Accurate Modeling of Cache Replacement Policies in a Data Grid

Ekow J. Otoo and Arie Shoshani
Scientific Data Management Group
Lawrence Berkeley National Laboratory
Berkeley, California

Main Results

- Caching of very large files (objects) on disks exhibit properties that are distinct from traditional web-caching and file-buffering.
- Performance metrics - Hit Ratio, Byte-Hit Ratio are not sufficient criteria for deciding which cache replacement policy is best.
- Average Retrieval Cost Per Reference is proposed as an appropriate metric for comparing cache replacement policies.
- We present an accurate framework for modeling and comparing cache replacement policies.
- Using this framework on real and synthetic workloads we conclude that the two best policies for caching in grid environments are:
  - Least cost beneficial based on K backward references (LCB-K) and
  - Greedy dual-size (GDS)

Outline

- Environment for disk caching
  - Storage Resource Manager (SRM)
- Characteristic properties
- The framework for the correct modeling of cache replacement policies
- Some implementation details
- Comparison of some cache replacement policies
- Discussion of our experiments and empirical results
- Conclusion and future work

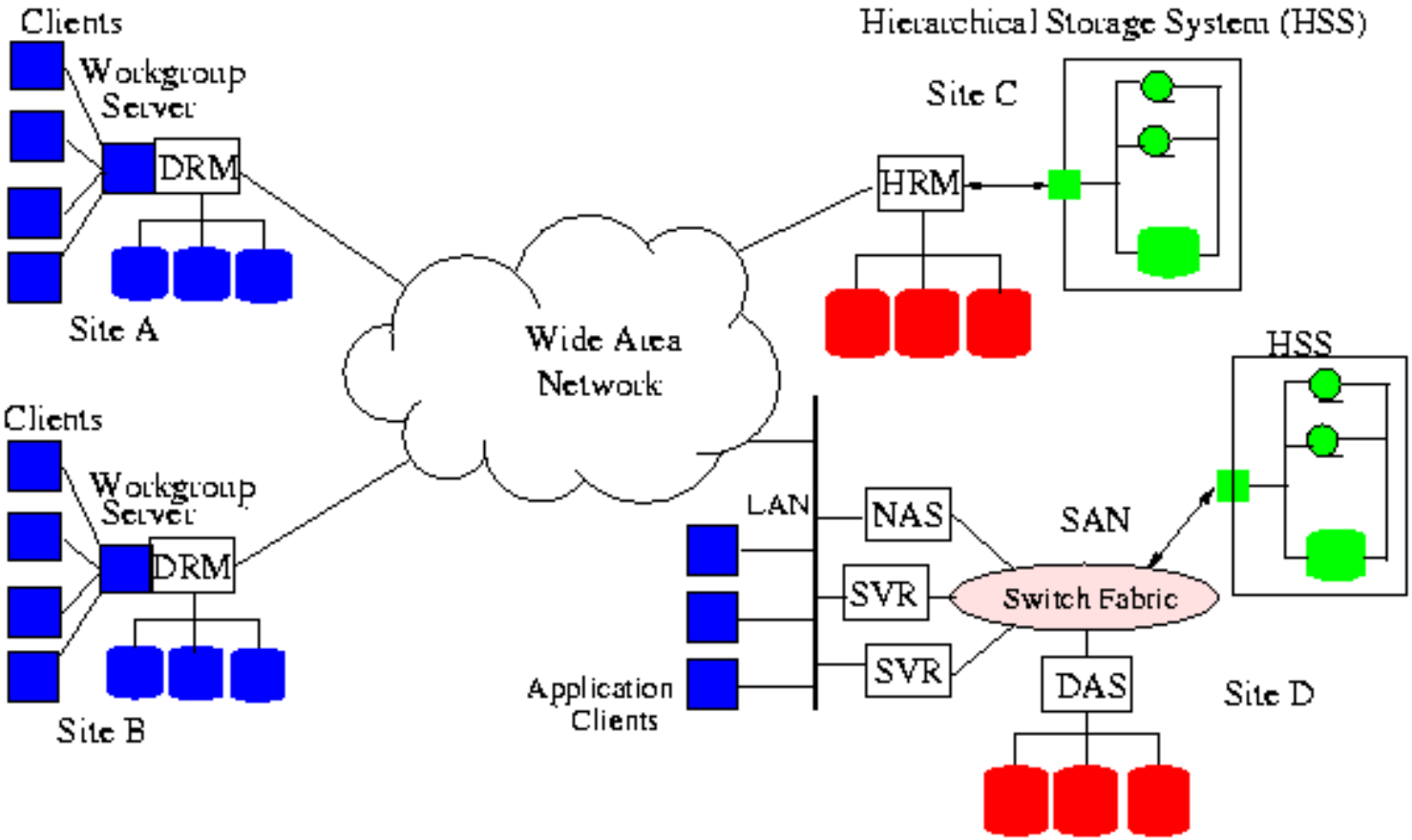
Environment

- **Caching is presented in the context of a storage resource manager (SRM) in data-grids.**

- **An SRM:**
  - **A middle-ware component to facilitate the sharing of data and storage resources**
  - **File Caching/Staging – effective means to alleviate network traffic by minimize data access latency and transfer costs.**

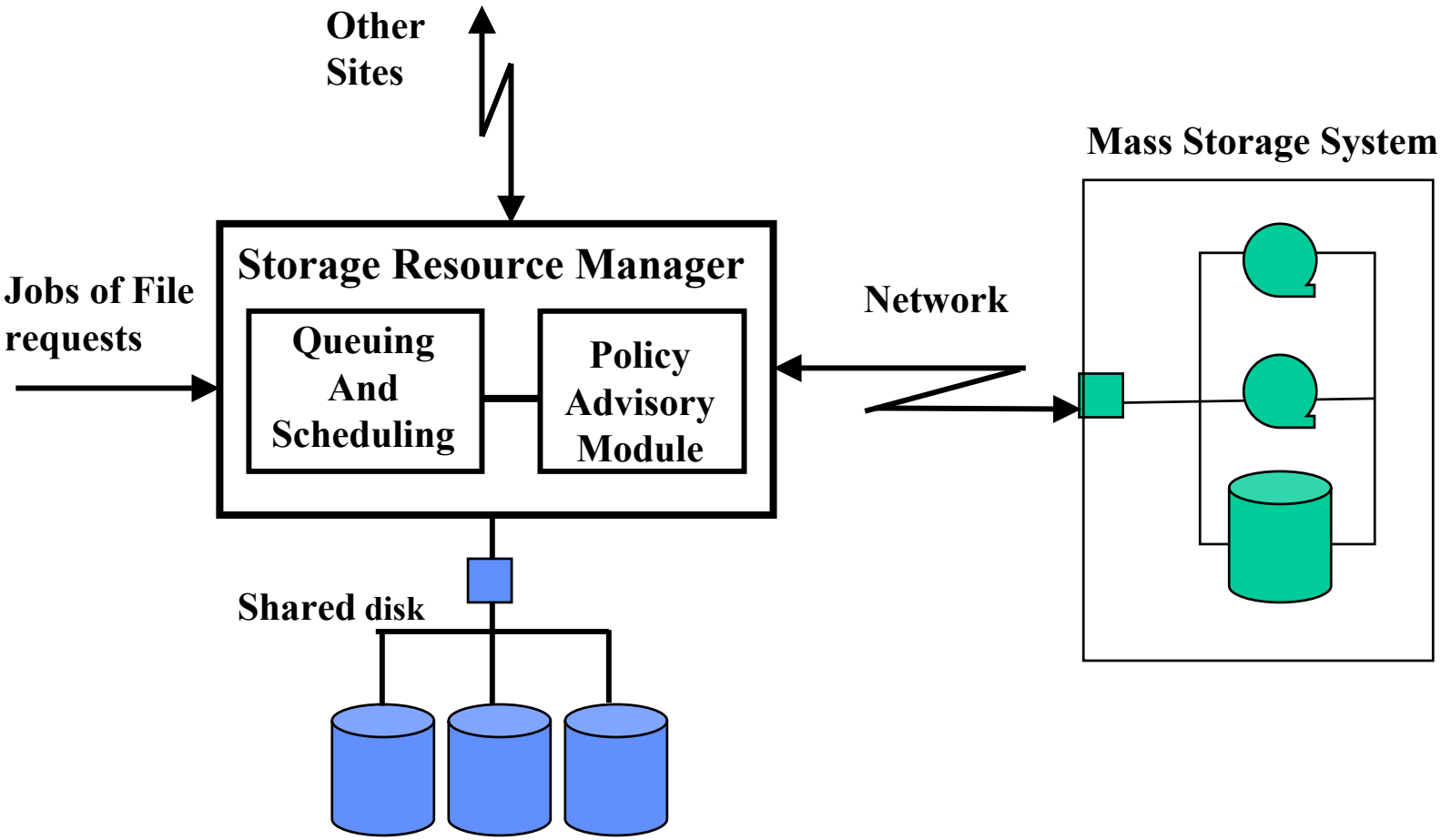
- **An SRM may be specialized**
  - **A disk resource manager (DRM), manages disk resources**
  - **A hierarchical resource manager (HRM) services request to tertiary storage.**

Use of Storage Resource Managers



Managing File Requests at Single Site

Use of a Shared Disk for Accessing Data from Remote MSS



Policies in SRMs

- **Service Policy**
  - Determines which job to service next
- **Caching Policy**
  - Determines which file of the selected request should be fetched into the disk cache
- **Cache Replacement Policy – Focus of this paper**
  - Determines the file in cache to be evicted
- **Replica Selection Policy**
  - Determines which replica of a file to fetch into disk
- **Resource Quota**
  - Determines how much resource should be made available to a single job or a client's session

Characteristic Properties

Caching in SRM

- Variable size files/objects ~gigabytes
- Long access latency
- Transfers in mins. to hrs.
- Caching is Mandatory
- May have 100 ~ 1000s of files per request.
- Predominantly read-only
- May require multiple files (bundles), in cache.

Web-Caching

- Variable size files/objects ~megabytes
- Short data access latency
- Relative short under the same conditions
- Caching is optional
- Typically one/few objects per request
- Cache coherence – required

- Caching is effective under high locality of reference in a limited storage capacity

Principle of Cache Replacement

- **General idea of a cache replacement policy**
  - Rank each object ι in cache according to some utility function $\phi(\iota)$ and evict the candidate with min (or max) $\phi(\iota)$.
  - The problem is in defining $\phi(\iota)$.

- **Caching under virtual memory paging**
  - Fixed page size, constant cost
  - [Belady (1966):] Optimal replacement if object replaced is one with the farthest future reference.
  - Use of history of references to approximately define $\phi(\iota)$.
    - Least Recently Used (LRU) – keeps track of last references
    - Least Frequently Used (LFU) – maintains reference counts

File Buffer Management & Tertiary Storage to Disk Caching

- **Buffer Management**
  - **LRU-K [O'Niel et al. 1993]: Makes use of the time of Kth backward reference to rank objects and evicts one with minimum value.**
  - **Greedy Dual [Young 1991]: A generalization of LRU**

- **Tertiary Storage to Disk**
  - **Hazard Rate Optimal [Olken 1983]: Ranks object according to a hazard rate function at time t since it was last brought into cache**

$$\phi_i(t) = \frac{h_i(t) * c_i(t)}{s_i}; \qquad h_i(t) = \frac{f_i(t)}{1 - F_i(t)}$$

Where $c_i(t)$ is the cost of retrieving the object and s_i is the size of the object.

Web-Caching

- **Self Adjusted LRU [Aggarwal and Yu, 1999]**
- **Greedy Dual Size (GDS) [Cao and Irani, 1997]**

Greedy Dual Algorithm

Initialize: Set L ← 0

Consider request for an object p

if* p is already in cache *then

set $\phi(p) \leftarrow L + c(p)/s(p)$

else

while* there is not enough room for p *do

set $L \leftarrow \min_{q \in cache} \phi(q)$

evict q such that $\phi(q) = L$

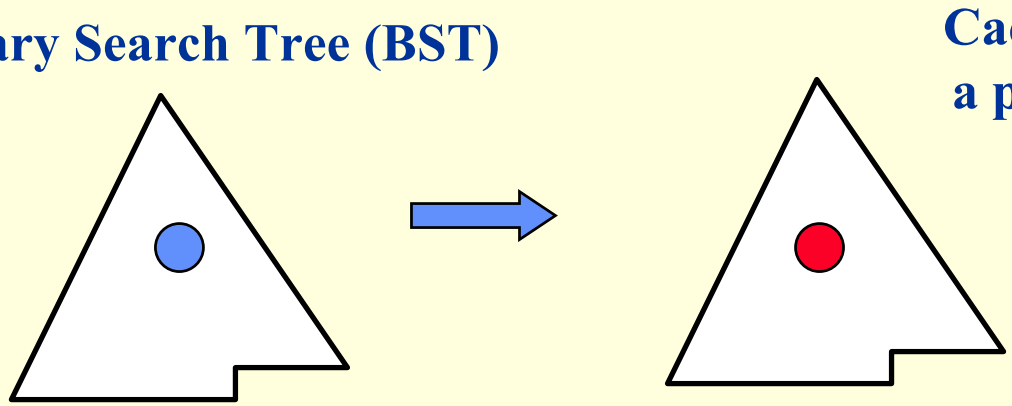
endwhile

Bring p into cache and set $\phi(p) \leftarrow L + c(p)/s(p)$

endif

Simulating GDS

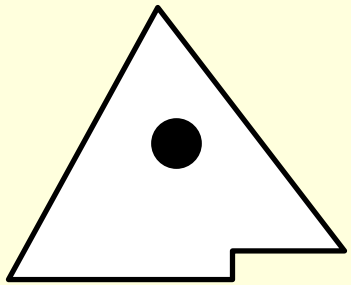
- Given a reference stream $r_1, r_2, r_3, \ldots, r_N$ and a specified cache size.
- When no delays are considered we can maintain the information of referenced objects in a balanced binary search tree and the actual cached objects in priority queue.



Greedy Dual Size

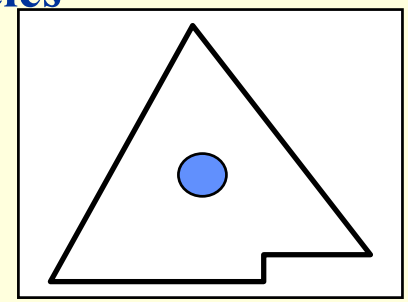
When Delays are Considered

- Each reference time of a file in the reference stream is composed of five event times :
  - Arrival time
  - Time when file caching starts
  - Time when file caching ends
  - Time when processing starts
  - Time when processing ends

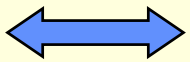


Binary Search Tree
Active lifetime: T

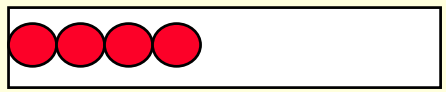
Varies with different policies



PQ of unpinned files in cache



A vector of pinned files in cache



Using State Diagram with Conditional Transition

States	Event Types						
	E0 Admit	E1 Start C	E2 End C	E3 Start P	E4 End P	E5 Evict	E6 Clear
S0: Start Not Ref.	$F_{\{0,0\}}()$ /S1						
S1: Ref. ¬ Cached		$F_{\{1,1\}}()$ / Cond(S3)					$F_{\{1,6\}}()$ / Cond(S0)
S2: In Cache but ¬ Pinned		$F_{\{2,1\}}()$ / S4				$F_{\{2,5\}}()$ / Cond(S1)	
S3: Space Reserved & Caching		$F_{\{3,1\}}()$ / S3	$F_{\{3,2\}}()$ / Cond(S4)				
S4:Cached & Pinned		$F_{\{4,1\}}()$ / S4	$F_{\{4,2\}}()$ / S4	$F_{\{4,3\}}()$ / S4	$F_{\{4,2\}}()$ / Cond({S3 |S4})		

LCB-K Caching Policy in Grid Environment

- **The replacement policy we advocate is based on a cost-beneficial function computed at time t_0 as**

$$\phi_i(t_0) = \frac{k_i(t_0)}{t_0 - t_{-k}} * \frac{g_i(t_0) * c_i(t_0)}{s_i}$$

t_0 is the current time,
$k_i(t_0)$ is the count of references for file *i* up to max of K
$c_i(t_0)$ is the cost in time of accessing the file *i*,
s_i is size of file *i*.
$g_i(t_0)$ is the total count of references to the file *i* over its active time T.
t_{-K} is the time of the k^{th} backward reference.

Performance Metrics

- **Hit Ratio is defined as** $\frac{\#\,Hits}{\#\,References}$

- **Byte-Hit Ratio is defined as** $\frac{\text{Total Size of Data Hit}}{\text{Total Size of Data Referenced}}$

- **Average Retrieval Cost Per Reference is defined as**

$$\frac{\text{Total Time of all Retrievals}}{\#\,\text{References}}$$

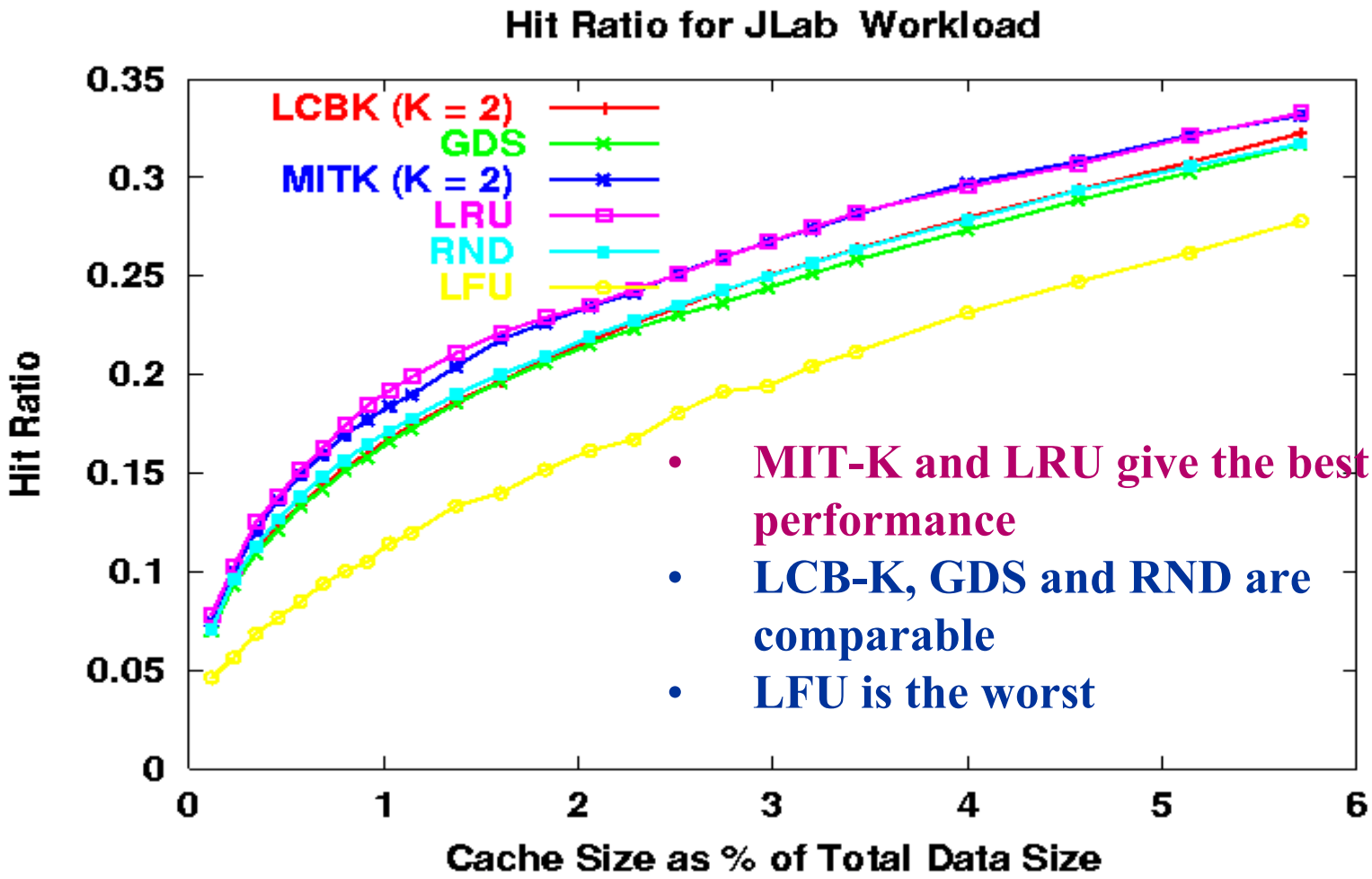
Workloads and Simulation Parameters

- **Workload from Jefferson Nat'l. Accelerator Facility (JLab)**
  - **A six month trace log of file accesses to tertiary storage**
  - **Replacement policy used in JLab is LRU**
  - **Log contains batched requests**
- **Workload from Nat'l. Energy Research Scientific Computing (NERSC)**
- **Synthetic workload based on JLab's log**
  - **250,000 files with large sizes uniformly distributed between 500K to 2.147 GBytes**
  - **Inter-arrival time is exponentially distributed with mean 90 sec**
  - **Number of references generated is about 500,000**
  - **High locality of reference**

Replacement Policies Compared

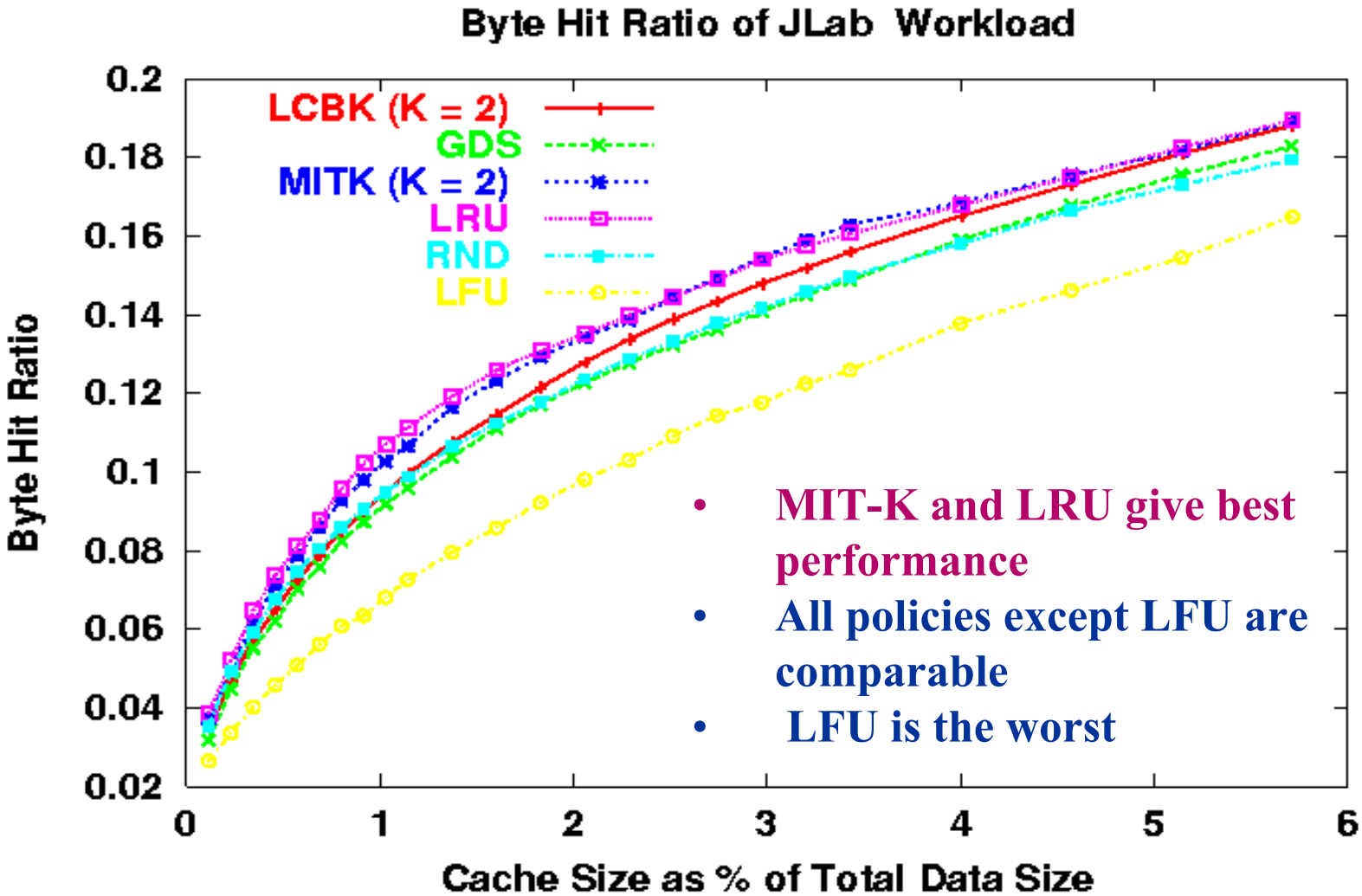
- RND: Random
- LFU: Least frequently used
- LRU: Least recently used
- MIT-K: Maximum inter-arrival times based on K backward references.
- LCB-K: Least cost beneficial based on K backward references.
- GDS: Greedy dual size
- Active lifetime T, of a file is set at five-days
- Results derived with variance reduction technique

Simulation Results



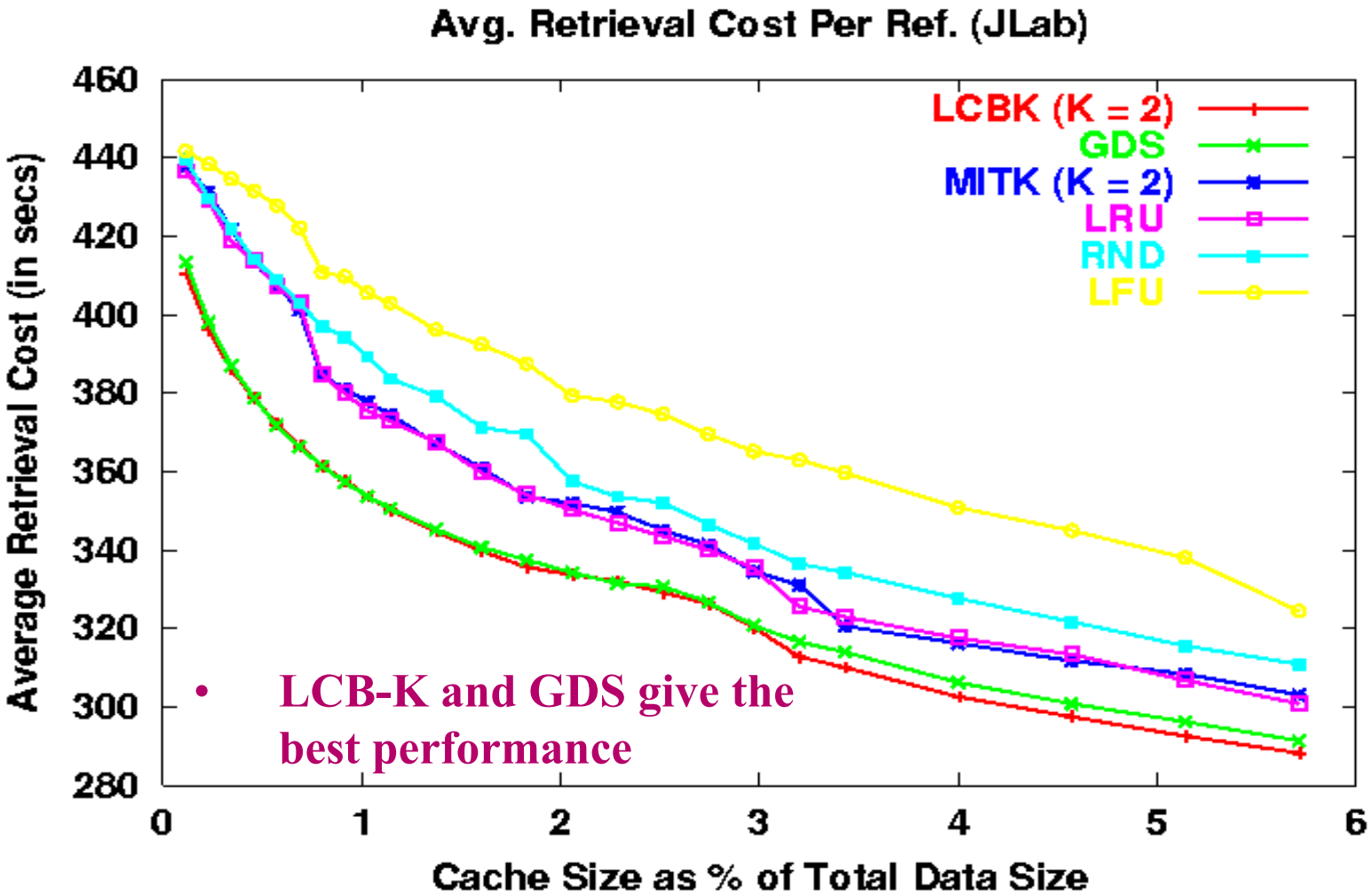
- MIT-K and LRU give the best performance
- LCB-K, GDS and RND are comparable
- LFU is the worst

Simulation Results - 2



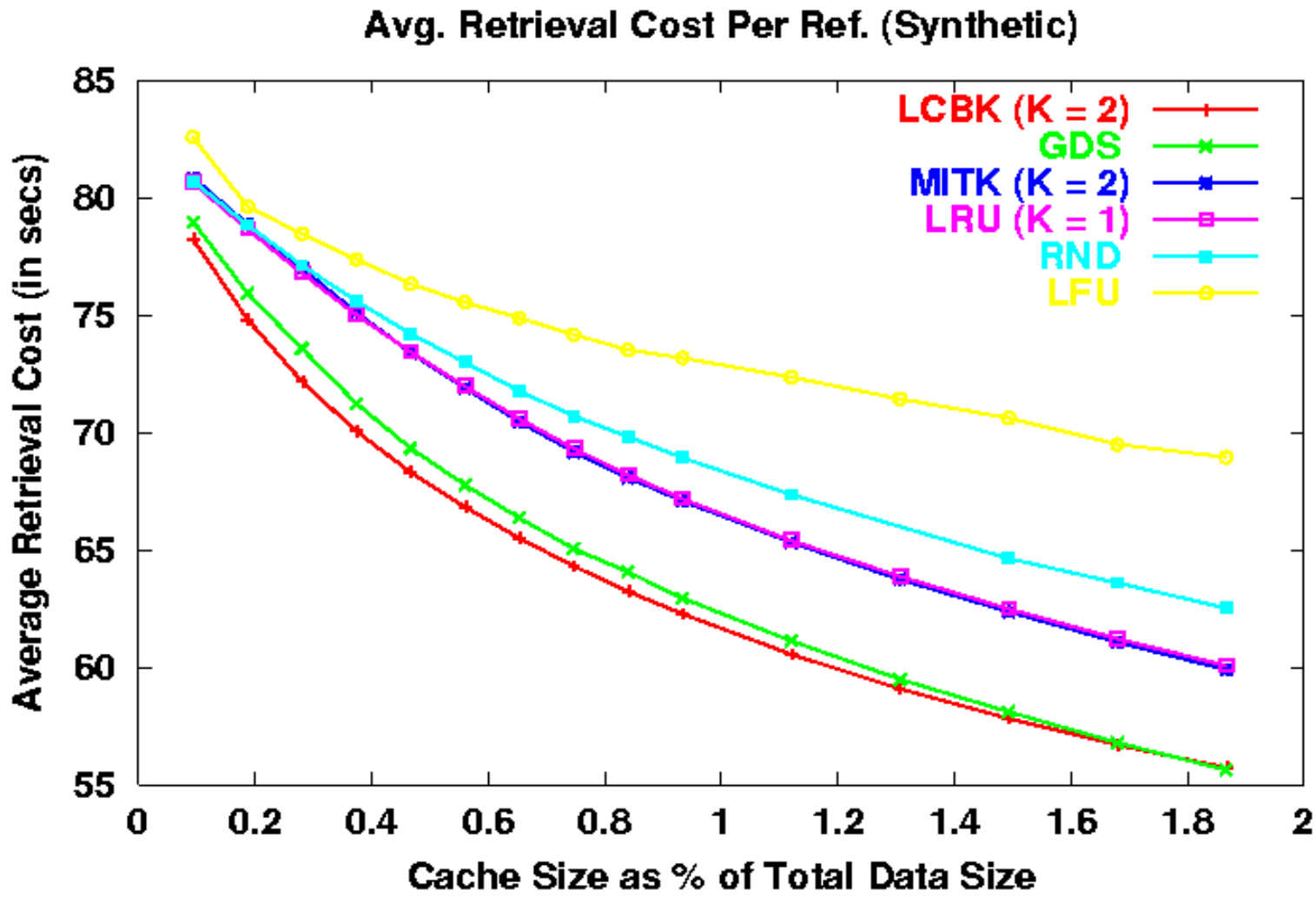
- MIT-K and LRU give best performance
- All policies except LFU are comparable
- LFU is the worst

Simulation Result - 3

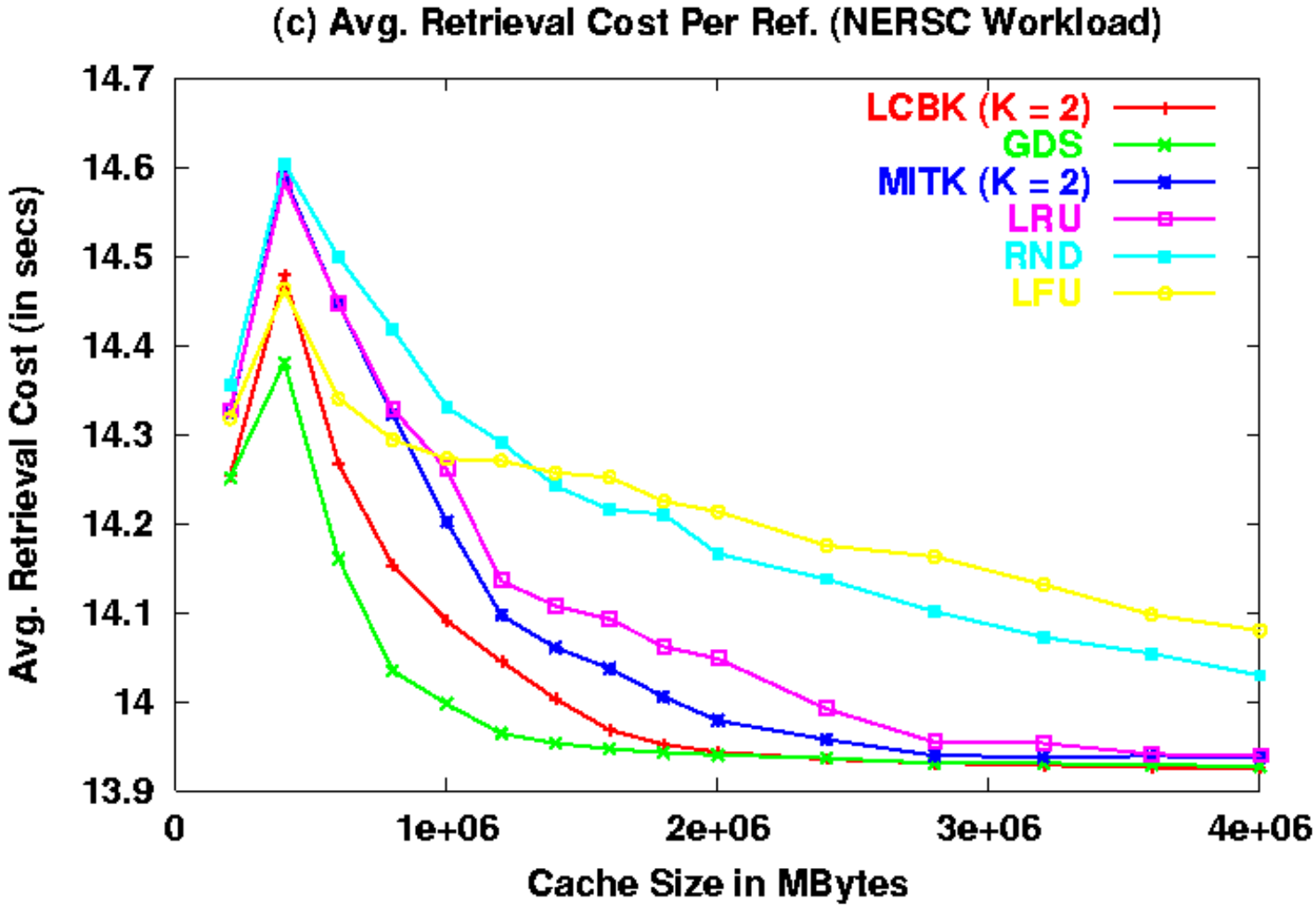


- LCB-K and GDS give the best performance

Simulation Results - 4



Simulation Results - 5



Conclusion and Future Work

- We have presented a realistic model for evaluating cache replacement policies taking into account delays at the data sources, transfers and processing.
- Used it for different policies with synthetic and real workloads of file accesses to mass storage systems.
- Worthwhile replacement policies for storage resource management on the GRID are LCB-K and GDS.
- An effective caching technique can be significant in reducing network traffic and load at data sources.

Conclusion and Future Work

- **The work has application in general storage resource management being implemented for deployment in data-grids**
- **This work is being used in an implementation of Policy Advisory Module (PAM) for SRMs**
- **Used in the framework for studying combinations of policies - service, caching, replacement, replica selection, etc., for best performance for SRMs**