Broadcast Disks

Outline of Presentation

- Introduction and Motivation
- Broadcast Disk for Non-Real Time Computing
  - Server Side
  - Client Side

Motivation

- Assymetric communication environments -
  - uplink costs higher than downlink.
  - client power may be insufficient to send query to server.
  - More powerful server transmission may be received by client.
- Certain types of data lend themselves naturally to *dissemination by broadcast* paradigm
  - Popular data e.g., weather, traffic information.
  - Data whose value and content may change dynamically e.g., stock prices.
- Broadcast Disks provides a scalable solution.
- Conceptually act as intermediate memory hierarchy between client and server.

Broadcast Disk Issues

- Client access latency.
- Client Power Consumption.

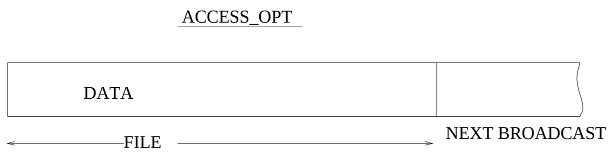
Solutions:

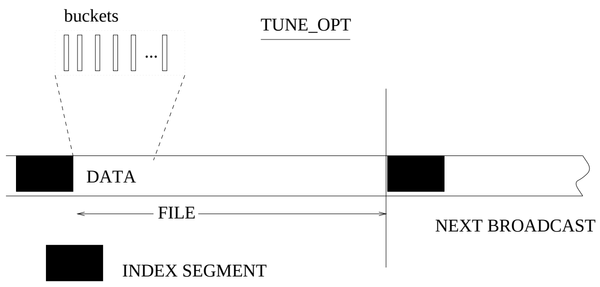
- Server-Side: Broadcast Disk Organization.
- Client-Side: Caching and Prefetching techniques.

Broadcast Disk - Indexing (Imilienski et. al.)

- Assume a single *flat* disk structure.
- Broadcast data can vary in content and size. Updates made inbetween rounds.
- Clients listen to the broadcast and fetch individual records (identified by a key)
- Need to minimize client *access latency* and power.
- *Tuning time* decides power consumption.
- Doze mode consumes significantly less power than active mode (50 μ W vs. 250 mW).
- How to organize the broadcast to minimize both ?
  - Approach: Interleave data with index. Data and index transmitted in equal-sized chunks called buckets.
  - Client switches to active mode to get its data.
  - Client exploits index information to save power by going into doze mode.
  1. Base Case 1: *Access_opt* produces large tuning time.
  2. Base Case 2: *Tune_opt* creates large access latencies.
  3. (1,m) Indexing.
  4. Distributed Indexing.

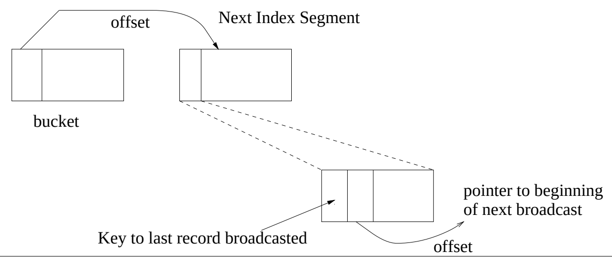
Broadcast Disk - Indexing

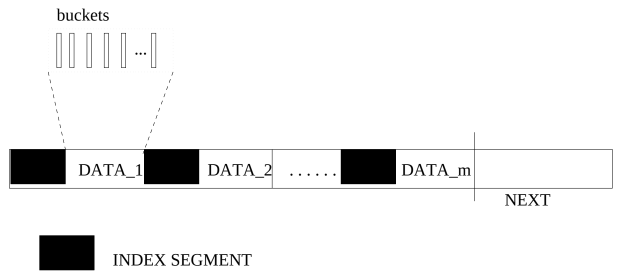




Broadcast Disk - 1,m Indexing

entire index is broadcast m times during a single broadcast of the data.





Broadcast Disk - Distributed Indexing

Idea : vary the degree of replication of the index within a single broadcast.

1. Nonreplicated
2. Entire Path Replicated
3. Partial Path Replicated

Indexing in Broadcast Disks

Comparisons between the different Approaches

- Tuning Time: $Tuning\_opt \approx (1:m) \approx Distributed \ll Access\_opt$
- Access Time: $Access\_opt < Distributed < (1:m) \ll Tuning\_opt$

  Example Consider AT&T Hobbit chip - 50μW in *doze mode* and $250mW$ in active mode, bucket size 128 bytes, data size 1250 buckets, index size 53 buckets, no. of levels in tree = 3, channel BW 10 Kbps.
  Then,

Algorithm	Access Time (buckets)	Tuning Time(buckets)	Power (Joules)
Access_opt	625	625	15.625
Tuning_opt	1303	4	0.106
(1,m) I.	909	5	0.130
Distributed I.	689	6	0.153

Broadcast Disk (Zdonik et al.)

- Limited BW - how to broadcast data of varying importance and popularity.
- Propose multiple Broadcast Disks of different sizes and different rates.
- Hot items on faster, cold ones on slower disks.
- Question: How to interleave the broadcasts from the different disks over the single channel ?
  - Fixed interarrival times for any data.
  - Periodicity property for the entire broadcast helpful for updates to content as well as structure.

Client Side in Broadcast Disks (Zdonik et al.)

- Caching and Prefetching useful for reducing client mismatch to broadcast profile.
- Caching: Purging needs to account for both access probability and broadcast frequency
  1. $PIX = \frac{\text{prob. of access}}{frequency}$.
  2. LIX - A version of PIX using LRU for each disk.
- Prefetching
  1. Latency based
  2. Tag-team
- Read/Write Case
- Security issues not addressed

Broadcast Disk Organization for a Real Time Setting

Issues

- Timing
- Fault Tolerance