Κινητός και Διάχυτος Υπολογισμός (Mobile & Pervasive Computing)

Δημήτριος Κατσαρός

Διάλεξη 7η

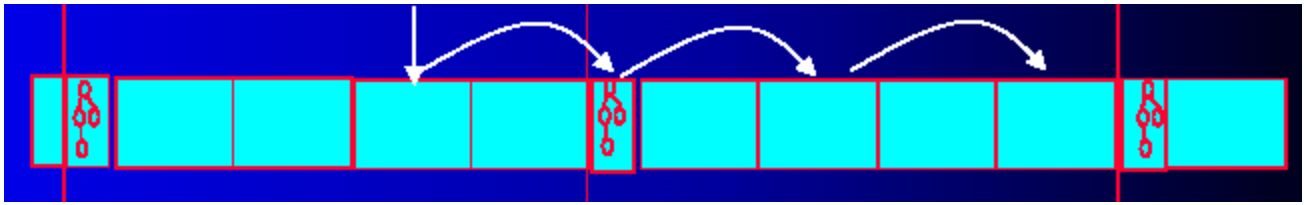
Περιεχόμενα

- **Ευρετήρια για ομοιόμορφα πρότυπα προσπέλασης**
  - **Δενδρικά (single rooted): (1,m)-indexing**
  - **“Distributed” (multi-rooted): Exponential index**

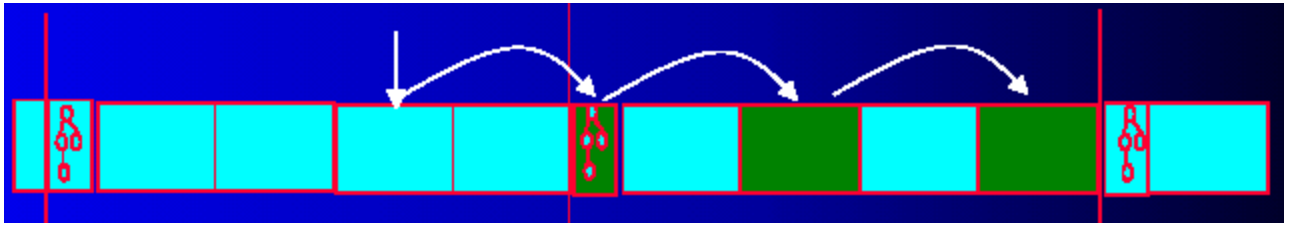
Παράμετροι ενδιαφέροντος (1/2)

- **Tuning time**: Ο χρόνος που ο κινητός πελάτης δαπανά "ακούγοντας" το κανάλι. Προσδιορίζει την κατανάλωση ενέργειας για την απόκτηση των δεδομένων
- **Latency (Access time)**: Ο χρόνος που περνάει (κατά μέσο όρο) από τη στιγμή που ο κινητός πελάτης "κάνει αίτηση" για κάποια δεδομένα μέχρι τη στιγμή που τα δεδομένα αυτά έρχονται στην κατοχή του πελάτη
  - **Probe wait**: Ο μέσος χρόνος από τη στιγμή συντονισμού στο κανάλι μέχρι να βρει τον δείκτη (pointer) για τον επόμενο index. Είναι ίσος με το μισό της απόστασης μεταξύ δυο τμημάτων index.
  - **Bcast wait**: Ο μέσος χρόνος από τη στιγμή που βρίσκεται ο πρώτος index μέχρι να "κατεβούν" όλα τα δεδομένα

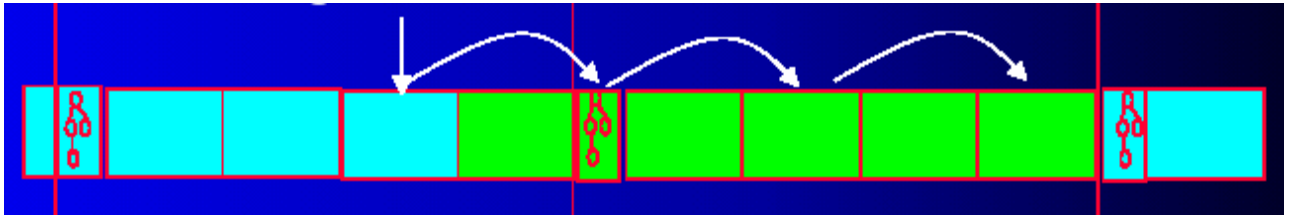
Παράμετροι ενδιαφέροντος (2/2)



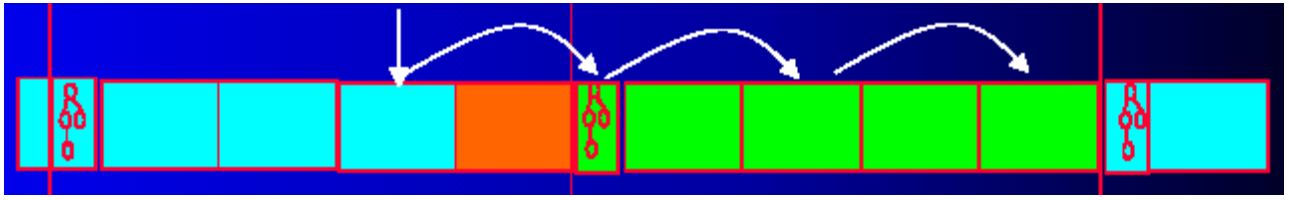
Querying



Tuning time



Latency



Probe wait

Bcast wait

Οργάνωση του καναλιού εκπομπής

- **Packet**: η βασική (μικρότερη) μονάδα μεταφοράς μηνυμάτων στα δίκτυα
- **Bucket**: η μικρότερη λογική μονάδα εκπομπής. Αποτελείται από σταθερό αριθμό packets. Όλα τα buckets έχουν το ίδιο μέγεθος
  - Index buckets
  - Data buckets
- **Index Segment**: σύνολο συνεχόμενων index buckets
- **Data Segment**: σύνολο συνεχόμενων data buckets

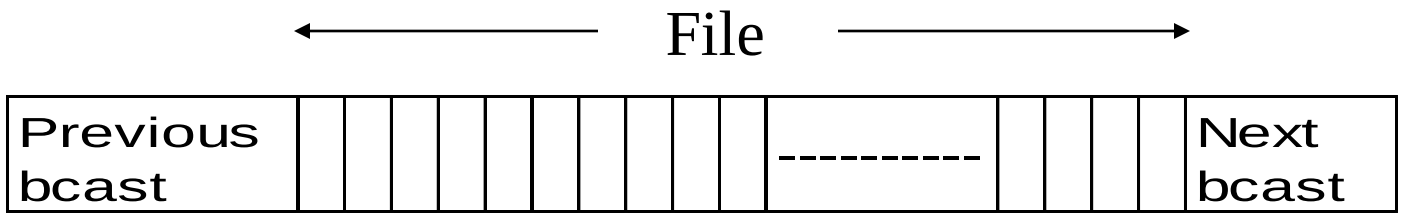
Οργάνωση του καναλιού εκπομπής

- Περιεχόμενα bucket
  - **Bucket_id**: το offset του bucket από την αρχή του κύκλου εκπομπής
  - **Bcast_pointer**: το offset μέχρι την αρχή του επόμενου κύκλου εκπομπής
  - **Index_pointer**: το offset μέχρι την αρχή του επόμενου index segment
  - **Bucket_type**: data bucket ή index bucket
- Index bucket: είναι μια ακολουθία της μορφής:
  - (**attibute_value, offset**): offset είναι ένας δείκτης στο bucket που περιέχει εγγραφή που προσδιορίζεται από την attribute_value

Clustering index

- **Clustering index**: ένα ευρετήριο (index) είναι clustered πάνω σε ένα attribute, εάν όλες οι εγγραφές με την ίδια τιμή για το attribute αυτό, εμφανίζονται συνεχόμενες σε ένα “αρχείο”
- Τα δυο άκρα στην βελτιστοποίηση Tuning time και Acceess time
  - Latency_opt
  - Tune_opt

Latency OPT

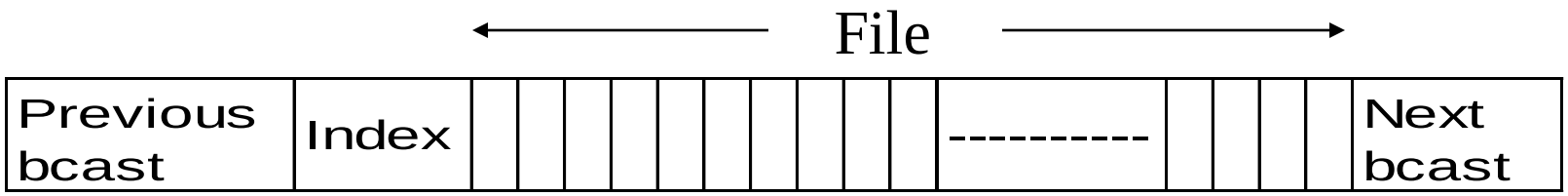


Latency είναι η βέλτιστη: Δεν υπάρχει επιβάρυνση για το index

Latency = Data/2 + C

Tuning time = Data/2 + C

Tuning OPT



Tuning time είναι ο βέλτιστος:

Latency = (Data + Index) / 2 + (Data + Index) / 2 + C

= Data + Index + C

Tuning time = k + C

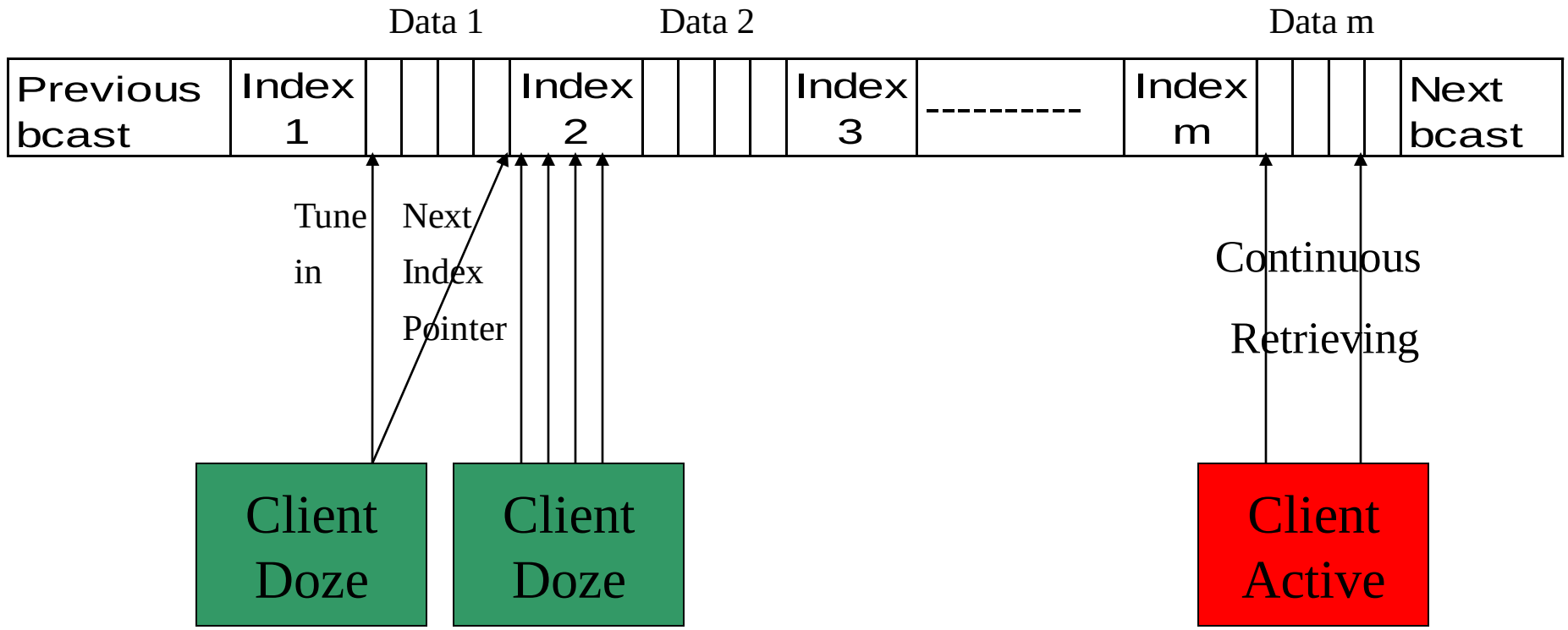
k: number of levels in the index tree

(1, m) Indexing

Το ευρετήριο (index) εκπεμπεται m φορές κατά τη διάρκεια μιας εκπομπής του αρχείου.

Πολυεπίπεδο ευρετήριο (index), δηλαδή δένδρο

Ολόκληρος ο index εκπέμπεται πριν από το 1/m κομμάτι του αρχείου.



Ανάλυση (1, m) Indexing

- Η κατανομή πιθανότητας του initial probe για τους πελάτες είναι ομοιόμορφη
- Data: το μέσο μέγεθος του αρχείου
- C: η coarseness του index attribute
- Ο index έχει δείκτες μόνο στην πρώτη εμφάνιση εγγραφής με συγκεκριμένη τιμή για το attribute
- Άρα, κατασκευάζουμε ευρετήριο μόνο για (Data/C) data buckets
- n, η χωρητικότητα ενός bucket, δηλ., ο αριθμός των ζευγών (attribute_value,offset) που μπορεί να στεγάσει
- k: ο αριθμός των επιπέδων του ευρετηρίου
- Index: ο αριθμός των buckets του

Ανάλυση (1, m) Indexing

Όταν το δένδρο είναι πλήρως ισοζυγισμένο

$$k = \left\lceil \log_n \left(\frac{Data}{C} \right) \right\rceil \qquad Index = \sum_{i=0}^{k-1} n^i$$

Latency: The *probe wait* is $\frac{1}{2} * (Index + \frac{Data}{m})$ and the *bcast wait* is $\frac{1}{2} * ((m * Index) + Data) + C$. Hence, the latency is

$$\frac{1}{2} * \left(Index + \frac{Data}{m} \right) + \frac{1}{2} * ((m * Index) + Data) + C,$$

i.e.,

$$\frac{1}{2} * \left((m+1) * Index + \left(\frac{1}{m} + 1 \right) * Data \right) + C.$$

Ανάλυση (1, m) Indexing

Tuning time: $1 + k + C$

- **Βέλτιστο m για ελαχιστοποίηση Latency:**
  - Παραγώγιση την εξίσωση της Latency ως προς m
  - Εξίσωση με 0
  - Επίλυση ως προς m

$$m^* = \sqrt{\frac{Data}{Index}}$$

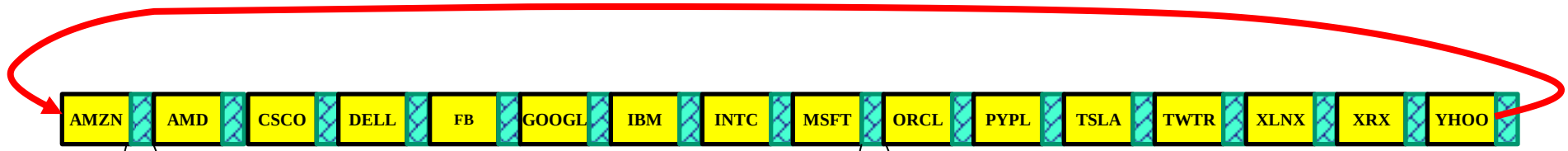
Περιεχόμενα

- **Ευρετήρια για ομοιόμορφα πρότυπα προσπέλασης**
  - **Δενδρικά (single rooted): (1,m)-indexing**
  - **"Distributed" (multi-rooted): Exponential index**

Exponential Index (1/3)

- Data buckets
  - Data part
  - Index table
    - Index entries:
      - κάθε entry indexes ένα segment από buckets και έχει τη μορφή **{distInt, maxKey}**
        - » distInt: καθορίζει την απόσταση των buckets από το τρέχον bucket (μετρημένο σε αριθμό buckets)
        - » maxKey: είναι η τιμή του μέγιστου κλειδιού αυτών των buckets
      - Τα μεγέθη των segments αυξάνουν εκθετικά (δυνάμεις του 2). Η i-οστή entry περιγράφει το segment των buckets τα οποία είναι σε απόσταση 2^{i-1} έως $2^{i}-1$
      - Οι τιμές distInt δεν χρειάζεται να καταγραφούν, αφού μπορούν εύκολα να συναχθούν (παρουσιάζονται στο παράδειγμα για λόγους κατανόησης)

Exponential Index (2/3)

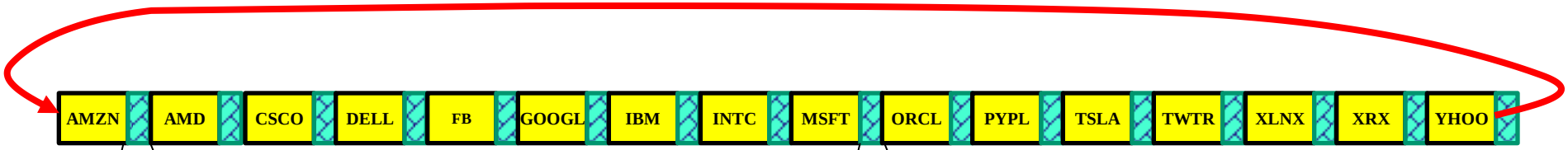


distInt	maxKey
1-1 bucket	AMD
2-3 bucket	DELL
4-7 bucket	INTC
8-15 bucket	YHOO
minKey=AMD, maxKey=YHOO	

distInt	maxKey
1-1 bucket	ORCL
2-3 bucket	TSLA
4-7 bucket	YHOO
8-15 bucket	INTC
minKey=AMD, maxKey=YHOO	

Exponential Index (3/3)

ΥΠΟΘΕΣΗ: Συντονισμός στο: **AMZN** Αναζήτηση του: **ORCL**



distInt	maxKey
1-1 bucket	AMD
2-3 bucket	DELL
4-7 bucket	INTC
8-15 bucket	YHOO
minKey=AMD, maxKey=YHOO	

Έστω: Συντονισμός στο **AMZN**

- Έλεγχος εάν βρήκα ό,τι αναζητώ. Εάν επιτυχία, ΤΕΛΟΣ
- Αλλιώς: BinSearch στα keys (Το αναζητούμενο είναι **INTC** < **ORCL** < **YHOO**)
- Δηλαδή, πρέπει να γίνει συντονισμός ξανά αμέσως μετά το **INTC**
- Άρα, αλλαγή σε doze mode για 7 buckets

distInt	maxKey
1-1 bucket	ORCL
2-3 bucket	TSLA
4-7 bucket	YHOO
8-15 bucket	INTC
minKey=AMD, maxKey=YHOO	

- Wake up στο **MSFT**
- Έλεγχος εάν βρήκα ό,τι αναζητώ. Εάν επιτυχία, ΤΕΛΟΣ
- Αλλιώς: BinSearch στα keys (Το αναζητούμενο **ORCL** υπάρχει στα κλειδιά του index)
- Δηλαδή, πρέπει να γίνει συντονισμός ξανά στο επόμενο (ΕΠΙΤΥΧΗΣ αναζήτηση)

ΣΥΝΟΛΙΚΑ:

- Tuning time= 3 (AMZ, MSFT, ORCL)
- Access time= 10 (AMZ→ORCL)