On Content Delivery to Heterogeneous Devices

Sharayu Moharir

School of Technology and Computer Science, TIFR

Joint work with Rahul Vaze

Motivation

Content Delivery Networks

1. Large amount of content
2. Device heterogeneity

Motivation

Device Heterogeneity

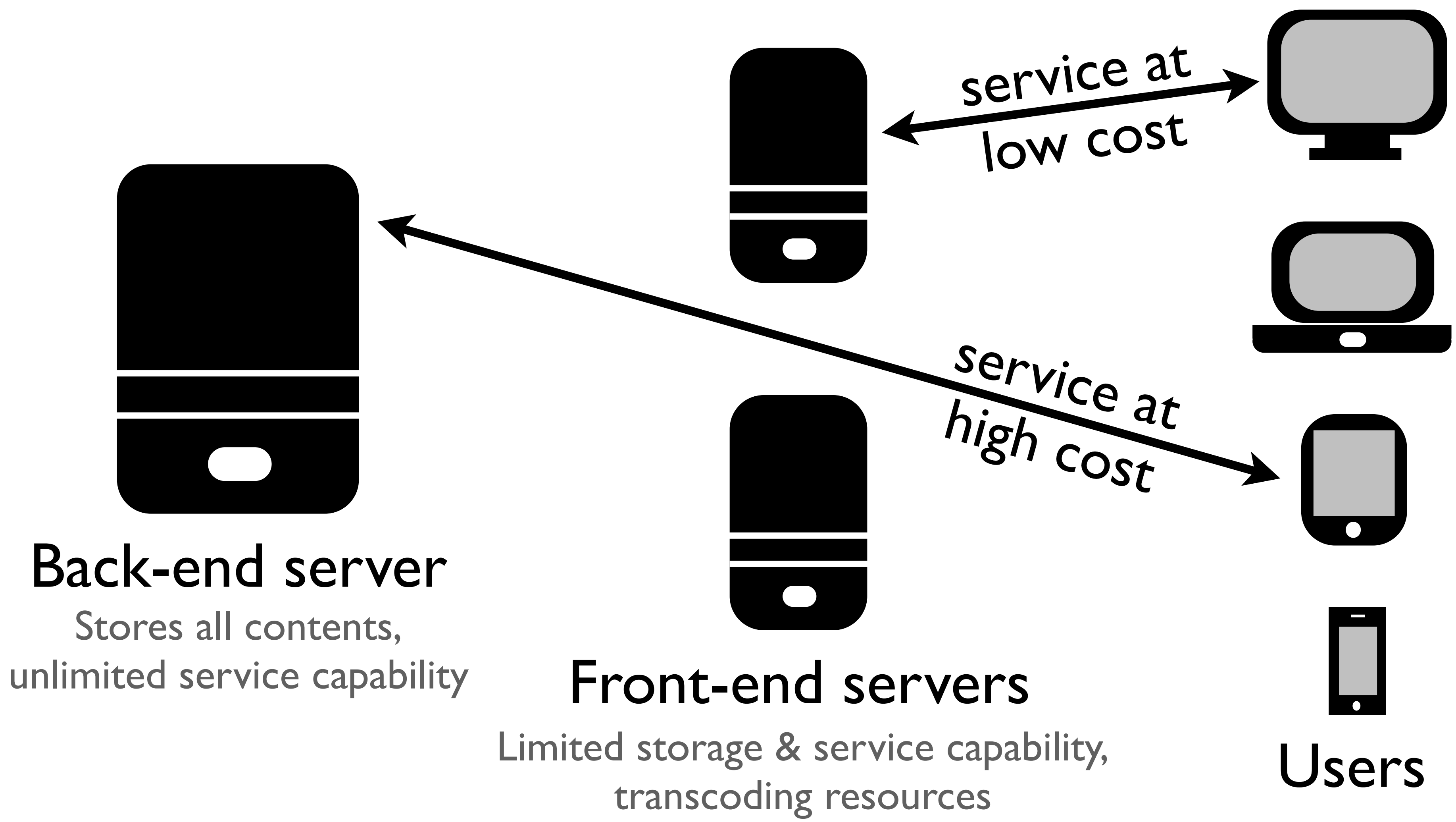
End-users

Different operating systems, screen sizes, bit-rate requirements, codec support etc.

New Challenge: Delivering content in multiple formats

New Resource: Computational power - transcoders

Content Delivery Network

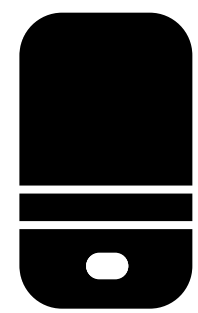


Tasks

- What to store on the front-end servers?
- How to use transcoding resources?

Setting

Front-end Servers



Front-end server

Limited storage and service capability,
transcoding resources

- n contents, n large
- Storage - Vanishing fraction of all contents ($o(n)$, e.g., $\sqrt{n}$)
- Service - Limited requests served concurrently
- Non-uniform storage and service capabilities

Setting

Cost of serving requests

1. Serve using front-end server	C_{min}
2. Fetch and serve	$C_{min} + C_{Fetch}$
3. Transcode and serve	$C_{min} + C_{Transcode}$
4. Serve using back-end server	C_{max}

No queues

$C_{min} < C_{max}$

$C_{Fetch}, C_{Transcode} > 0$

Goal: Optimize content replication on front-end servers to minimize the cost of serving requests.

Setting

Content & Format Popularity

- Heavy tailed content popularity*

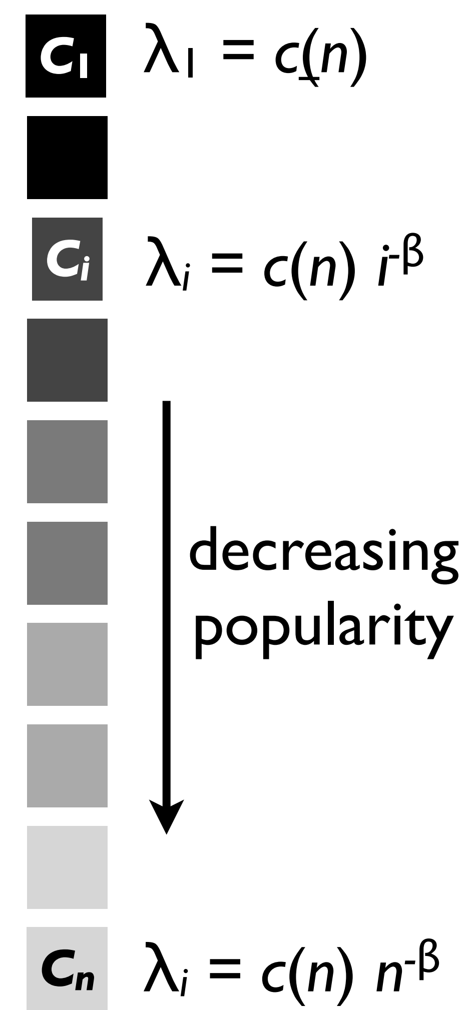
  Zipf distribution

  - Requests for $C_i \sim \text{Poisson}(\lambda_i)$
  - $\lambda_i \propto i^{-\beta}, \beta > 0$

- Format popularity

  Non-uniform & content dependent

- Supportable load



*Liu et al., **Measurement and analysis of an internet streaming service to mobile devices**, *IEEE Transactions on Parallel and Distributed Systems.*

Setting

- n contents, n large
- Heavy tailed content popularity
- K front-end servers, K is a constant
- o(n) contents on each front-end server

1. Serve using front-end server	C_{min}
2. Fetch and serve	$C_{min} + C_{Fetch}$
3. Transcode and serve	$C_{min} + C_{Transcode}$
4. Serve using back-end server	C_{max}

Goal: Optimize content replication on front-end servers to minimize the cost of serving request.

Candidate Strategies

I. Transcode on the fly* (ToF):

Store master format, transcode on demand to serve requests
e.g., VUCLIP - mobile VoD service, dynamic adaptive transcoding

II. Lazy Transcoding and Re-transcoding (LTR):**

Store transcoded versions, delete obsolete formats periodically

*U.S. Patent No. 8,869,218
**U.S. Patent No. 8,782,285

DIST-LTR

Routing	Random routing - Probability request routed to server *j* $\propto$ service capacity of server *j*
Content Replication	On a request arrival for $C_{i,f}$: Case 1 - Server busy: serve using back-end server Case 2 - $C_{i,f}$ available: serve request Case 3 - $C_{i,f}$ not available: fetch or transcode, replace content(s) not being used with $C_{i,f}$

Definition: Γ_{ALG} = Cost per request

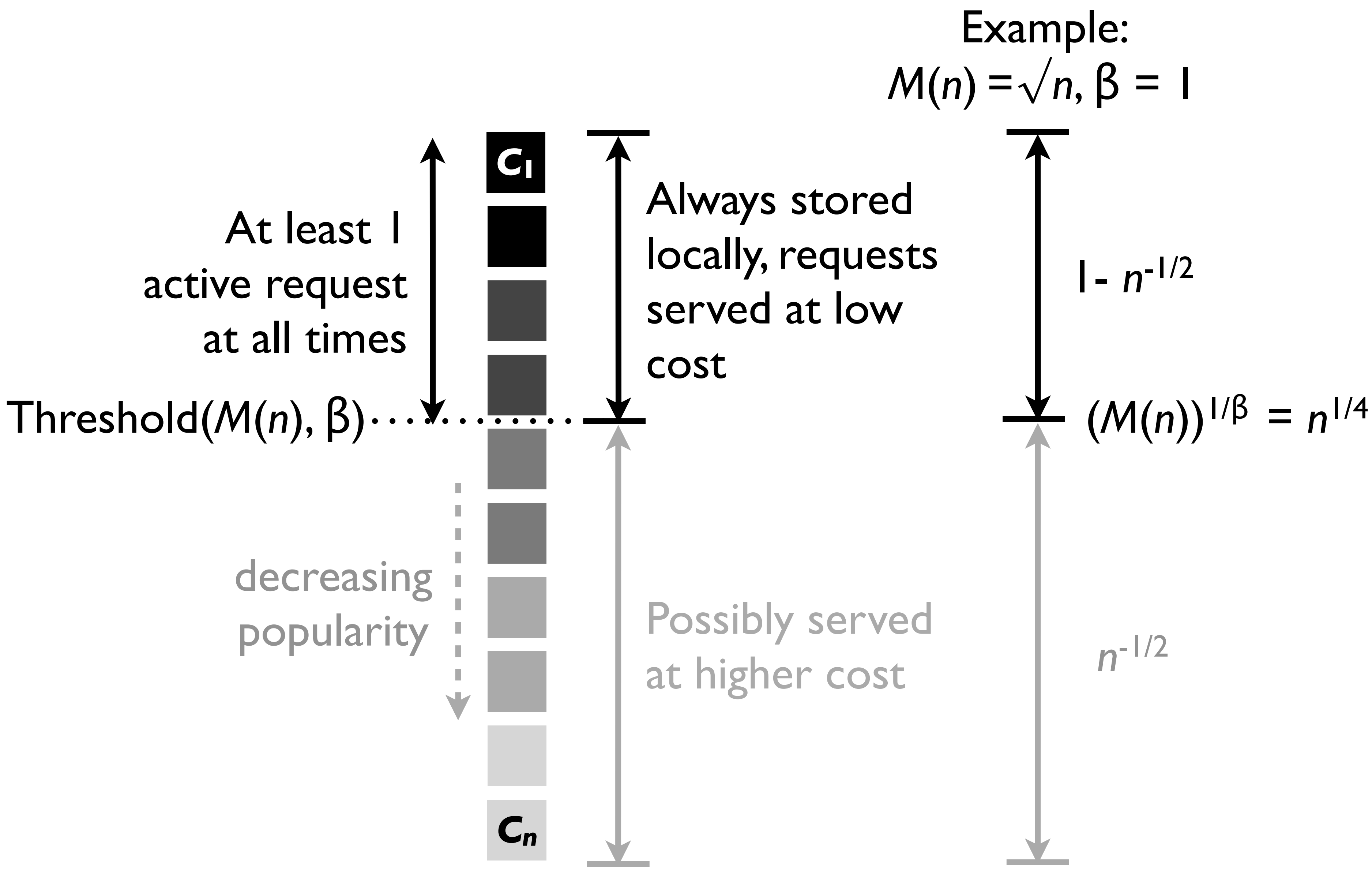
Theorem

$$\lim_{n \to \infty} E[\Gamma_{DIST\text{-}LTR}] = C_{min}$$

- "Blind" routing
- No coordination across front-end servers
- Content popularity statistics unknown

Proof Outline

Assume that the front-end server can serve $M(n)$ parallel requests
Recall: Content popularity ~ Zipf(β), β>1



Transcode on the Fly

Definitions

Γ_{ALG} = Cost per request
q = Expected fraction of requests for the master format

Theorem

$$\lim_{n\to\infty} E[\Gamma_{ToF}] \geq c_{min} + \min\{c_{Transcode}, c_{max}-c_{min}\}\,(1-q)$$

- ✦ Routing using global information
- ✦ Co-ordination across front-end servers
- ✦ Use knowledge of content popularity
- ✦ Static/adaptive content replication

Request for other formats - transcode/serve using back-end server

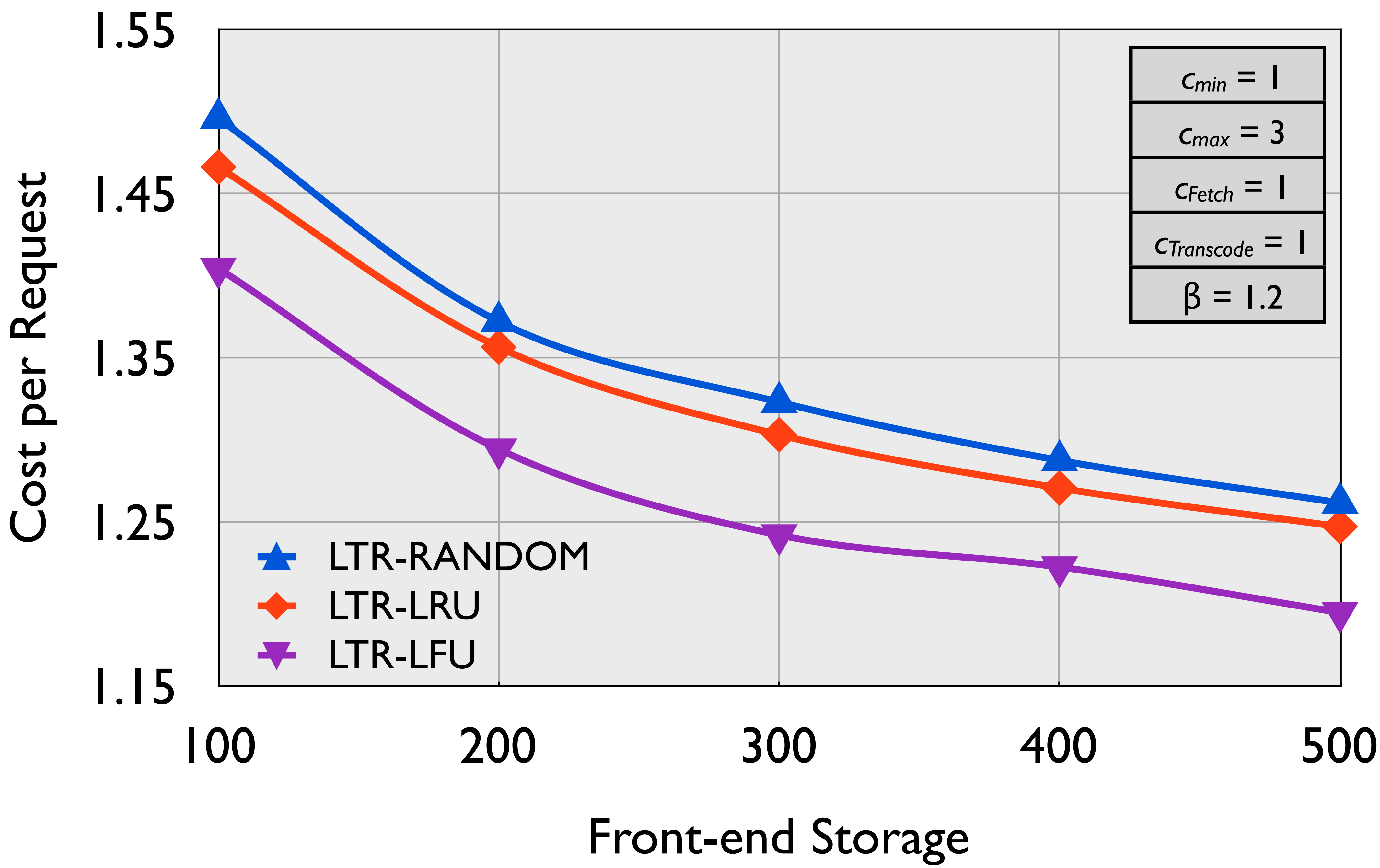
DIST-LTR

Routing	Random routing - Probability request routed to server *j* $\propto$ service capacity of server *j*
Content Replication	On a request arrival for $C_{i,f}$: Case 1 - Server busy: serve using back-end server Case 2 - $C_{i,f}$ available: serve request Case 3 - $C_{i,f}$ not available: fetch or transcode, replace content(s) not being used with $C_{i,f}$

- Randomly chosen content (LTR-RANDOM)
- Least recently used content (LTR-LRU)
- Least frequently used content (LTR-LFU)

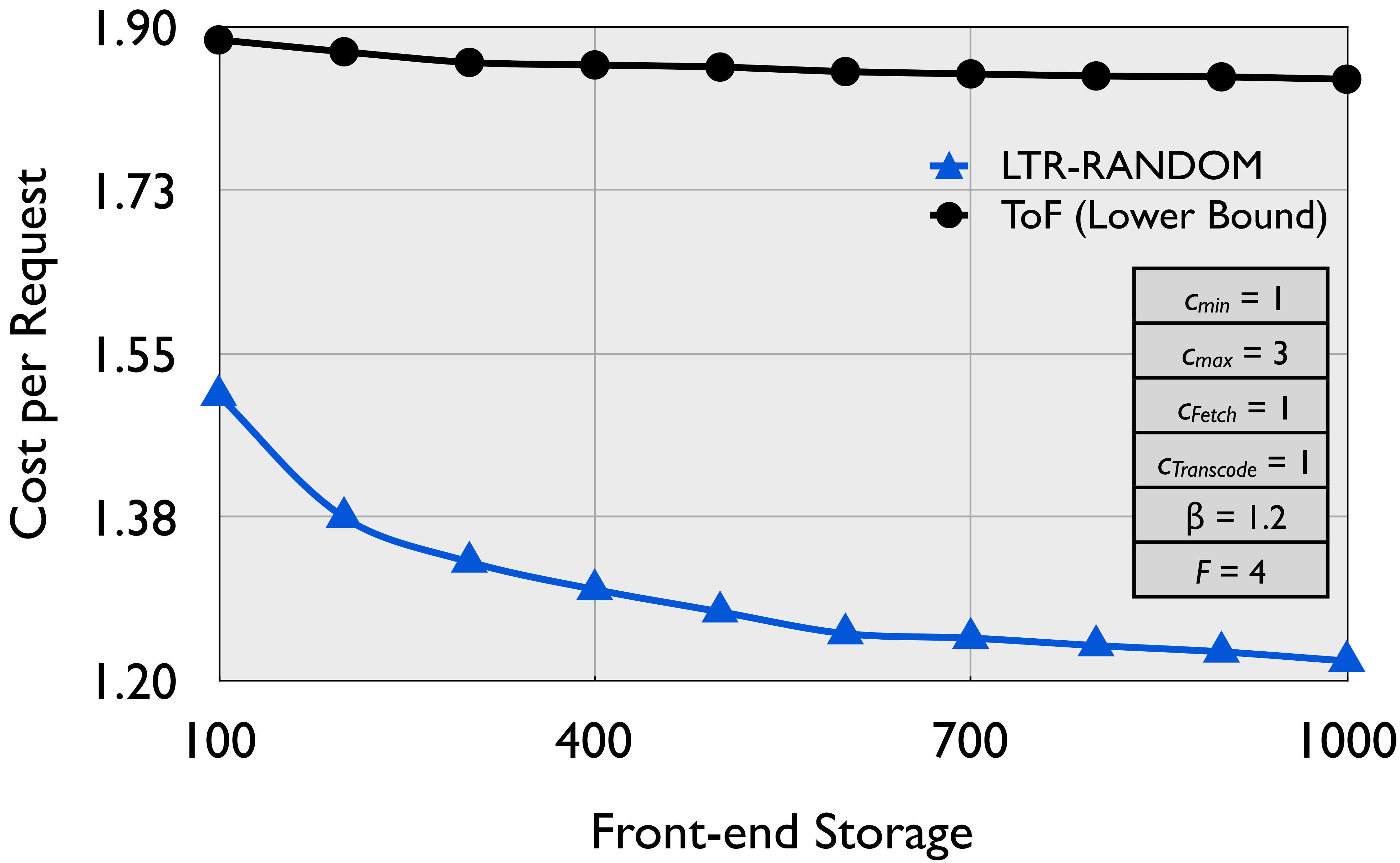
Simulations

Cost vs Zipf Parameter



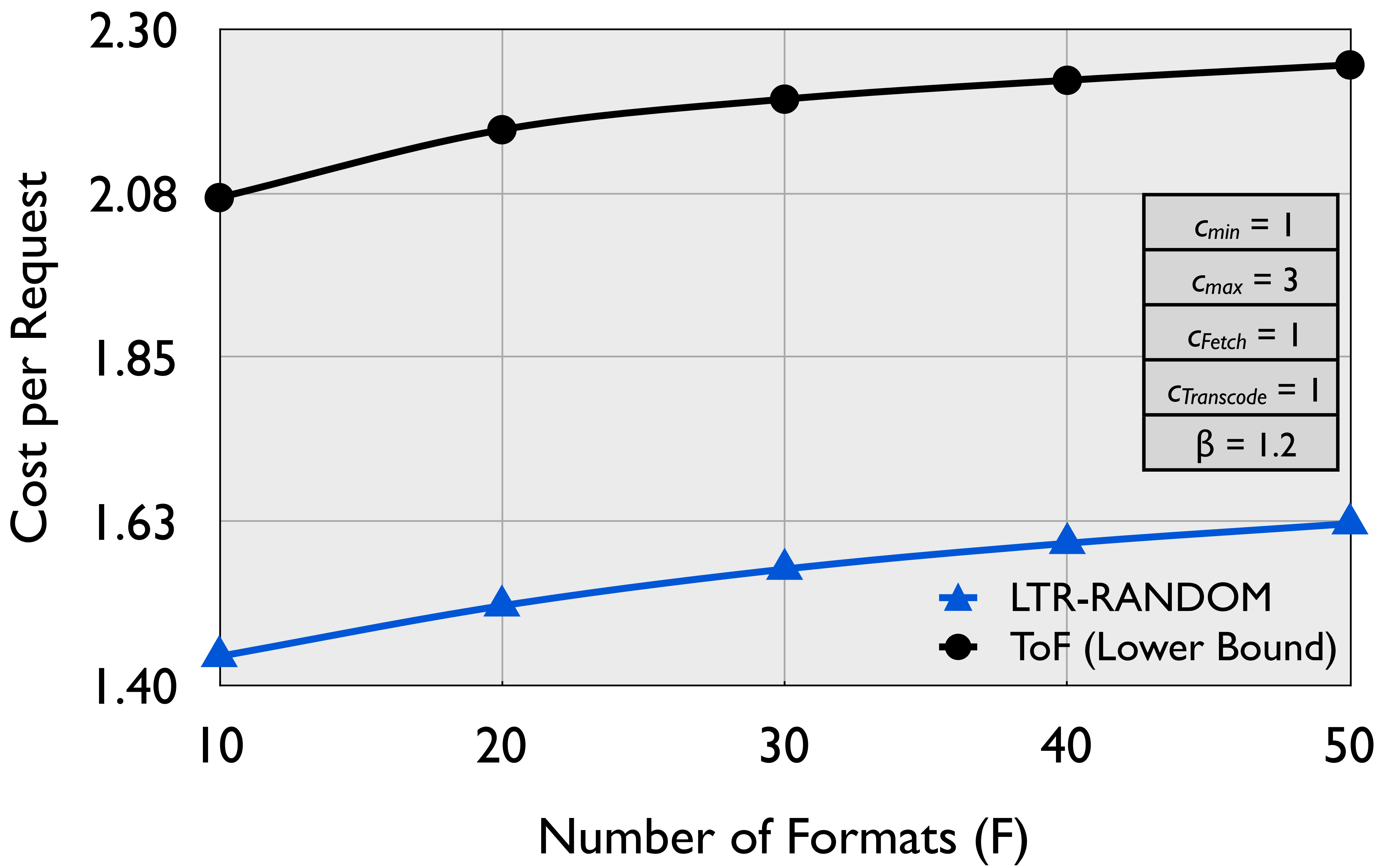
Simulations

Cost vs Front-end Storage



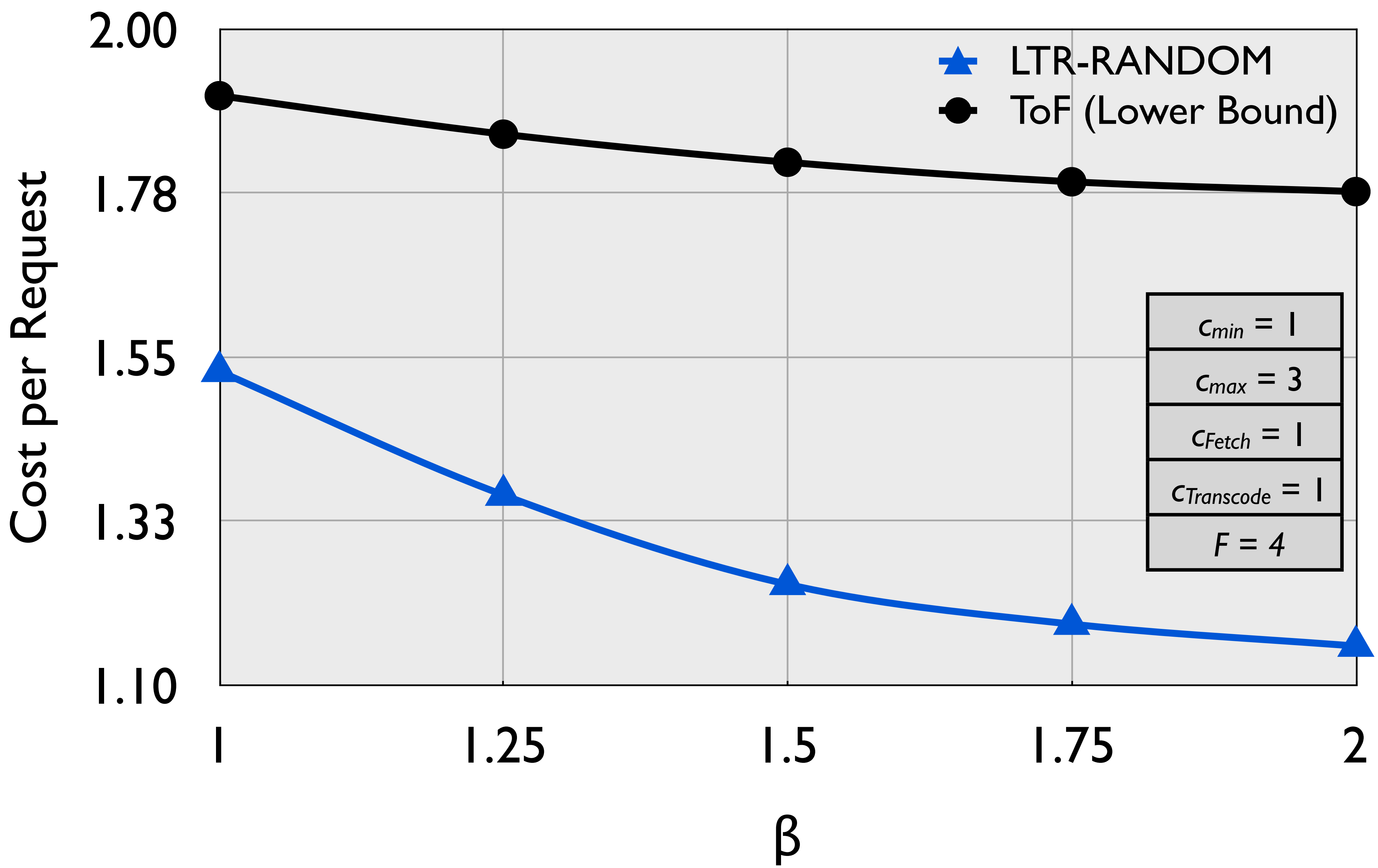
Simulations

Cost vs Number of Formats



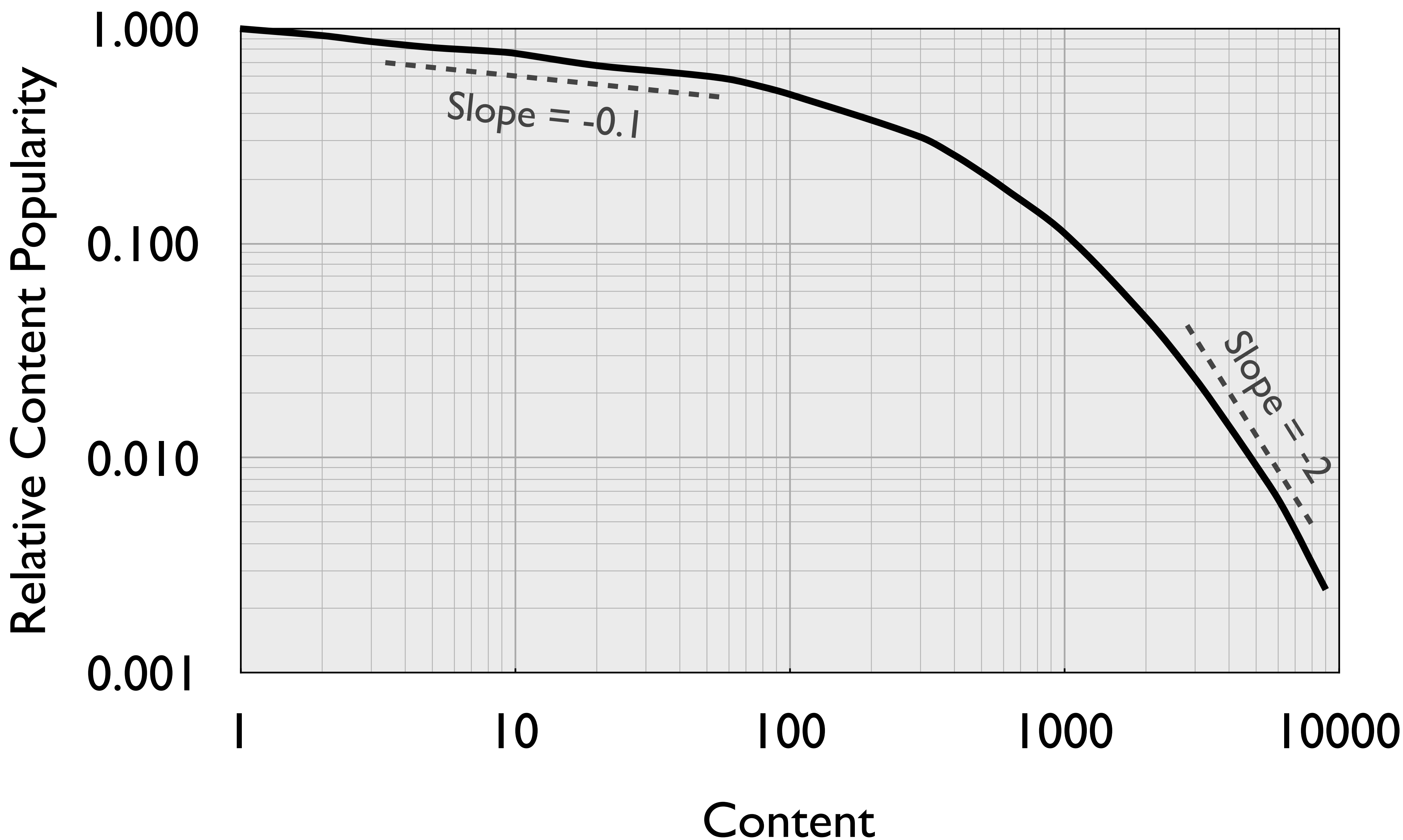
Simulations

Cost vs Zipf Parameter



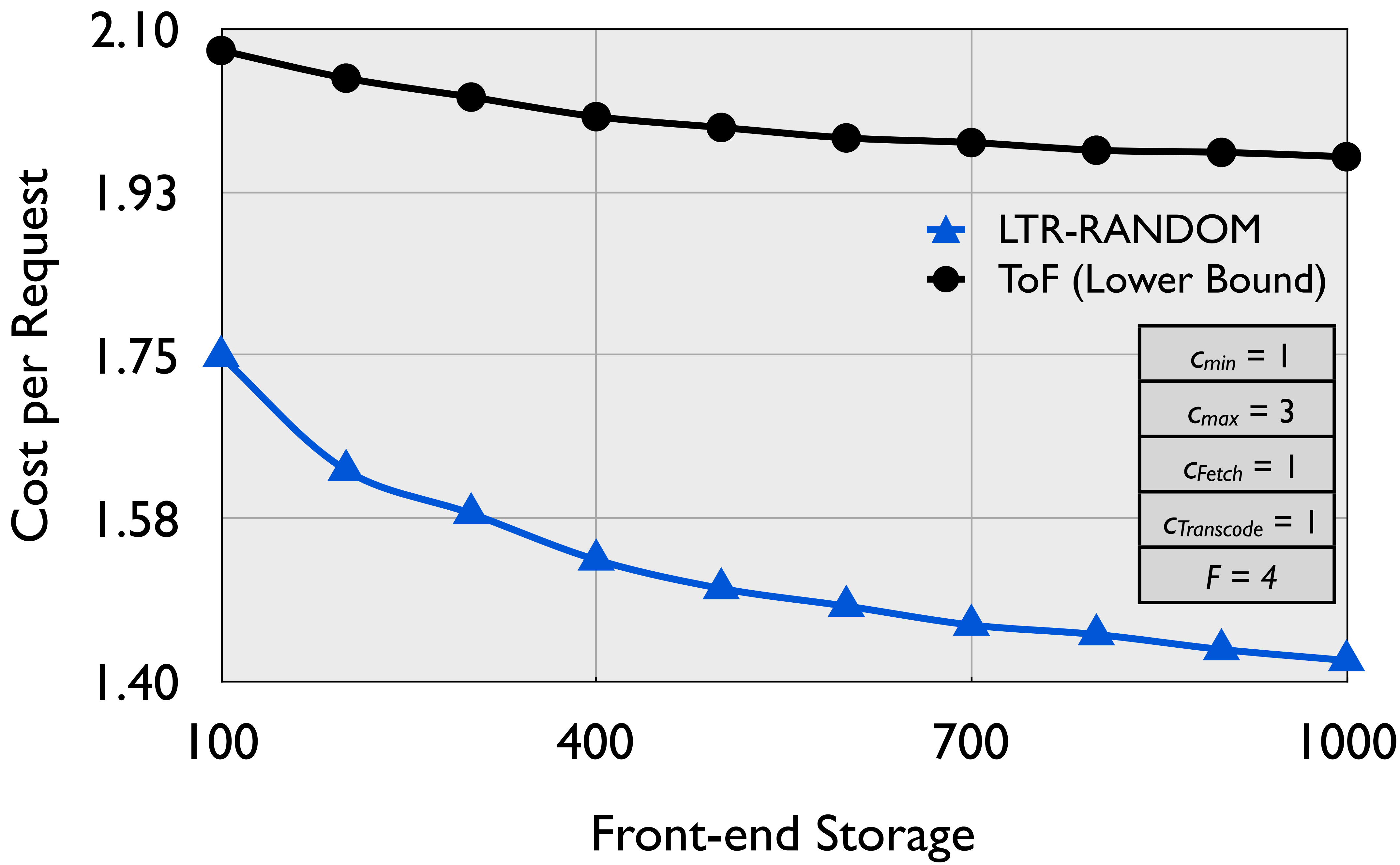
Netflix Data

Content Popularity



Simulations

Netflix Content Popularity



Related Work

Device Heterogeneity

- Measurement and analysis of an internet streaming service to mobile devices
  Liu, Li, Guo, Shen, Chen & Lan, *IEEE Transactions on Parallel and Distributed Systems*
- Joint online transcoding & geo-distributed delivery for dynamic adaptive streaming
  Wang, Sun, Wu, Zhu & Yang, *IEEE INFOCOM* 2014

Large content catalogs

- Serving content with unknown demand: the high-dimensional regime
  S.M., Ghaderi, Sanghavi & Shakkottai, *ACM Sigmetrics* 2014
- Adaptive replication in distributed content delivery networks
  Leconte, Lelarge & Massoulie, *ITC* 2015
- Bipartite graph structures for efficient balancing of heterogeneous loads
  Leconte, Lelarge & Massoulie, *Sigmetrics* 2012
- Queueing system topologies with limited flexibility
  Tsitsiklis & Xu, *Sigmetrics* 2013

Conclusions

Task - Content replication for content delivery in multiple formats

Candidate Approaches -

- *Transcode on the fly*: Store content in one high-quality master format
- *DIST-LTR*: Stores multiple formats of the same content

Results -

- The *transcode on the fly* approach is strictly suboptimal
- *DIST-LTR* is asymptotically optimal, even without coordination

Thanks