Accelerating Internet Streaming Media Delivery using

Network-Aware Partial Caching

Azer Bestavros
and
Shudong Jin

Boston University

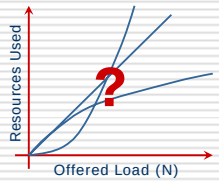
http://www.cs.bu.edu/groups/wing

International Conference on Distributed Computing Systems

Vienna, Austria — July 3, 2002

Scalable Content Delivery: Why?

- Need to manage resource usage as demand goes up
  - Server load:
    - CPU, memory, etc.
  - Network load:
    - Bytes, Byte-Hops, etc.



- Also need to worry about QoS to clients
  - Delay, response time, jitter, *etc.*

Scalable Content Delivery: How?

Replicate It!

Scalable Content Delivery: How?

- Replicate it from the client side
  - Client caching/prefetching, proxy caching, cooperative caching, server selection, *etc.*
- Replicate it from the server side
  - Servers on steroids, server farms, reverse proxy caching and CDNs, *etc.*
- Replicate it in the network
  - Network caches, multicast, anycast, traffic engineering, *etc.*

Scalable Content Delivery: What?

- Streaming content
  - Emerging as largest sink of net resources
  - Potential for savings is huge due to more predictable access patterns
  - QoS of delivery to client is key
  - Complicated by the bursty nature of network and server conditions
  - Further complicated by the increasingly peer-to-peer nature of content delivery

A gold mine of R&D problems!

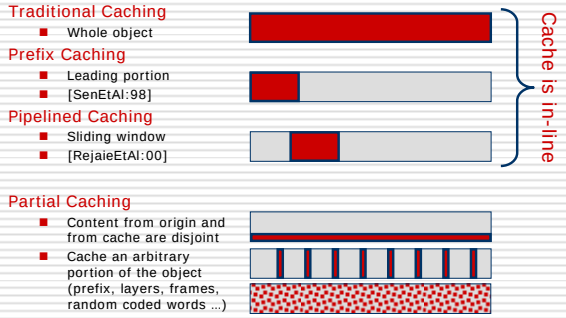
Streaming Media Replication

- Two flavors of replication
  - Caching: Replicate the artifact by storing it at an alternate location (server, CDN, proxy, or client)
  - Multicast: Duplicate content en route to destinations, either in the network (IP multicast) or at edges (end-system multicast)

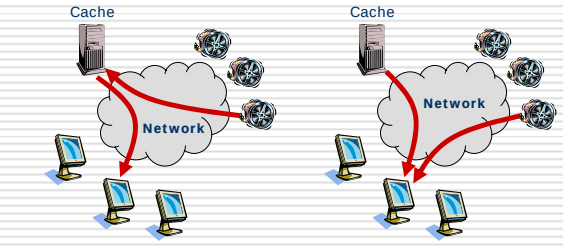
	Scalability	QoS
Caching	Poor ~ $O(0.6\,n)$	Unexplored!
Multicast	Excellent ~ $O(\log n) \leftrightarrow O(\sqrt{n})$	Inefficient

We advocate the use of caching primarily to control QoS of streaming delivery

Streaming Media Replication



Caches: Proxies vs Accelerators



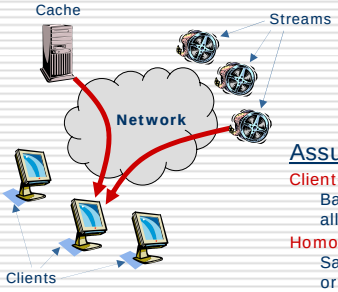
[SalehiEtAl:Sigmetrics'96][WangEtAl:Infocom'98] [SenEtAl:Infocom'99][Rejaie:Infocom'00] ...

[JinBestavros:ICDCS'02]

Partial Caching

- Cache is not the only source: Content served from many sources ("manycast")
- Provides cache management with a new dimension to decide "worth" of caching
  - All or none: Either cache A or B
  - Partial: Cache 40% of A and 60% of B
- Cache "allocation" versus "replacement"

Partial Caching: Architecture



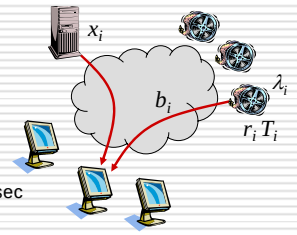
Assumptions "*for now*"

Client-side Caching:
Bandwidth from cache to all clients is large

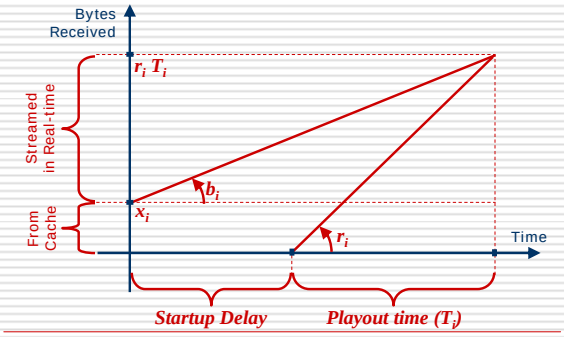
Homogeneous Clientele:
Same bandwidth from an origin server to all clients

Partial Caching: Model

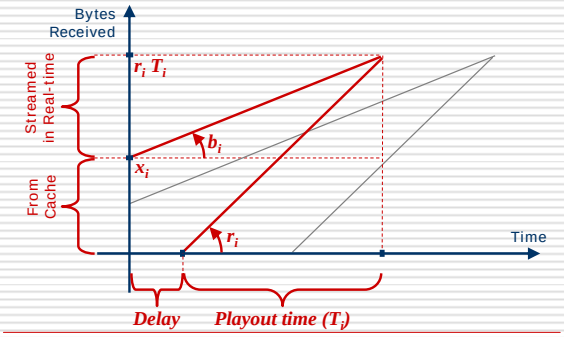
- N objects
- For object i
  - Duration = T_i sec
  - CBR = r_i Mb/sec
  - Access Freq = λ_i
  - B/W to origin = b_i Mb/sec
- Cache size is C with x_i of object i in cache



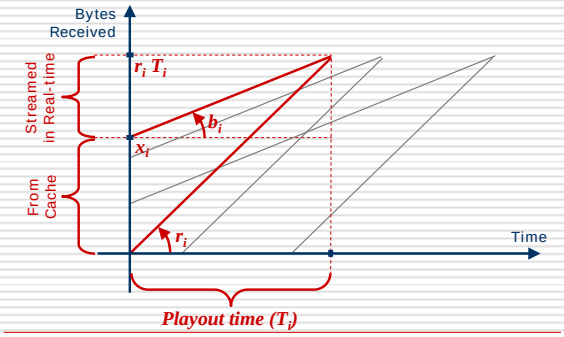
Partial Caching: Startup Delay



Partial Caching: Startup Delay



Partial Caching: Immediate Play



Partial Caching: Formulation

A constrained optimization Problem

- Populate the cache by finding a set of x_i values that minimizes the average delay (or other cost functions) over all N objects

Minimize $\frac{1}{\sum_{i=1}^{N} \lambda_i} \sum_{i=1}^{N} \lambda_i [T_i r_i - T_i b_i - x_i]^+ / b_i$

Such that $\sum_{i=1}^{N} x_i \leq C$ and $x_i \geq 0$

Partial Caching: Solution

- Can be reduced to a fractional knapsack problem [JinBestavrosIyengar:ICDCS'02]
- Optimal solution
  - Sort objects in order of their λ_i / b_i ratio
  - Cache up to $(r_i - b_i)T_i$ of each such object until cache is full
- But, how do we find λ_i and b_i? Are they even static?

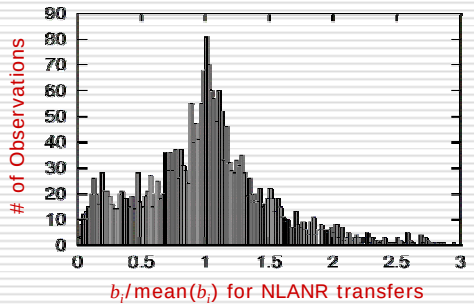
Partial Caching: Approximating λ_i

- Assuming that λ_i is stationary (over some reasonable time scale or period), it can be approximated using the frequency of access over that period.
- Cache needs to keep track of access frequencies—efficient techniques exist (sketches).
- Need to periodically update cache content based on measured frequencies.

Partial Caching: Measuring b_i

- Many techniques exist for the estimation of b_i (*e.g.*, using packet-pairs, cartouche probing, TCP equation, *etc.*) which could be done by origin servers periodically
- Need to periodically update cache content based on measured bandwidths
- But, bandwidth is not constant over any interesting time scale! Need to deal with bandwidth variability

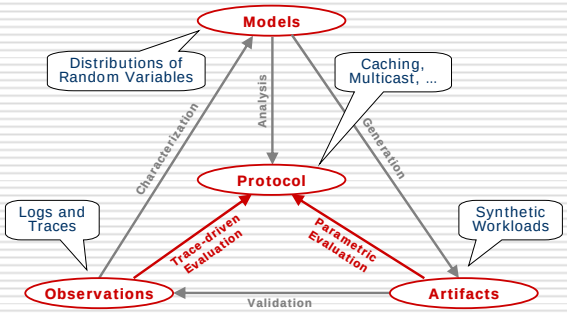
Partial Caching: b_i Variability



Partial Caching: b_i Variability

- Over Provisioning:
  - Level of over provisioning should be tied to distribution of bandwidth variability
  - Instead of using the "mean" bandwidth, one could use a more conservative estimate (e.g., 10th percentile)
- Integral Caching:
  - Over provisioning in the extreme reduces "partial" caching to "integral" caching
  - Cache entire objects with highest λ_i/b_i ratio

Evaluation Methodology

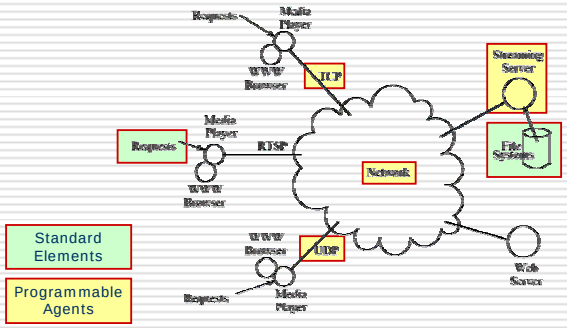


GISMO Workload Generator

- GISMO: A toolkit to generate synthetic streamed media workloads [JinBestavros:PER'02]
- GISMO generates
  - A set of "placeholder" streaming media objects, which can be installed on servers
  - Requests to these objects, initiated by clients subject to a prescribed access model

http://csr.bu.edu/gismo

GISMO: Components



GISMO: Standard Parameters

	Parameter	Setting
Requests	Popularity	Zipf-like
	Temporal Correlation	Truncated Pareto
	Seasonal Access Patterns	User-defined
	Partial Access	Truncated Pareto
Objects	Media Object Size (S)	Power Law
	VBR Long-Range Dependence	Self-similar
	VBR Marginal Distribution - Body	Lognormal
	VBR Marginal Distribution - Tail	Pareto

Partial Caching: Performance

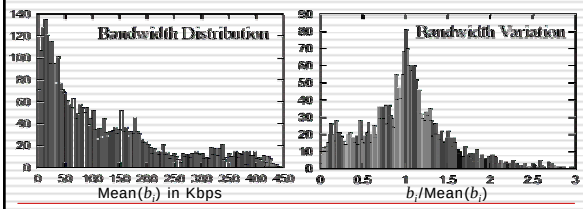
- Used GISMO to generate synthetic workload (objects & request stream)

Number of Objects	5,000
Object Popularity	Zipf-like
Number of Requests	100,000
Request Arrival Process	Poisson
Object Size	Lognormal, $\mu = 3.85, \sigma = 0.56$
Object Bit-rate	2KB/frame, 24 frames/sec.
Total Storage	790 GB

Partial Caching: Performance

- Implemented GISMO agents to model network bandwidth and UDP transfers

Bandwidth Distribution	NLANR logs
Bandwidth Variation	NLANR logs and measurement



Partial Caching: Performance

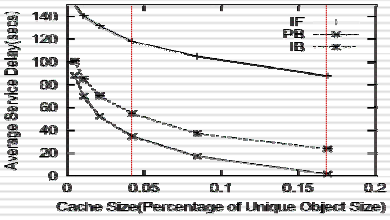
Algorithms

- Partial Bandwidth-based (PB)
- Integral Bandwidth-based (IB)
- Integral Frequency-based (IF)

Metrics

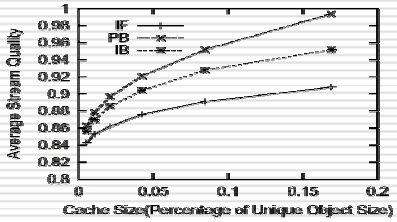
- Average Service Delay
  Average playout delay over all requests
- Average Stream Quality
  % of stream that yields immediate playout
- Traffic Reduction Ratio
  Ratio of traffic without cache to that with cache

Partial Caching: Performance



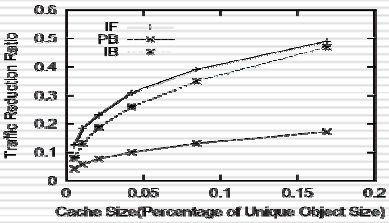
Partial Caching significantly improves the timeliness of stream delivery!

Partial Caching: Performance



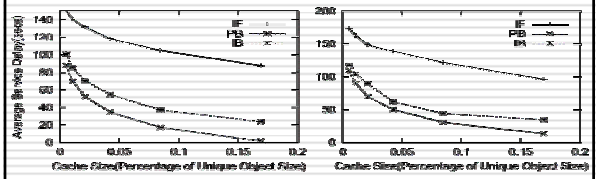
Partial caching improves immediate playout quality.

Partial Caching: Performance



Partial caching does not result in significant reduction in traffic (a feature not a bug!)

Partial Caching: Performance



Low bandwidth variability | High bandwidth variability

Bandwidth variability reduces relative performance of PB vs IB (as expected)

Partial Caching: Extensions

Dealing with heterogeneous clientele

- Cluster clients into "equivalence classes"
  - Using BGP data [KrishnamurthyEtAl:Infocom'01]
  - Using DNS clustering [BestavrosMehrotra:WWC'01]
  - Using MINT caricatures [BestavrosEtAl:Infocom'02]
- For each equivalence class j estimate λ_{ij} and b_{ij} and minimize

$$\frac{\sum_i \sum_j \lambda_{ij} [T_i r_i - T_i b_{ij} - x_i]^+ / b_{ij}}{\sum_i \sum_j \lambda_{ij}}$$

Partial Caching: Extensions

Dealing with finite cache-to-client bandwidth

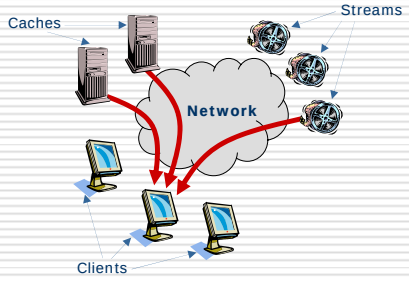
- If cache-to-client bandwidth for class j is $b_j^{'}$ then playout delay is:

$$\max\left(\frac{T_i r_i - T_i b_{ij} - x_i}{b_{ij}}, \frac{x_i}{b_j^{'}}\right)$$

- Reformulation of optimization problem is possible

Partial Caching: Extensions

Dealing with multi-cache downloads



Partial Caching: Multi-Cache

- For a given class of clients j, caches are sorted in order of $b_j^{'}$
- Caches as a hierarchy: Cache at one level is origin server to those at lower levels
- Problems:
  - Need to account for bottleneck bandwidth sharing between caches and clients
    - ➔ Estimate shared bottleneck bandwidth [Harfoush:PhD'02]
  - Need to make sure that content received from multiple caches is not redundant
    - ➔ Use RS or Tornado encoded content [ByersEtAl:Sigcomm'98]

Take-Home Conclusions

- New caching architectures in wake of the proliferation of streaming media content
- Caching cannot be studied in isolation of other enabling protocols/technologies (*e.g.*, multicast, network measurements, coding …)
- Characterization (of access patterns, topology, *etc.*) is key to the evaluation of novel protocols and architectures

Related Links

http://www.cs.bu.edu/groups/wing

http://www.cs.bu.edu/~best

Scalable Content Delivery: What?

- Static bulk content
  - Early focus of scalable content delivery work
  - Moderate savings (~ 40% max) possible
  - Diminishing % of today's web transactions
- Dynamic bulk content
  - Need to worry about freshness of content
  - Fairly straightforward…

Case closed for bulk content replication!

Scalable Content Delivery: What?

- Dynamic "tailor-made" content
  - Not a unidirectional content exchange!
  - Replicate assembly process vs content
  - Complicated by issues of consistency, coherence, trust, security, code safety, *etc.*

Wide open!

On Consistency [BradleyBestavros:GI'02]

June 21st 03:25 am