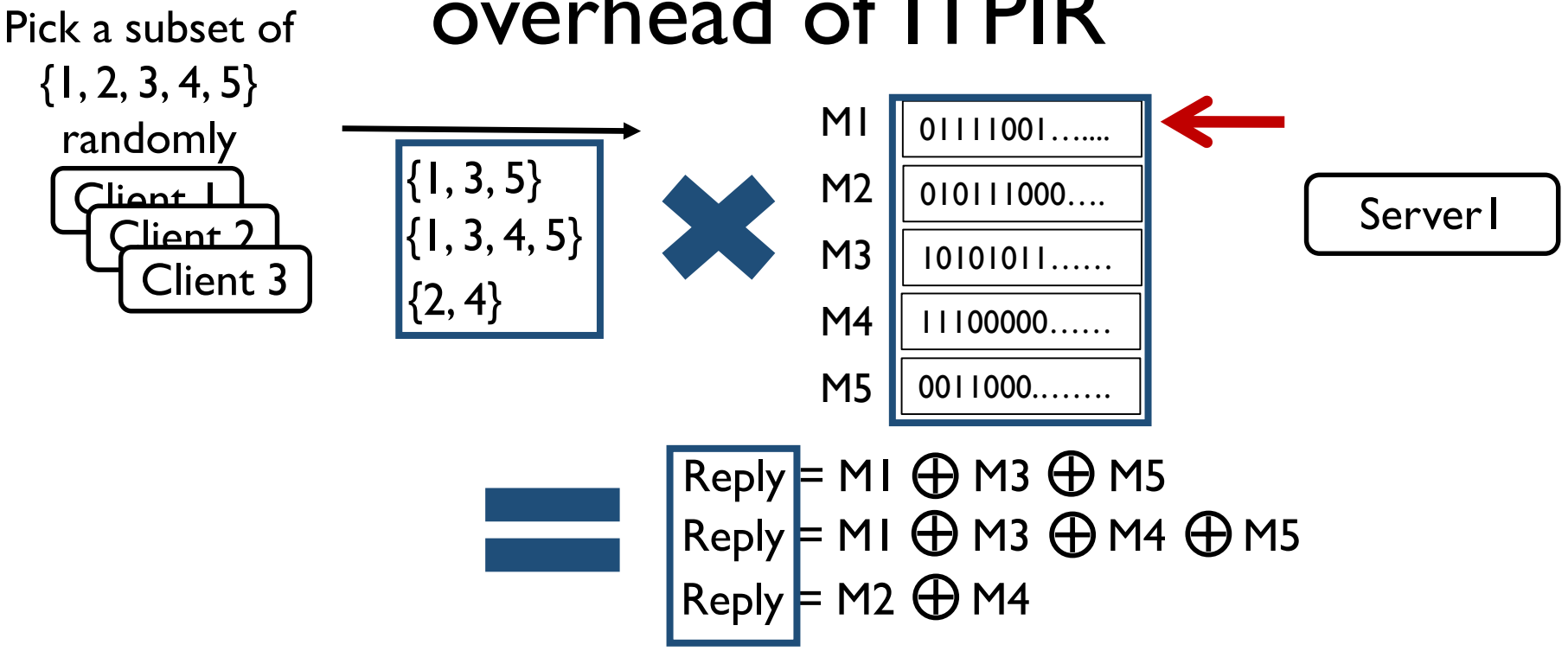


Challenges of using PIR

ITPIR	CPIR
content can disseminate in an uncontrolled manner	content disseminates in a controlled manner
cheap operations (XORs) but process entire library per request	expensive operations and process entire library per request
assumes fixed-size objects	assumes fixed-size objects

Popcorn

Popcorn batches requests to amortize the overhead of ITPIR

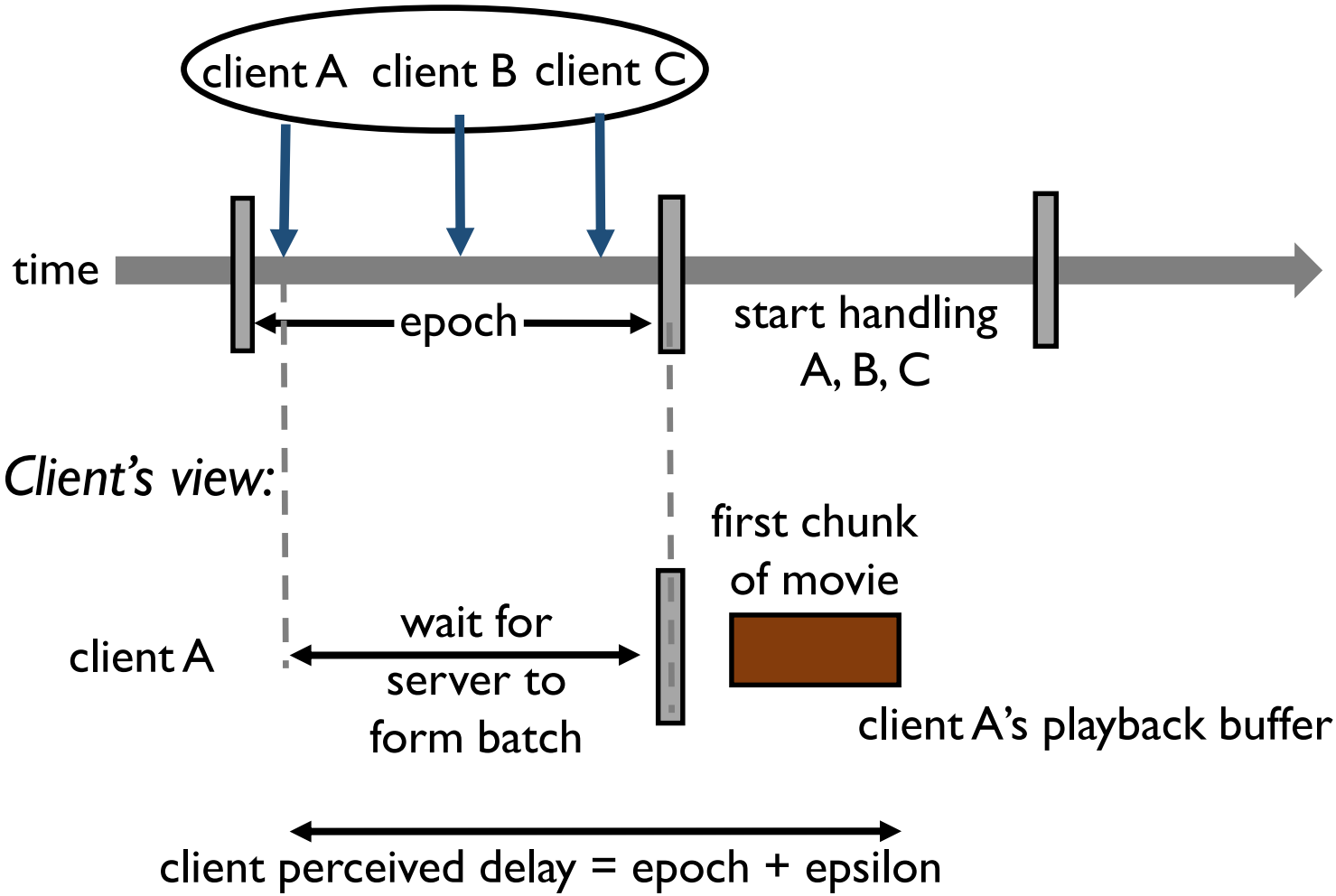


Observation: Very similar disk I/O for each request!

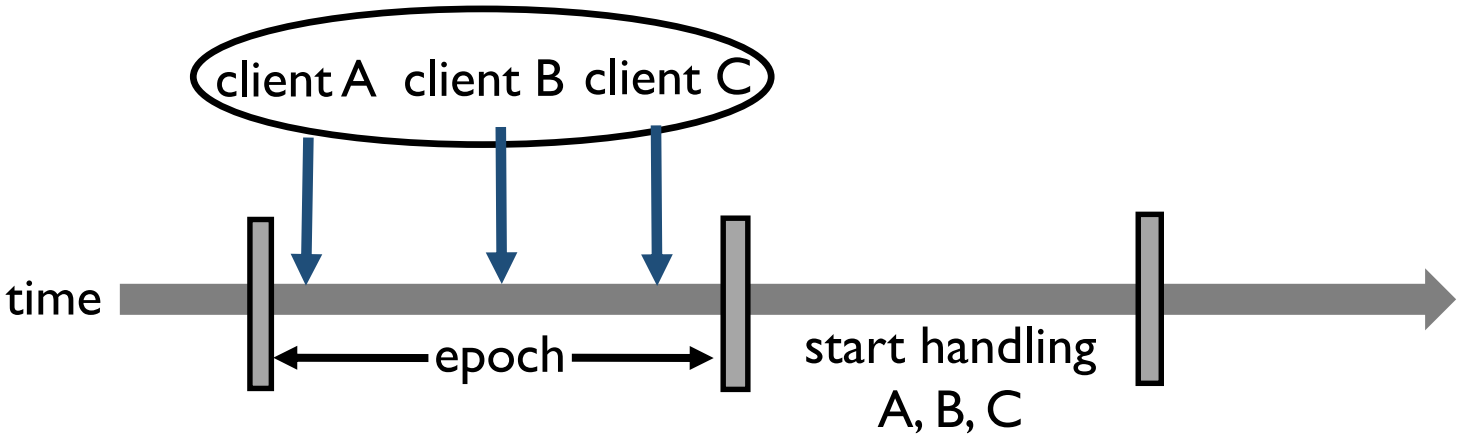
Benefits of batching:

- Disk I/O transfers are amortized.
- CPU cycles are reduced as matrix multiplication algorithms exploit cache locality.

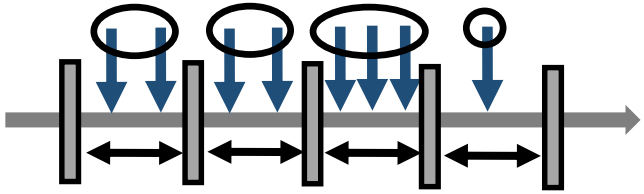
Straw man: Group requests that arrive during an epoch



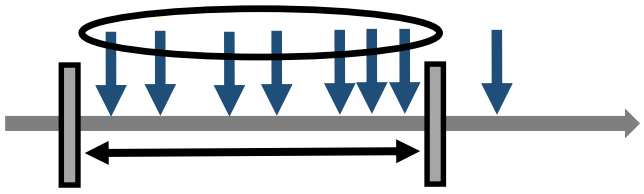
Straw man: Group requests that arrive during an epoch



Server's choices:



Small batch, small delay



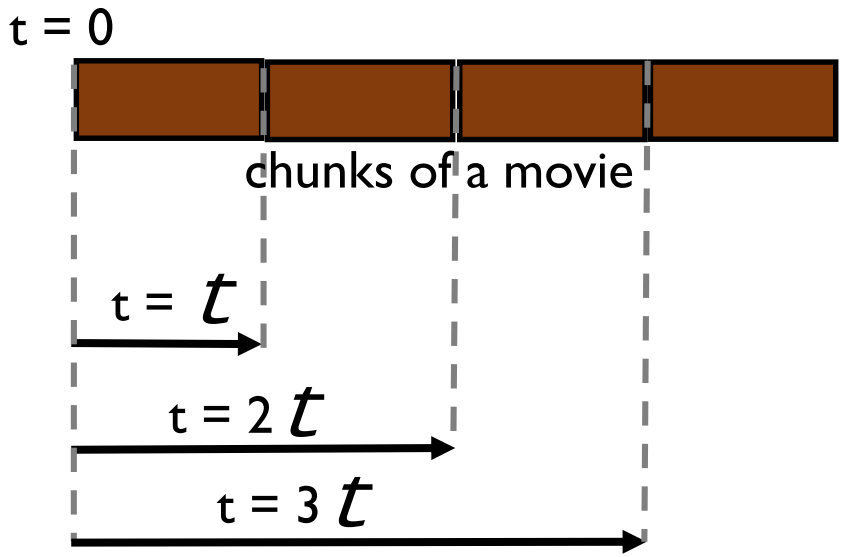
Large batch, large delay

Issue: Hard to get both small delay and large batch

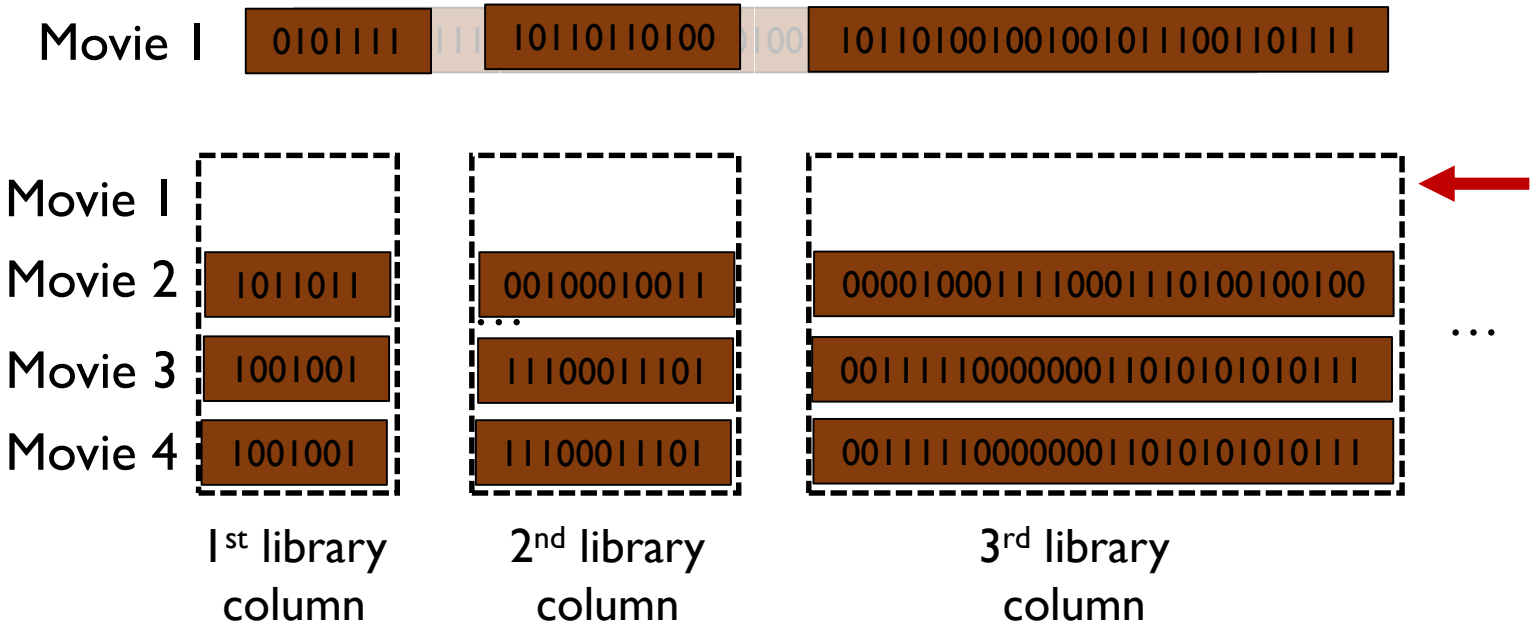
Popcorn exploits streaming to form large batches with small startup delay

t = times at which a client needs movie chunks

t = time it takes to consume a single chunk



Observation: Client needs only the first chunk immediately.



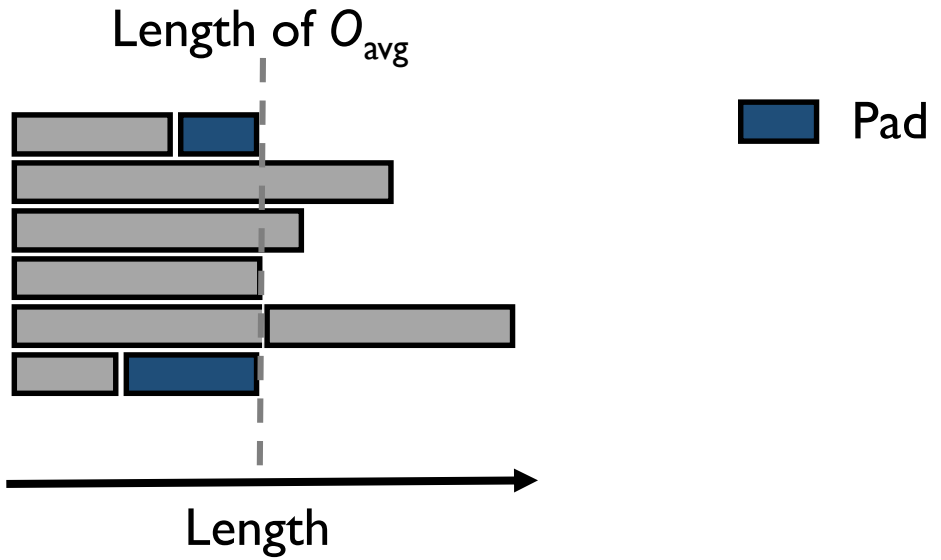
Narrow first column => small startup delay

Wider columns => longer processing times …

… ***but bigger batches***

ITPIR	CPIR	Popcorn
content can disseminate in an uncontrolled manner	content disseminates in a controlled manner	content disseminates in a controlled manner
cheap operations but process entire library *per request*	expensive operations, process entire library *per request*	cheap operations, process entire library *per batch*
assumes fixed-size objects	assumes fixed-size objects	?

Popcorn exploits compression to address fixed-size requirement



- Small variations in bitrate have limited impact on user satisfaction [SIGCOMM 11, LANC 11, CCNC 12].

- 85% of movies close to the average size.

Outline

Popcorn

✓Background on PIR.

✓Design (tailoring of PIR) of Popcorn.

• Evaluation of Popcorn.

Experiment method

Baselines:

- Non-private system (Apache server)
- State-of-the-art CPIR [XPIR PETS16]
- State-of-the-art ITPIR [Percy++]
- ITPIR++: ITPIR extended with the straw man batching scheme

Netflix-like library: 8000 movies, 90 minutes, 4Mbps

Workload: 10K clients arrive within 90 minutes according to a Poisson process

Estimate per-request dollar cost using Amazon's pricing model

- CPU: $0.0076/hour
- Disk I/O bandwidth: $0.042/Gbps-hour
- Network: $0.006/GB

System	# of CPUs	Disk I/O (Gbps)	Network (relative to non-private)	$ relative to non-private
Non-private	0	0	1x	1x
CPIR	11.6	64	5x	265x
ITPIR	3.1	64	2x	256x
ITPIR++ (delay 15s)	0.65	3	2x	14x
Popcorn (delay 15s)	0.74	0.23	2x	3.87x

Popcorn is private and affordable but …

- Assumes that the ITPIR servers do not collude.
- Incurs costs that are linear in the size of the library.
- Does not support recommendations, aggregate view statistics.

  Solution: Use prior work [Canny S&P '02, Toubiana et al. NDSS '10]

Related work

- Improving performance of PIR.
  - Distributing work [FC13, TDSC12], cheaper crypto [PETS16, ESORICS14, ISC10, TKDE13, WEWoRC07], bucketing [DBSec10, PETS10], batching [FC15, JoC04], secure co-processors [PET03, FAST13, NDSS08, IBM Systems Journal01]
- Protecting library content in ITPIR [RANDOM98, S&P07, WPES13]
- Handling variable-sized objects [CCSW14, NDSS13]
- Prior PIR implementations [Percy++, PETS16, CCSW14]
- Video-on-demand [MMCN95]

Take-away points from Popcorn

- It is possible to build a private, functional, and low-cost media delivery system …
- … by tailoring PIR to media delivery.
- The per-request cost in Popcorn is 3.87x that of a non-private baseline.