Adaptive File Caching in Distributed Systems

Ekow Otoo, Frank Olken and Arie Shoshani

Scientific Data Management Group

Lawrence Berkeley National Laboratory

Berkeley, CA 94720

Outline

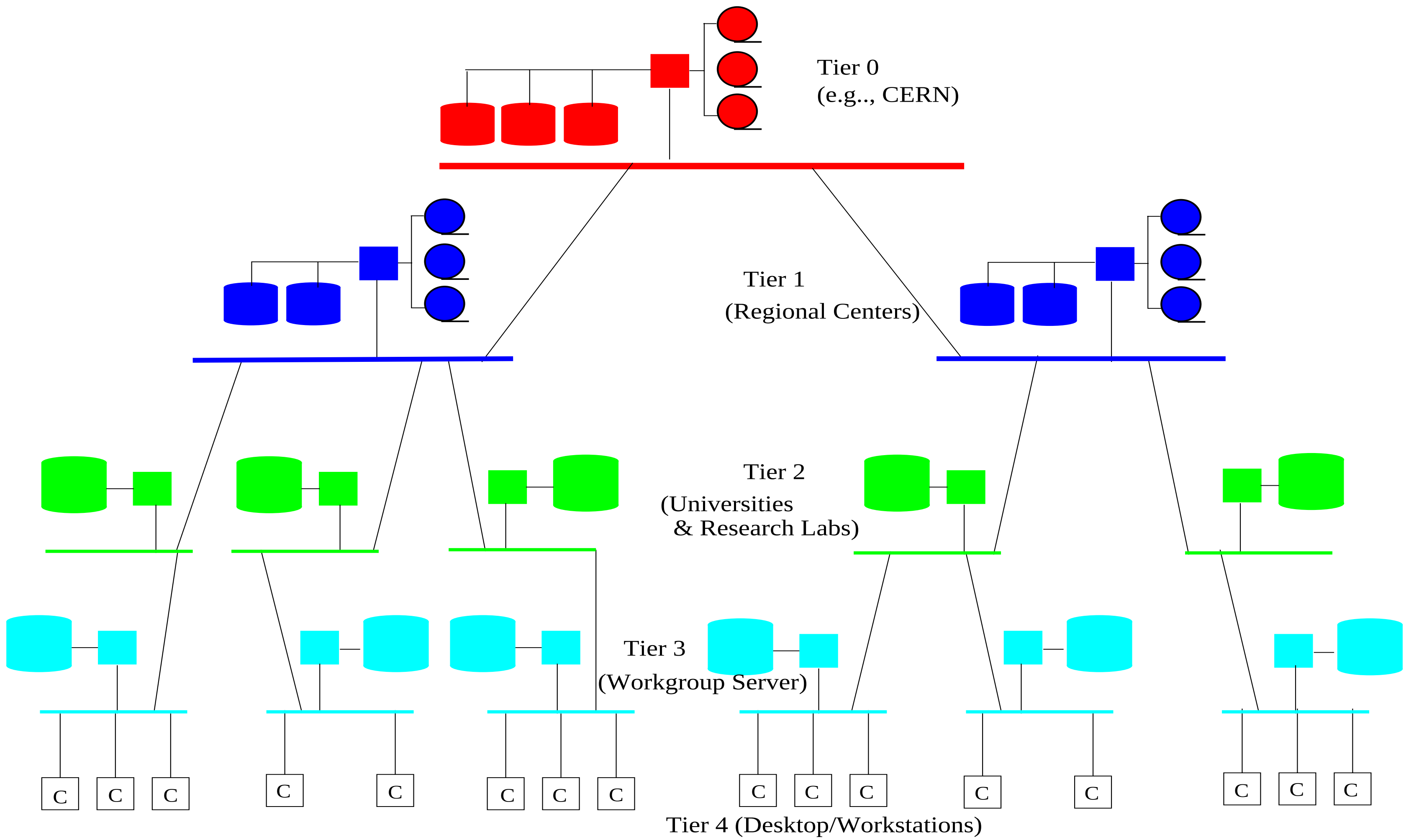
- Problem Statement
- Storage Resource Manager (SRM)
- Methodology and Approach
- Current Focus of Work
- Disk Cache Replacement Policies
- Computing an Optimal Replacement Policy
- File Admission Policies
- A Simulation Model of an HRM
- Future Work

Problem Statement

Environment

- Storage and computational resources (nodes) interconnected by a high-speed network.
- The storage nodes are organized into multi-level tiers.
- The granularity of data access is a file.
- Files may be replicated, in some controlled manner, onto more than one storage node
  - A replica may be registered in a replica catalogue - *globally known.*

Network Computing Model



Storage Resource Manager (SRM)

- SRMs manage and coordinate file requests and transfers from and into storage resources.
- Files may be *permanent or volatile*
- Clients request files from local SRMs.
- An SRM, in consultation with a replica catalogue, coordinates caching and transfers from other SRMs.

  DRM: An SRM that manages a disk resource.

  HRM: An SRM that manages its own disk resource
  + migrating and staging files to and from tape resources.
- Tape access is done through a hierarchical storage system (HSS) such as HPSS.

Objectives

- To determine a service policy for an SRM that achieves one or more of the following objectives:
  - Minimizes response times of file accesses by jobs
  - Maximizes throughput of storage resources
  - Maximize the hit ratio of storage caches
- Subject to the following constraints:
  - Bounded storage capacities at the nodes
  - Bounded network bandwidth capacity
  - File migration policies between tiers
  - Limited number of files per job at any one time

Methodology and Approach

- Study increasingly complex variants of the problem, starting with a single tier model.
- Explore analytical and simulation studies

Some Definitions:

$C_{HSS}(i) =$ Cost to retrieve file i from HSS

$p_{d_{HRM}} =$ Prob. file on HRM-Disk; $\ell_d =$ Disk latency;

$\beta_d =$ Disk Trans. Rate; $\ell_t =$ Tape Latency;

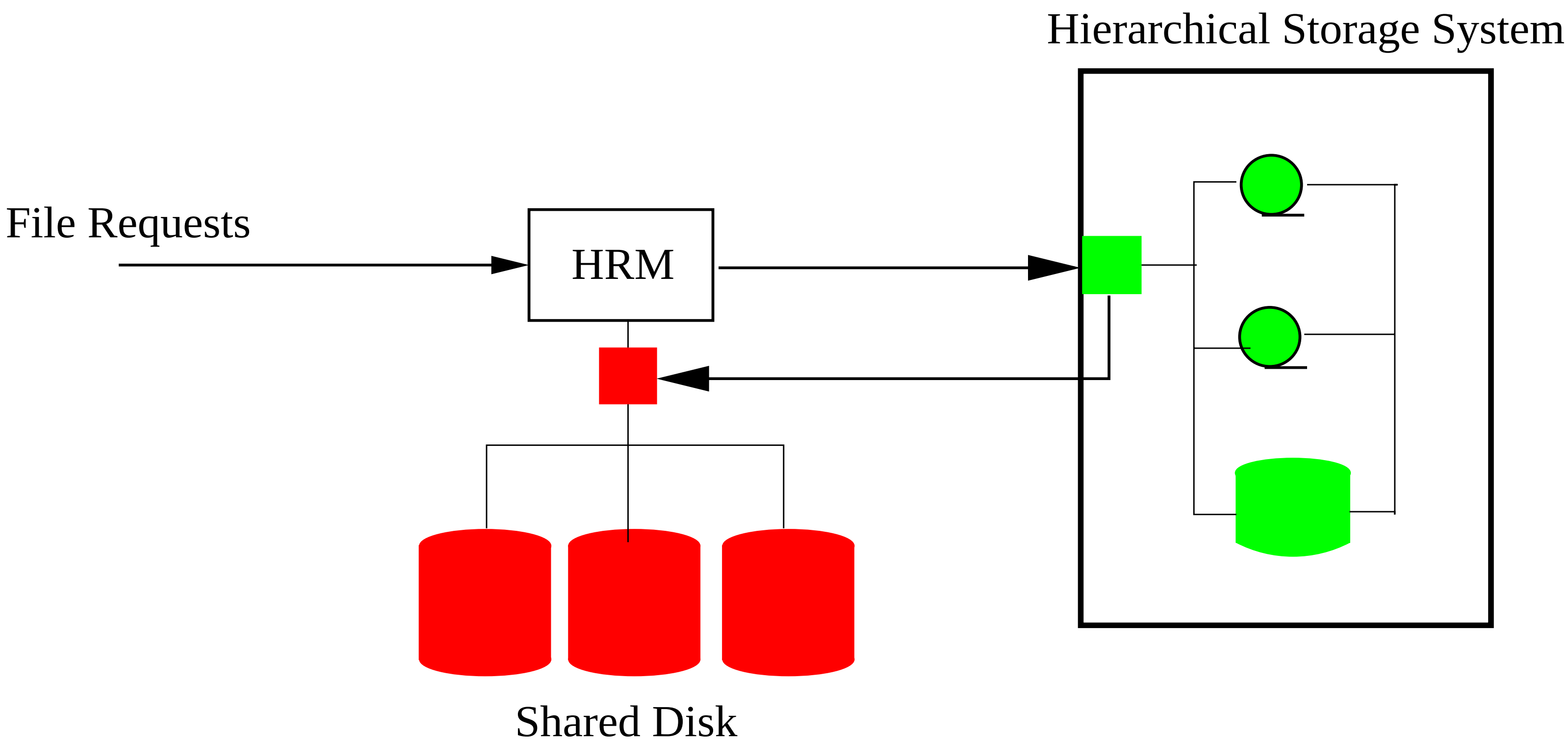
$\beta_t =$ Tape Trans. Rate; $s_i =$ File Size.

$p_{d_{HSS}} =$ Prob. that file is on HSS-Disk

$\ell_{d_{HSS}} =$ HSS-Disk latency

Variants of the Problem

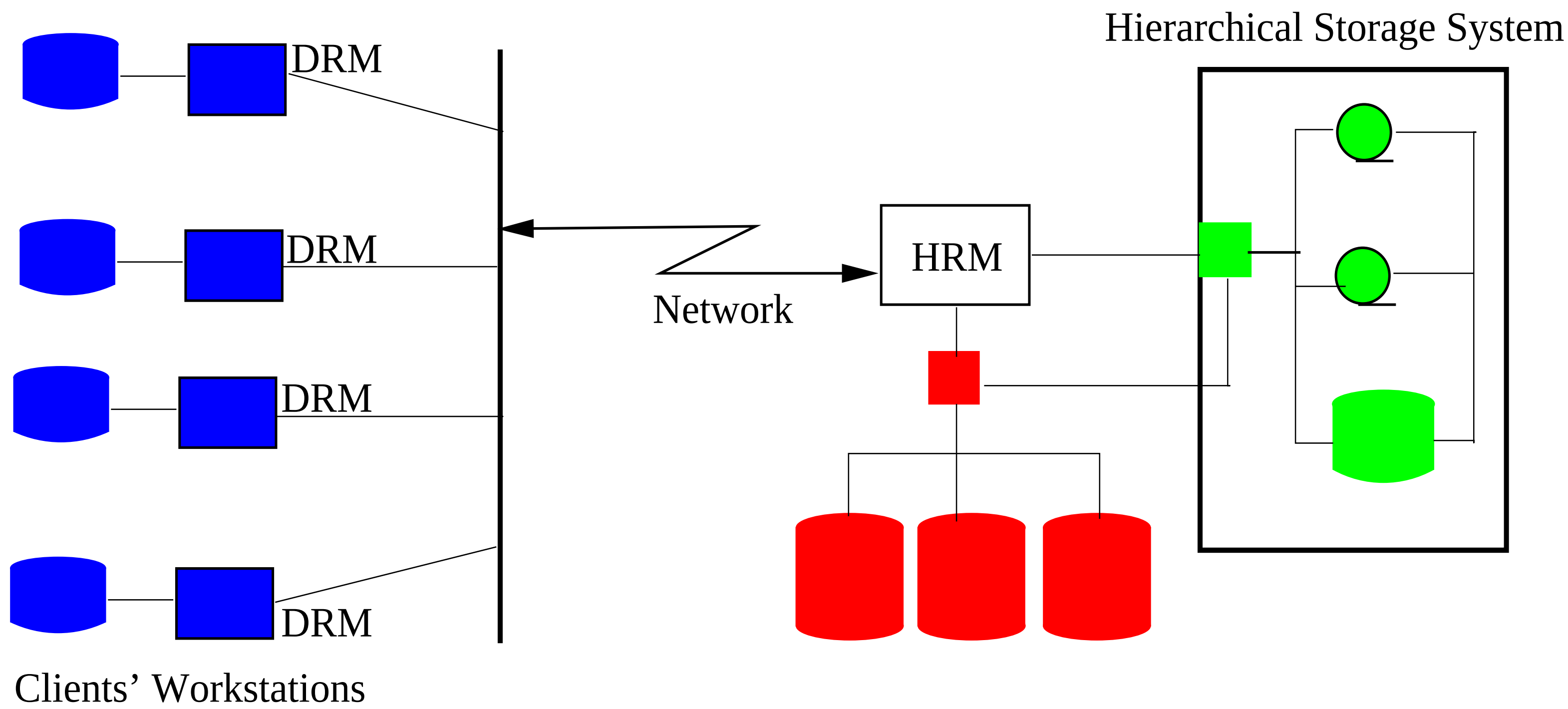
CASE 1: Clients Submit Requests to an HRM



$$C_{HRM}(i) = \ell_d + \beta_d.s_i + (1 - p_{d_{HRM}}).(C_{HSS}(i))$$
$$C_{HSS}(i) = \ell_{d_{HSS}} + \beta_{d_{HSS}}.s_i + (1 - p_{d_{HSS}}).(\ell_t + \beta_t.s_i)$$

Requests to a DRM Accessing Remote HRM

CASE 2: Multiple Clients with Independent Disks Accessing Remote HSS



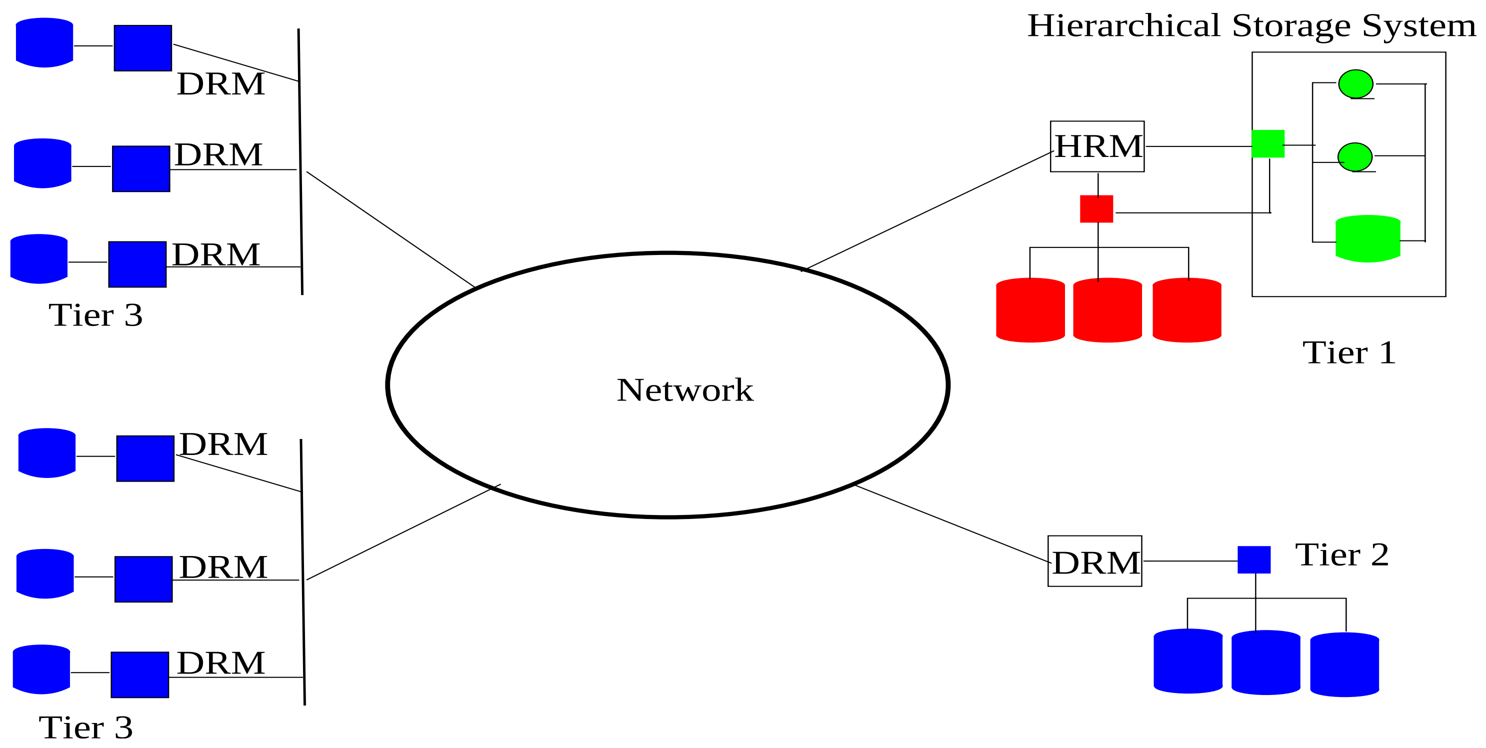
$$C_{Remote}(i) = \ell_{net} + \beta_{net}.s_i + (1 - p_{d_{DRM}})C_{HRM}(i)$$

ℓ_{net} = Network Latency; β_{net} = Network Trans. Rate;

$p_{d_{DRM}}$ = Prob. that file is on local DRM disk.

Requests to DRM Accessing Remote DRMs & HRMs

CASE 3: Multiple Clients with Independent Disks Accessing Multiple Remote HSS and Other DRM



Current Focus

- Client submit requests to an HRM.
- Each request is for a set of files in any order.
- The requested files may be either resident on disk or may still be in a tape library.
- There is a limited number of files accessible by a job at any one time.
- File requests are queued for subsequent processing if an SRM is busy.
- Two Immediate Problems to Address
  - What files to read from the HSS - *File Admission Policy.*
  - What files to evict from HRM's disk cache - *Cache Replacement Policy.*

Disk Cache Replacement Policies

Some Previous Known Results

(1) Optimal Policy from Paging Analogy: (If future references are known). *Belady - 1966; Aho, Denning and Ullman - 1971*
Replace files that would be requested furthest in the future.

- Not very realistic since future reference pattern is not known.
- It is based on fixed size files.
- Computing an optimal policy for replacement is expensive.
- Approximation algorithms used. Example is LRU with clock algorithm.

Disk Cache Replacement Policies, Cont.

(2) File Analogue of the LRU-K Algorithm:

(Using past reference pattern) *O'Neil et al. 1993.*

Drop a file i from the cache where i is the file with the maximum backward K-distance $b_\tau(i, K)$.

Definition 1.1 (Backward K-distance $b_\tau(i, K)$) *Given a reference string, $\sigma = r_1, r_2, \ldots, r_\tau$, up to a time τ, the backward K-distance $b_\tau(i, K)$, is the distance backward to the K^{th} most recent reference to the file i.*

$$b_\tau(i, K) = \begin{cases} x, & \textit{if } r_\tau \textit{ has the value } i, \\ \infty, & \textit{if } i \textit{ does not appear at least } K \textit{ times in } \sigma. \end{cases}$$

Disk Cache Replacement Policy, Cont.

(3) Hazard Rate Function (HOPT):

(Reference distribution known) *Olken, 1983.*

Based on variable space policy with fixed rental charges.

- Let $C_{HSS}(i)$ be the cost of accessing file i of size s_i from an HSS and let $h_i(t) = f_i(t)/(1 - F_i(t)$ be a hazard rate for file i.
- ***The replacement policy calls for evicting the file i with the minimum value of*** $\gamma_i(t)$ ***where***

$$\gamma_i(t) = \frac{h_i(t) * C_{HSS}(i)}{s_i}.$$

Disk Cache Replacement Policy, Cont.

(4) Profit Function:

(Average reference rate known) *Scheuermann et al., 1996*

- For a file i, let λ_i be the average rate of reference over the last τ time period. Let s_i be the size of the file and let C_i the cost of retrieving the file from HSS onto the HRM's disk.
- ***The victim for replacement is the file with minimum estimated profit*** $\pi_i(\tau)$ ***where***

$$\pi_i(\tau) = \frac{\lambda_i * C_i}{s_i}$$

- This is equivalent to HOPT for Poisson arrivals with known parameters λ_i for each file i.

Our Approach for Disk Cache Replacement

1. Use HOPT, but we need a method to estimate hazard rate $h_i(t)$.
2. To estimate $h_i(t)$, we adopt the technique of LRU-K.

Some Details:

- Requests for multiple files are queued thereby allowing for a tentative schedule of admitting the files to be processed.
- Using *hazard rate function*, we have

$$\gamma_i(t) = \frac{h_i(t) * C_{HSS}}{s_i}$$

where $C_{HSS} = (\ell_{d_t} + (1 - p_{d_t}).\ell_t) + (\beta_d + (1 - p_{d_t})\beta_t)s_i$.

- The main problem is in evaluating $h_i(t)$ for each file i on disk.

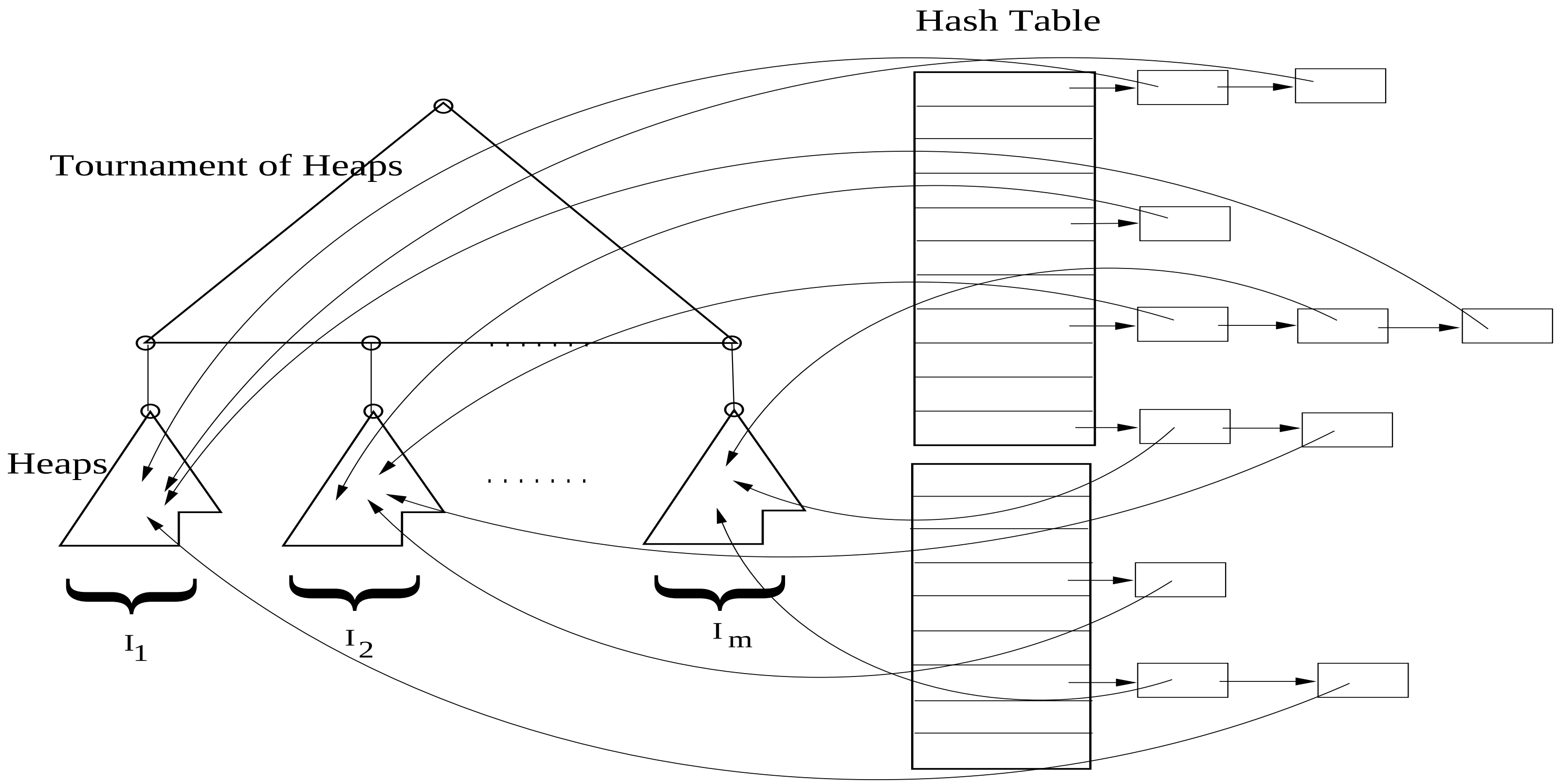
Disk Cache Replacement, Cont.

- If file request arrival is *Poisson*, i.e, inter-reference time of a file i, is exponentially distributed as $F_i(t) = 1 - e^{-\lambda_i t}$, then $h_i(t) = \lambda_i$.
- $h_i(t)$ is approximated by using the last K inter-reference times.
- We can compute the $\gamma_i(t)$ as

$$\gamma_i(t) = \frac{K}{t_{current} - t_K} * \frac{C_{HSS}}{s_i}$$

- This is equivalent to HOPT for Poisson arrivals using LRU_K to estimate rate λ_i.

Computing $\gamma_i(t)$



Computing $\gamma_i(t)$, Cont.

- Assuming that we have N files in a disk cache.
- Let the number of independent heaps be $\ln N$.
- The update cost of a heap after each file reference is $O(\ln N)$.
- The replacement cost of a file is also $O(\ln N)$.

Disk File Admission Policy

- We require not only *optimizing objective functions* but also *fairness.*
- *FIFO:* Select a file of the first arrival request that is already cached. Otherwise select one with maximum pending requests.
- *Round Robin:* Select a file cyclically from pending requests. From each request, choose the file that is already cached. Failing that choose one with maximum other pending requests.
- *Priority Assignment function:* Define $\varphi(i)$ as

$$\varphi(i) = C_0\beta(i) + C_1\kappa + C_3(Time_{current} - Time_{arrival})$$

for some appropriate values of C_0, C_1, C_3.

Disk File Admission Policy, Cont.

- The parameters of the priority assignment function are:

  $\beta(i)$**:** A 0/1 value indicating whether the file is cached or not.

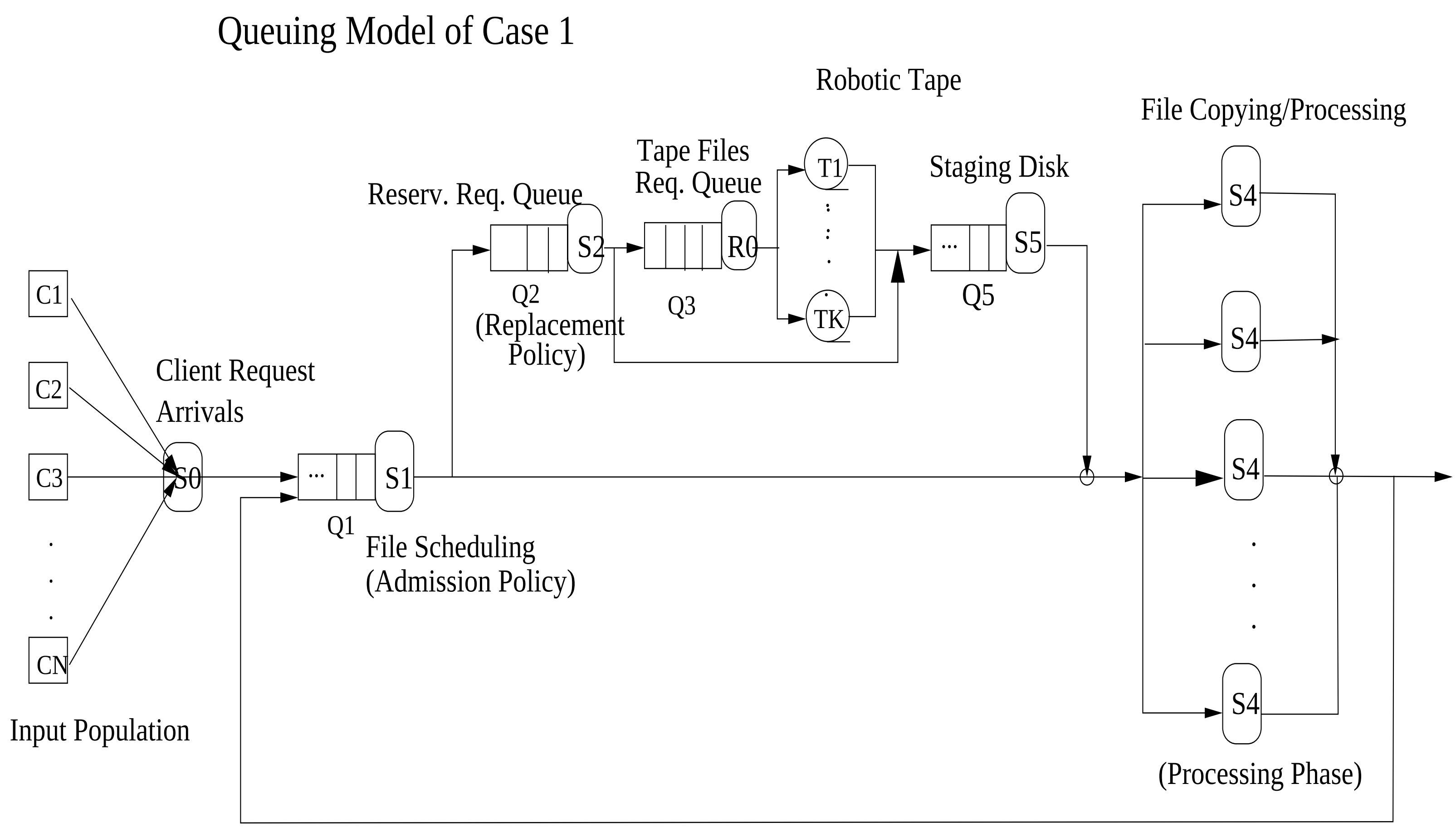
  κ**:** The count of the number of pending requests for file i.

  $Time_{current}$**:** The current time.

  $Time_{arrival}$**:** The earliest time since file i was requested.

- ***These approaches, in combination with the replacement policies, are being studied using discrete event simulation.***

A Simulation Model of Case 1



Future Work

- Embed results into works on SRMs being developed (DRM and HRMs)
- Extend our study to include multi-tier file requests and multi-tier file access policies
- Explore the use of user level hints in file caching and replacement policies. For example
  - Requests for files may be in any order or requested in some specific order.
  - File requests may be identified as being correlated.
- Use workloads from scientific experiments to drive our simulation.