2003 IEEE International Conference on CLUSTER COMPUTING HONG KONG

An Effective Disk Caching Algorithm in Data Grid

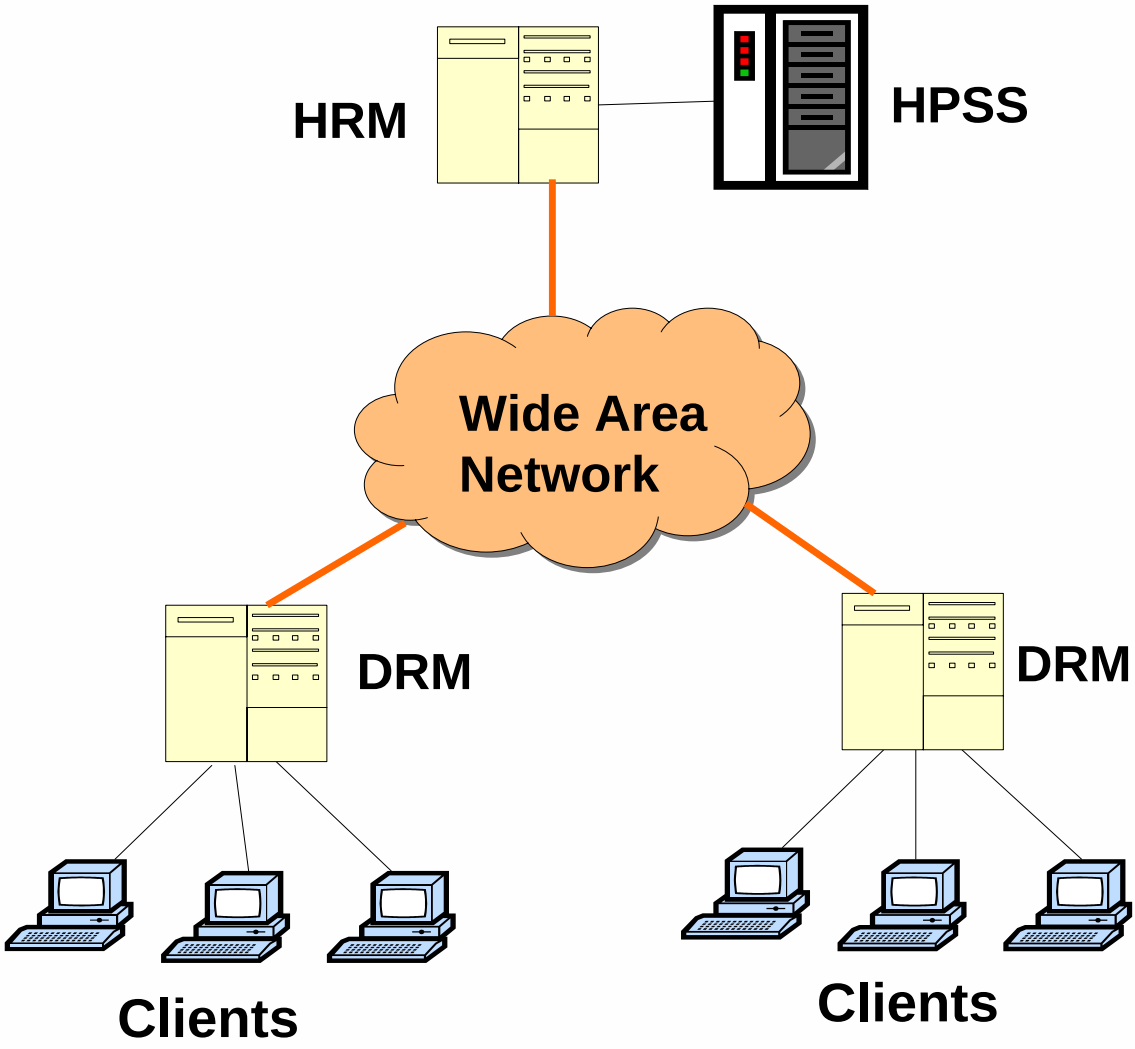
Song Jiang and Xiaodong Zhang

College of William and Mary, USA

Why Disk Caching in Data Grids?

- It takes a long latency (up to several minutes) to load these data at their remote sources at the mass storage system (MSS) ;
- It takes a very long time (up to a few hours) to complete file transfers for a request over wide-area networks;
- A researcher's workstation or even her local computer center may not be able to keep all the required dataset for a long time for his needs.

Disk Caching in Storage Resource Managers (SRM)



HPSS: High Performance Storage System

DRM: Disk Resource Manager

HRM: Hierarchical Resource Manager

A general Utility function to evaluate the value of a file:

For a file i at time t, $\phi_i(t) = L_i(t) * \frac{C_i}{S_i}$

$L_i(t)$ denotes its locality strength,

S_i denotes the file size,

C_i denotes its retrieving cost

Locality estimation of files is the most critical factor determining hit ratio of disk caching

Drawbacks in the existing Locality Estimation Methods:

(1) **Recency Based**

- Locality of large file access in Data Grids is weaker than that of I/O block and web file caching
- Hard to deal with weak locality file requests;
- Example: Greedy Dual-Size (GDS)

(2) **Frequency Based**

- Pollution problem
- Examples: Greedy Dual-Size with Frequency (GDSF), Hybrid, Lowest relative Value (LRV)

(3) **Re-use Density Based**

- Overcome the drawbacks of previous methods;
- Could be irrelevant to locality because Density is computed over wall clock time;
- Examples: Least Cost Beneficial Based on the K backward References (LCB-K)

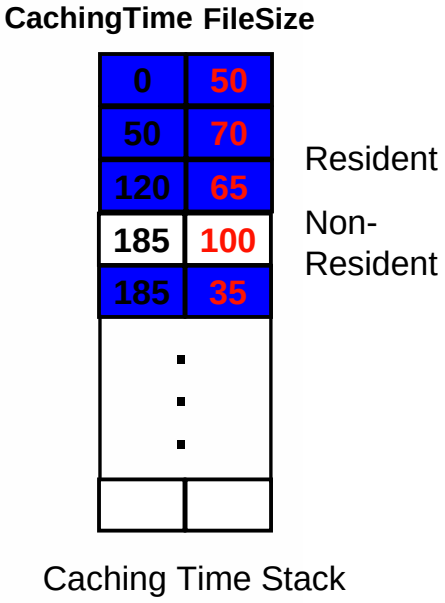
2003 IEEE International Conference on CLUSTER COMPUTING
HONG KONG

Our Principle on Disk Replacement

- Only relevant history information is used in locality estimation: the order of file requests, and cache size demands
- Total cache size is used with the locality estimation to answer the question: Does a file have enough locality so that it if it is cached it could be hit for its next reference before it is replaced from the cache with its specific size ?

Our Utility Function $\phi_i(t) = \frac{1}{CachingTim\ e_i} * \frac{C_i}{S_i}$

Caching Time of a file describes how much cache is consumed since the last reference to the file. A caching time is maintained for a file for certain period of time even it is replaced, so that the utility of its next access can be more accurately assessed.



Caching Time Stack

Least Value Based on Caching Time (LVCT)

(1) If the access to file ***f*** is a hit, update the caching time stack.

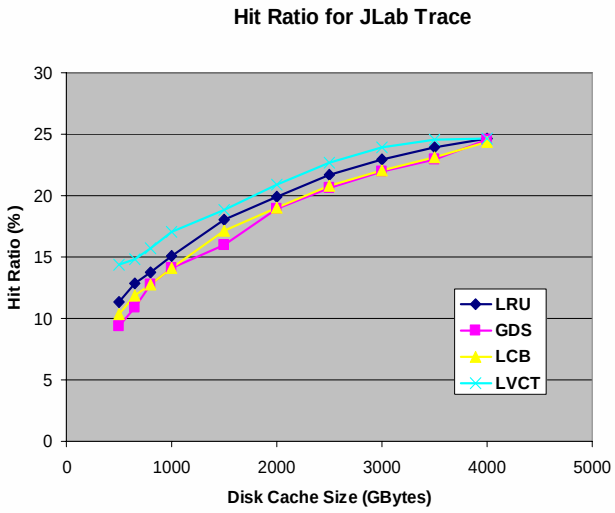
(2) If the access to file ***f*** is a miss:

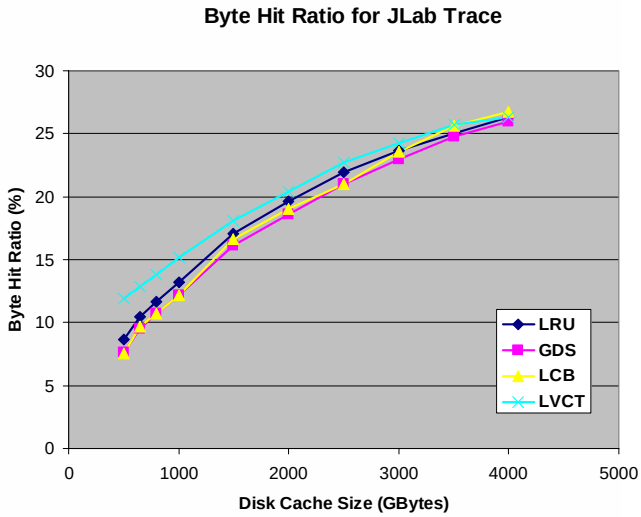
- Select a set of resident files whose utility values are smallest among resident files;
- If the utility value of file ***f*** is smaller than any of the files in the set, ***f*** is not cached.
- Otherwise, ***f*** is not cached
- The caching time stack is updated accordingly.

Trace Description

- Collected in a MSS system, JASMine, at Jefferson's National Accelerator Facility (JLab),
- Represent the file access activities for a period of about 6 months.
- 207,331 files accessed in the trace, and their total size is 144.9 TBytes.

Simulation Results





LVCT Advantage

- Replace files with large caching times timely because they are less likely to generate hits even if cached;
- Cache space is saved to serve small caching time files;
- Improvement is more apparent with small cache sizes.

Main Results

- Disk caching in data grids exhibit properties different from transitional I/O buffering and Web file caching;
- We identify a critical drawback in abstracting locality information from Grid request streams, i.e. counting on misleading time measurements.
- We propose a new locality estimator using more relevant access events;
- The real-life workload traces demonstrate the effectiveness of our replacement.