HY463 - Συστήματα Ανάκτησης Πληροφοριών
Information Retrieval (IR) Systems

Parallel and Distributed IR
Παράλληλη και Κατανεμημένη ΑΠ

Γιάννης Τζίτζικας

Ενότητα: 17
Ημερομηνία :

Διάρθρωση Περιεχομένου

Μέρος Α: Παράλληλη Ανάκτηση Πληροφοριών (Parallel IR)

Μέρος Β: Κατανεμημένη Ανάκτηση Πληροφοριών (Distributed IR)
- Επιλογή Πηγής
- Ενοποίηση Αποτελεσμάτων

Μέρος Γ: Ανάκτηση Πληροφοριών σε Ομότιμα Συστήματα (Peer-to-Peer Systems)

Μέρος Α

Παράλληλη Ανάκτηση Πληροφοριών

Παράλληλη Ανάκτηση Πληροφοριών: Διάρθρωση

- Κίνητρο
- Μέτρα Απόδοσης Παράλληλων Προγραμμάτων
- Παράλληλη Επεξεργασία και Ανάκτηση Πληροφοριών
  - Parallel Multitasking
  - Partitioned Parallel Processing
- Διαμερισμός Εγγράφων (για MIMD αρχιτεκτονική)
- Διαμερισμός Όρων (για MIMD αρχιτεκτονική)

Κίνητρο

- Όσο πιο μεγάλη είναι μια συλλογή κειμένων, τόσο πιο ακριβή γίνεται η διαχείρισή της από ένα ΣΑΠ
- Ανάγκη για αρχιτεκτονικές και τεχνικές για **βελτίωση της απόδοσης**
  - The volume of electronic text available online today is staggering.
  - The WWW contains over 30 billions pages of text.

Παράλληλος Προγραμματισμός

- Παράλληλος Προγραμματισμός: Η ταυτόχρονη χρήση πολλών επεξεργαστών για την επίλυση ενός προβλήματος
- Ταξινομία αρχιτεκτονικών (κατά Flynn):
  - SISD single instruction, single data
  - SIMD single instruction, multiple data
    - N processors running the same program on different parts of the data, e.g. Thinking machine
  - MISD multiple instruction, single data
    - N processors running different programs on a single data stream in shared memory
  - MIMD multiple instruction, multiple data
    - N processors, N instruction streams, N data streams
    - the most common architecture. It also captures ***distributed*** computing architectures
      - the main difference between MIMD parallel computer and a Distributed System is the communication cost (which is less in MIMD)

Μέτρα Απόδοσης Παράλληλων Προγραμμάτων (Parallel Program Performance Measures)

Speedup

$$S = \frac{\text{Running time of best available sequential algorithm}}{\text{Running time of parallel algorithm}}$$

Αν έχω N επεξεργαστές, τότε στην ιδανική περίπτωση Speedup=N

Δυστυχώς, αυτό δεν είναι πάντα (συνήθως) εφικτό διότι:

- ένα πρόβλημα μπορεί να μην αναλύεται σε N ανεξάρτητα υποπροβλήματα
- επιπλέον κόστος ελέγχου (scheduling, συχρονισμός)
- το πρόβλημα μπορεί να περιλαμβάνει ένα εγγενώς σειριακό υποπρόβλημα

Μέτρα Απόδοσης Παράλληλων Προγραμμάτων (Parallel Program Performance Measures)

[Amdahl's Law]

Αν f είναι το ποσοστό του προβλήματος που πρέπει να επιλυθεί σειριακά, τότε **η μέγιστη επιτάχυνση** (speedup) που μπορεί να επιτευχθεί με χρήση N επεξεργαστών είναι:

$$S \leq \frac{1}{f + (1-f)/N} \leq \frac{1}{f}$$

Αν f=0 τότε S <= 1/(0+(1-0)/N) = 1/(1/N)=N

Αν f=1 τότε S <= 1/(1+(1-1)/N) = 1

Αν f=0.5 τότε S <= 1/(0.5+(1-0.5)/N) = 1/(0.5 + 0.5/N)=2N/(N+1)

για N=2 S=4/3 = 1.3

για N=10 S=20/11 = 1.81

Ανάκτηση Πληροφοριών και
Παράλληλη Επεξεργασία

Ανάκτηση Πληροφοριών και Παράλληλη Επεξεργασία

- Προσεγγίσεις
  - (A) Σχεδιασμός **νέων** τεχνικών ΑΠ που να είναι κατάλληλες για παράλληλη επεξεργασία
  - (B) **Προσαρμογή υπαρχόντων** τεχνικών για παράλληλη επεξεργασία
    - θα εστιάσουμε σε αυτή την προσέγγιση και θα δούμε πως γνωστές τεχνικές μπορούν να εφαρμοστούν σε αρχιτεκτονικές MIMD

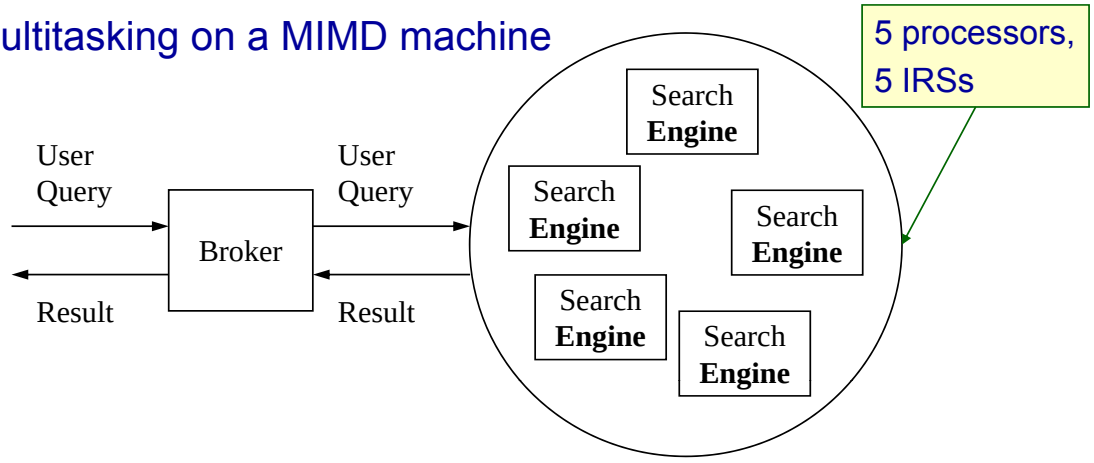
MIMD Architectures

- MIMD (multiple instruction, multiple data)
  - N processors, N instruction streams, N data streams

- Ένα ΣΑΠ μπορεί να εκμεταλευτεί μια MIMD μηχανή με δυο τρόπους:
  - Parallel multitasking;
  - Partitioned parallel processing.

MIMD Architectures: **Parallel Multitasking**

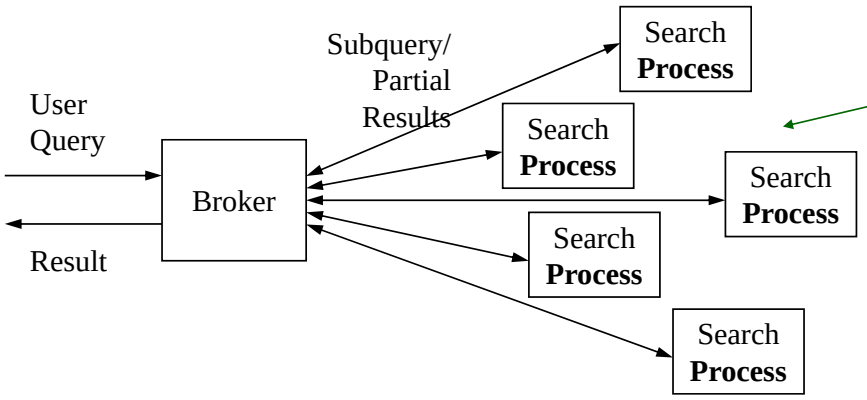
Parallel multitasking on a MIMD machine



- όσο περισσότεροι επεξεργαστές υπάρχουν, τόσο περισσότερες είναι οι επερωτήσεις που μπορούν να απαντηθούν στον ίδιο χρόνο
- ο χρόνος αποτίμησης μιας επερώτησης παραμένει ο ίδιος
- η πρόσβαση στο δίσκο μπορεί να προκαλέσει συμφόρηση
  - αντιμετώπιση: επανάληψη δεδομένων (replication)

MIMD Architectures: **Partitioned Parallel Processing**

Partitioned parallel processing on a MIMD machine

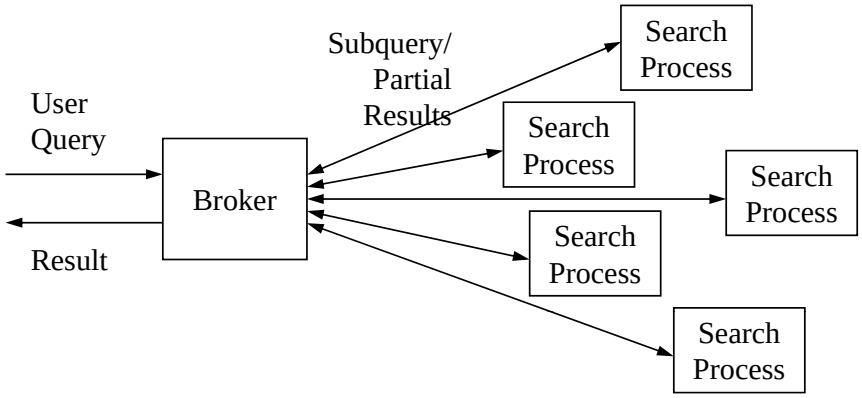


Η αποτίμηση έχει διαμεριστεί σε 5 διαδικασίες. Κάθε μία εκτελείται σε έναν επεξεργαστή

- εδώ ο χρόνος αποτίμησης μιας επερώτησης είναι μικρότερος
  - οι υπολογισμοί για την αποτίμηση **μιας** επερώτησης **κατανέμονται** σε πολλούς επεξεργαστές
  - κάθε επεξεργαστής υπολογίζει ένα τμήμα της επερώτησης και στέλνει τα αποτελέσματα στον Broker.

MIMD Architectures: Partitioned Parallel Processing

Partitioned parallel processing on a MIMD machine



Πώς να διαμερίσουμε την αποτίμηση μιας επερώτησης ;

=>

Πώς να διαμερίσουμε τα δεδομένα ενός ΣΑΠ;

Τα βασικά δεδομένα που επεξεργάζεται ένα αλγόριθμος ανάκτησης

Indexing Items

Documents		k_1	k_2	. . .	k_i	. . .	k_t
	d_1	$w_{1,1}$	$w_{2,1}$	. . .	$w_{i,1}$	. . .	$w_{t,1}$
	d_2	$w_{1,2}$	$w_{2,2}$	. . .	$w_{i,2}$	. . .	$w_{t,2}$
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	d_j	$w_{1,j}$	$w_{2,j}$	. . .	$w_{i,j}$	. . .	$w_{t,j}$
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	d_N	$w_{1.N}$	$w_{2.N}$	. . .	$w_{i.N}$	. . .	$w_{t.N}$

Document Partitioning

Indexing Items

Documents		k_1	k_2	. . .	k_i	. . .	k_t
	d_1	$w_{1,1}$	$w_{2,1}$	. . .	$w_{i,1}$	. . .	$w_{t,1}$
	d_2	$w_{1,2}$	$w_{2,2}$	. . .	$w_{i,2}$	. . .	$w_{t,2}$
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	d_j	$w_{1,j}$	$w_{2,j}$	. . .	$w_{i,j}$	. . .	$w_{t,j}$
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	d_N	$w_{1,N}$	$w_{2,N}$	. . .	$w_{i,N}$	. . .	$w_{t,N}$

- the *N* documents are distributed across the *P* processors
- each parallel process evaluates the query on the subcollection of *N/P* documents assigned to it

Term Partitioning

Indexing Items

Documents	k_1	k_2	. . .	k_i	. . .	k_t
d_1	$w_{1,1}$	$w_{2,1}$	. . .	$w_{i,1}$	. . .	$w_{t,1}$
d_2	$w_{1,2}$	$w_{2,2}$	. . .	$w_{i,2}$	. . .	$w_{t,2}$
. . .	. . .	. . .	. . .	. . .	. . .	. . .
d_j	$w_{1,j}$	$w_{2,j}$	. . .	$w_{i,j}$	. . .	$w_{t,j}$
. . .	. . .	. . .	. . .	. . .	. . .	. . .
d_N	$w_{1,N}$	$w_{2,N}$	. . .	$w_{i,N}$	. . .	$w_{t,N}$

- the *t* indexing items are distributed across the *P* processors
- the evaluation process for each document is spread over multiple processors

Document and Term Partitioning **for Inverted Files**

- Document Partitioning
  - **Physical** Document Partitioning
  - **Logical** Document Partitioning
- Term Partitioning

Παράδειγμα Συλλογής Κειμένων και του Ανεστραμμένου Ευρετηρίου

Document Corpus

Doc	Text
1	Pease porridge hot
2	Pease porridge cold
3	Pease porridge in the pot
4	Pease porridge hot, pease porridge not cold
5	Pease porridge cold, pease porridge not hot
6	Pease porridge hot in the pot

Inverted File

Dictionary	Inverted Lists
cold	<2,1> <4,1> <5,1>
hot	<1,1> <4,1> <5,1> <6,1>
in	<3,1> <6,1>
not	<4,1> <5,1>
pease	<1,1> <2,1> <3,1> <4,2> <5,2> <6,1>
porridge	<1,1> <2,1> <3,1> <4,2> <5,2> <6,1>
pot	<3,1> <6,1>
the	<3,1> <6,1>

MIMD
Inverted Files: **Physical** Document Partitioning

Doc	Text	
1	Pease porridge hot	*P1*
2	Pease porridge cold	
3	Pease porridge in the pot	*P2*
4	Pease porridge hot, pease porridge not cold	
5	Pease porridge cold, pease porridge not hot	*P3*
6	Pease porridge hot in the pot	

Η συλλογή εγγράφων κατανέμεται στους επεξεργαστές

Κάθε υποσυλλογή έχει το **δικό της ανεστραμμένο αρχείο**

P1

cold	<2,1>
hot	<1,1>
pease	<1,1> <2,1>
porridge	<1,1> <2,1>

P2

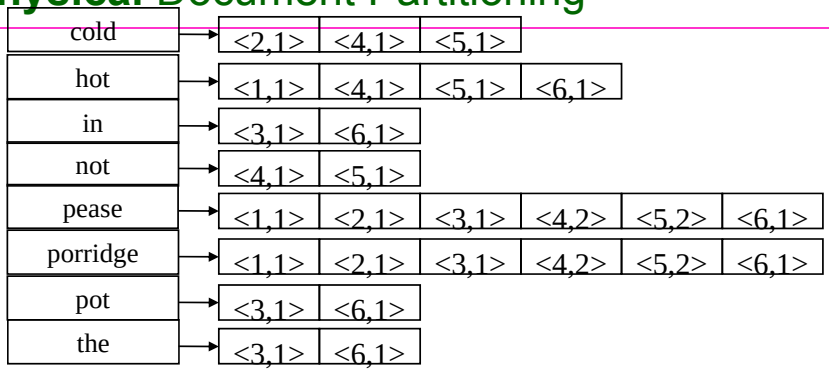
cold	<4,1>
hot	<4,1>
in	<3,1>
not	<4,1>
pease	<3,1> <4,2>
porridge	<3,1> <4,2>
pot	<3,1>
the	<3,1>

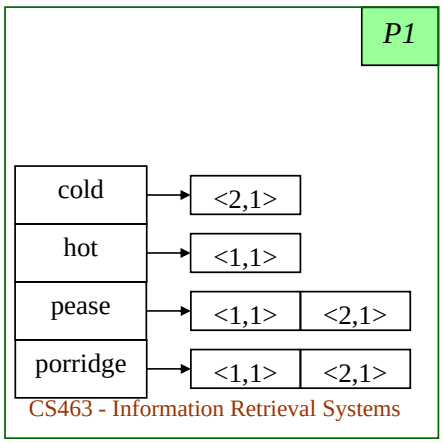
P3

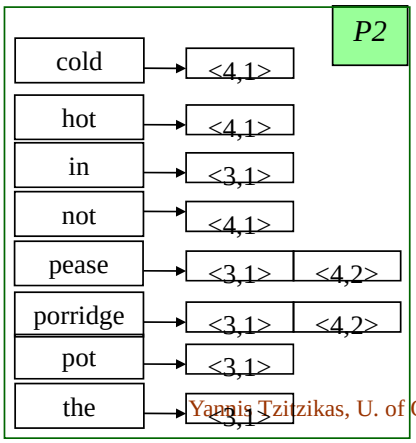
cold	<5,1>
hot	<5,1> <6,1>
in	<6,1>
not	<5,1>
pease	<5,2> <6,1>
porridge	<5,2> <6,1>
pot	<6,1>
the	<6,1>

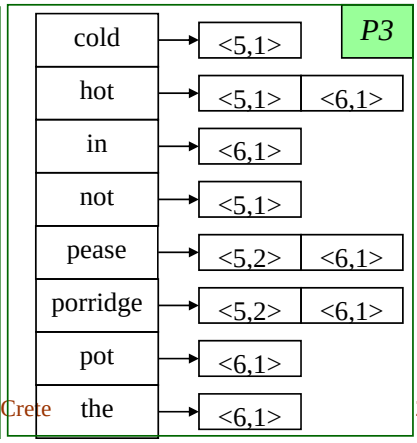
MIMD
Inverted Files: **Physical** Document Partitioning

Original
Inverted File









MIMD
Inverted Files: **Physical** Document Partitioning

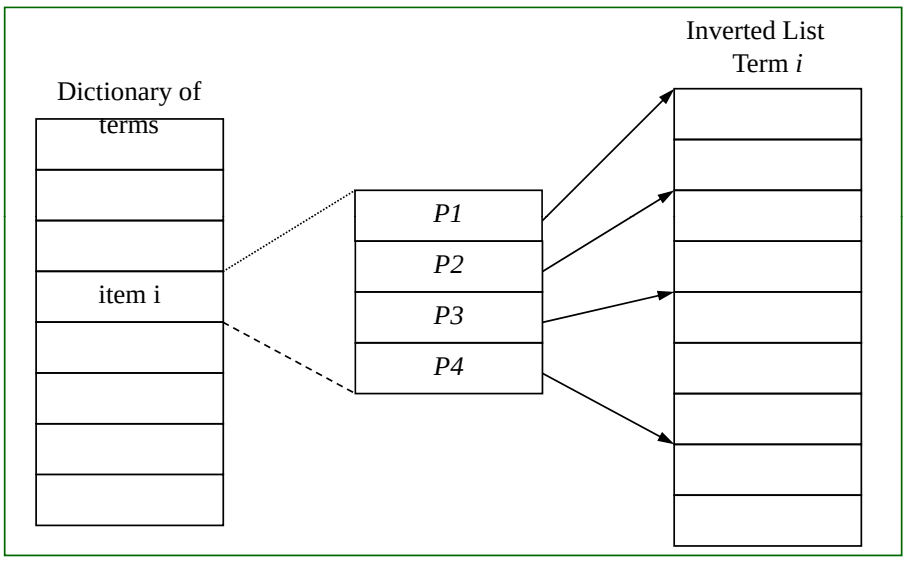
- Κατασκευή Ανεστραμμένων Ευρετηρίων
  - Κάθε επεξεργαστής κατασκευάζει (εν παραλλήλω), ένα **πλήρες ευρετήριο** για τα έγγραφα του.
  - Κάνουμε ένα **βήμα συγχώνευσης** προκειμένου να υπολογίσουμε τα καθολικά στατιστικά (global statistics), δηλαδή **IDF**, και κατόπιν τα στέλνουμε στα ευρετήρια των επεξεργαστών.
- Αποτίμηση Επερωτήσεων
  - Ο μεσίτης (broker) ξεκινά *P* παράλληλες επεξεργασίες
  - Κάθε επεξεργασία εκτελεί τον ίδιο αλγόριθμο (scoring) στα έγγραφα που έχουν εκχωρηθεί στον επεξεργαστή
  - Ο μεσίτης παράγει την τελική διάταξη των εγγράφων

Document and Term Partitioning **for Inverted Files**

- Document Partitioning
  - **Physical** Document Partitioning
  - **Logical** Document Partitioning
- Term Partitioning

MIMD
Inverted Files: **Logical** Document Partitioning

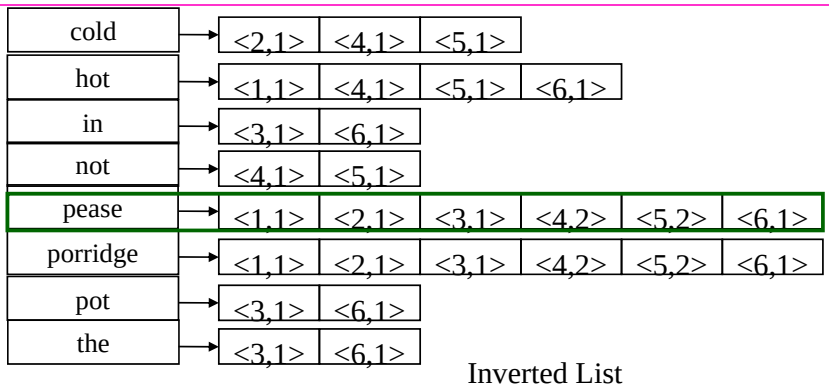
Η συλλογή εγγράφων κατανέμεται στους επεξεργαστές, αλλά κάθε υποσυλλογή **δεν έχει το δικό της ευρετήριο**, αλλά η δομή του ευρετηρίου επιτρέπει στον κάθε επεξεργαστή την άμεση πρόσβαση στο κομμάτι του ευρετηρίου που τον ενδιαφέρει



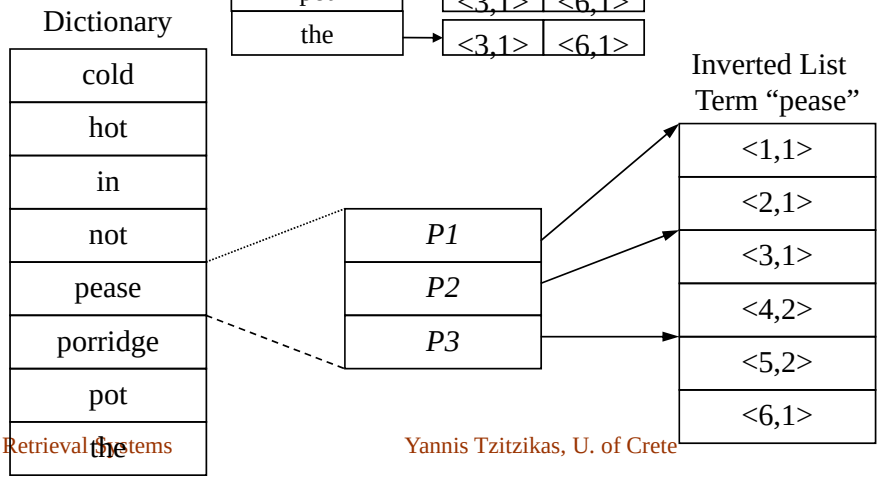
Broker

Logical Document Partitioning

Original Inverted File



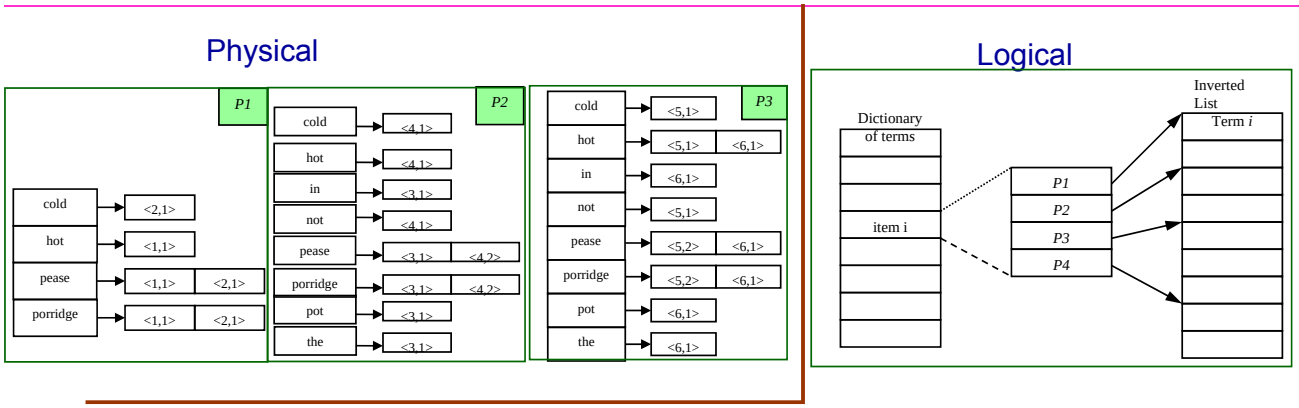
Extended Dictionary



MIMD
Inverted Files: **Logical** Document Partitioning

- Κατασκευή Ανεστραμμένου Ευρετηρίου
  - Τα έγγραφα διαμερίζονται στους επεξεργαστές;
  - Κάθε επεξεργαστής ευρετηριάζει τα δικά του και δημιουργεί **ανεστραμμένες λίστες** ("lala", doc1(2), doc2(6)) , ταξινομημένες αλφαβητικά
  - Συγχωνεύουμε τις λίστες αυτές για έτσι προκύπτει το **τελικό ευρετήριο**
    - θυμηθείτε από την διάλεξη 9: Merging partial indices to obtain the final
- Αποτίμηση Επερωτήσεων (όπως και το Physical Doc. Partitioning)
  - Ο μεσίτης (broker) ξεκινά *P* parallel επεξεργασίες
  - Κάθε επεξεργασία εκτελεί τον ίδιο αλγόριθμο (scoring) στα έγγραφα που έχουν εκχωρηθεί στον επεξεργαστή
  - Τα αποτελέσματα γράφονται σε έναν κοινό πίνακα (shared array)
  - Ο μεσίτης παράγει την τελική διάταξη των εγγράφων

Διαφορές μεταξύ Physical kai Logical document partitioning



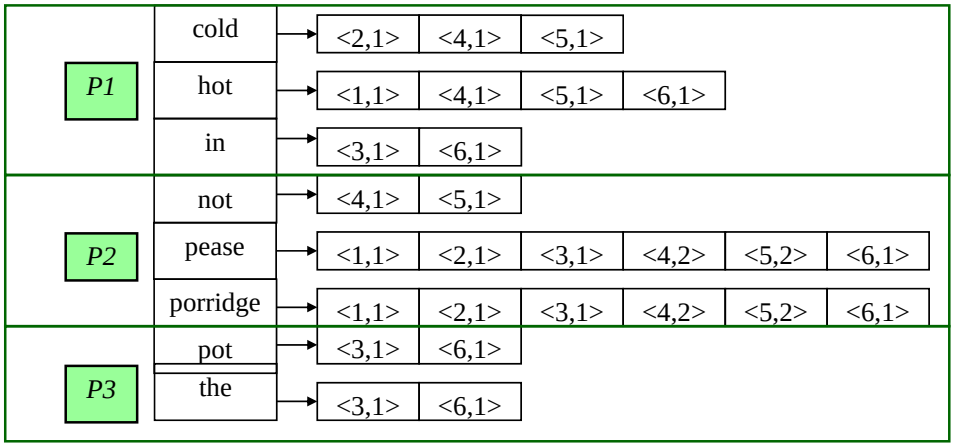
- Logical Document Partitioning
  - Κάθε λέξη του λεξιλογίου είναι αποθηκευμένη μόνο 1 φορά
  - οι διαδικασίες προσπελαύνουν το ίδιο κεντρικό ευρετήριο
    - προσβάσεις ανάγνωσης, άρα δεν έχουμε συμφόρηση
    - λιγότερη επικοινωνία (στο Physical, υπάρχει η φάση υπολογισμού των καθολικών στατιστικών (IDF)).

MIMD Architectures: Partitioned Parallel Processing
Document and Term Partitioning **for Inverted Files**

- Document Partitioning
  - **Physical** Document Partitioning
  - **Logical** Document Partitioning
- **Term Partitioning**

MIMD
Inverted Files:**Term** Partitioning

Term Partitioning



MIMD
Inverted Files:**Term** Partitioning

- Κατασκευή Ανεστραμμένου Ευρετηρίου (όπως στο Log. D. Par.)
  - Τα έγγραφα διαμερίζονται στους επεξεργαστές;
  - Κάθε επεξεργαστής ευρετηριάζει τα δικά του και δημιουργεί **ανεστραμμένες λίστες** (“lala”, doc1(2), doc2(6)) , ταξινομημένες αλφαβητικά
  - Συγχωνεύουμε τις λίστες αυτές για έτσι προκύπτει το **τελικό ευρετήριο**
  - **Κατόπιν το ευρετήριο διαμερίζεται στους επεξεργαστές**

- Αποτίμηση Επερωτήσεων
  - Η επερώτηση αναλύεται στους όρους της, και κάθε ένας στέλνεται στον επεξεργαστή που έχει την αντίστοιχη ανεστραμμένη λίστα
  - Οι επεξεργαστές υπολογίζουν **μερικά-σκορ** (partial document scores) και τα στέλνουν στον μεσίτη
  - Ο μεσίτης υπολογίζει τα τελικά σκορ **συνδιάζοντας τα μερικά**, και παράγει την τελική απάντηση

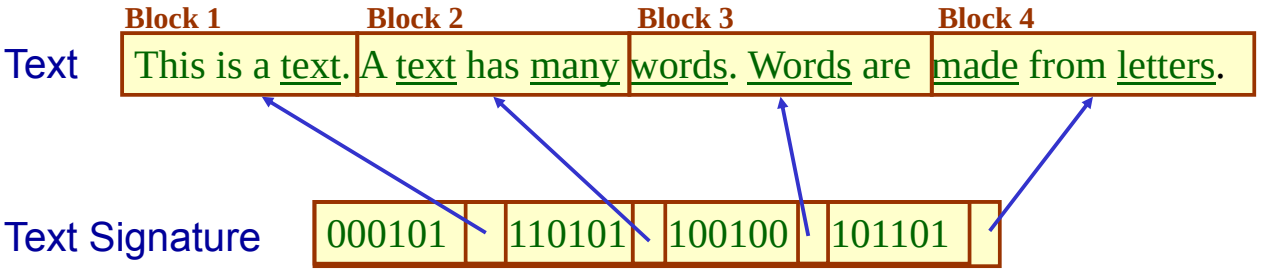
MIMD: Document and Term Partitioning for Inverted Files: **Σύνοψη**

- Οργάνωση ευρετηρίου σε μία MIMD μηχανή:
  - Document partitioning (physical or logical);
  - Term partitioning.
- Document partitioning
  - simpler inverted index construction and maintenance than term partitioning;
  - performs better when term distributions in the documents and queries are more skewed
- Term Partitioning
  - performs better when terms are uniformily distributed in user queries.
  - Επίσης όταν οι επερωτήσεις περιέχουν λίγους όρους

MIMD: Partitioned Parallel Processing
Document Partitioning **for Signature Files**

Signature Files: Επανάληψη

b=3 (3 words per block) B=6 (bit masks of 6 bits)



Signature Function

h(text)= 000101
h(many)= 110000
h(words)=100100
h(made)= 001100
h(letters)=100001

MIMD: Partitioned Parallel Processing

Document Partitioning **for Signature Files**

Doc	Text	
1	Pease porridge hot	*P1*
2	Pease porridge cold	
3	Pease porridge in the pot	*P2*
4	Pease porridge hot, pease porridge not cold	
5	Pease porridge cold, pease porridge not hot	*P3*
6	Pease porridge hot in the pot	

Sign. File 1 | Sign. File 2

Sign. File 3 | Sign. File 4

Sign. File 5 | Sign. File 6

- Each processor creates the signatures of its own documents
- Each processor evaluates the query signature totally. The broker then merges the results

Μέρος Β

Κατανεμημένη Ανάκτηση Πληροφοριών (Distributed Information Retrieval)

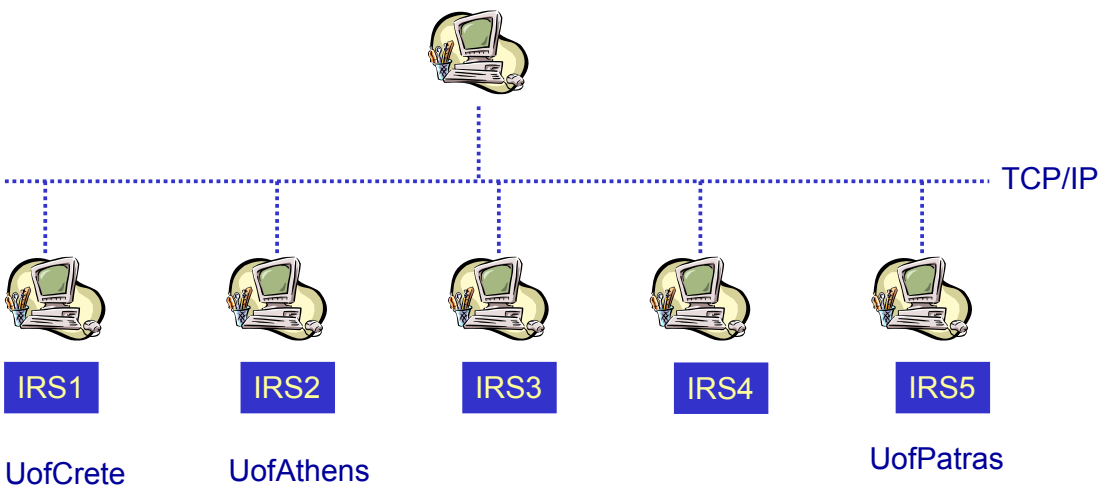
Κατανεμημένη Ανάκτηση Πληροφοριών: Διάρθρωση

- Κίνητρο
- Σχέση μεταξύ Παράλληλης και Κατανεμημένης Αν. Πληροφοριών
- Σχεδιαστικά Ζητήματα
- Διαμέριση Συλλογών και Εγγράφων
- Επιλογή Πηγής
- Ενοποίηση Αποτελεσμάτων

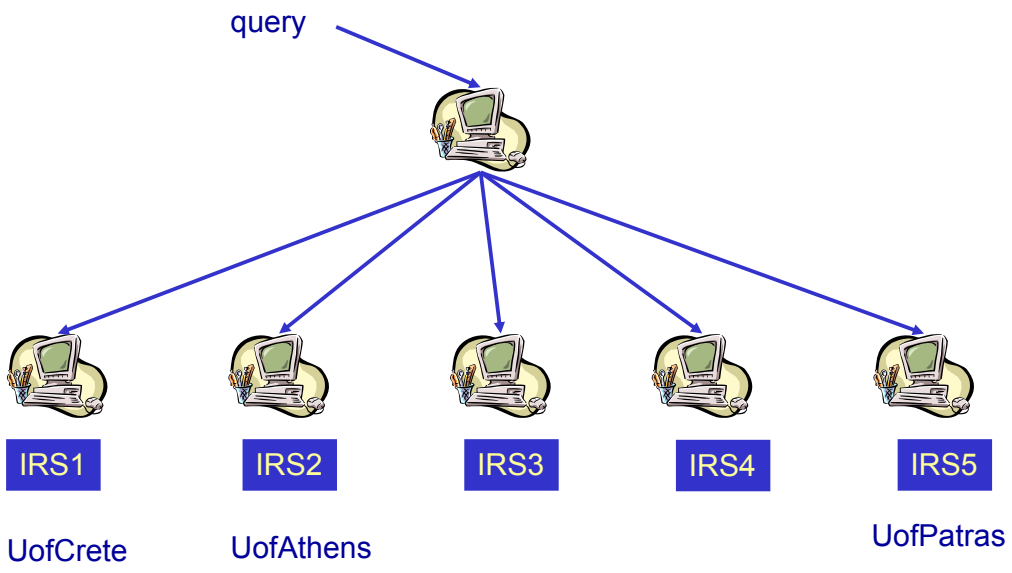
Κατανεμημένη ΑΠ: Κίνητρο

- Το κίνητρο για Παράλληλη ΑΠ ήταν η **βελτίωση της απόδοσης**
- Για την Κατανεμημένη ΑΠ δεν είναι μόνο αυτό.
- Είναι και η ανάγκη **ενοποιημένης πρόσβασης** στα έγγραφα πολλών συστημάτων ανάκτησης πληροφοριών

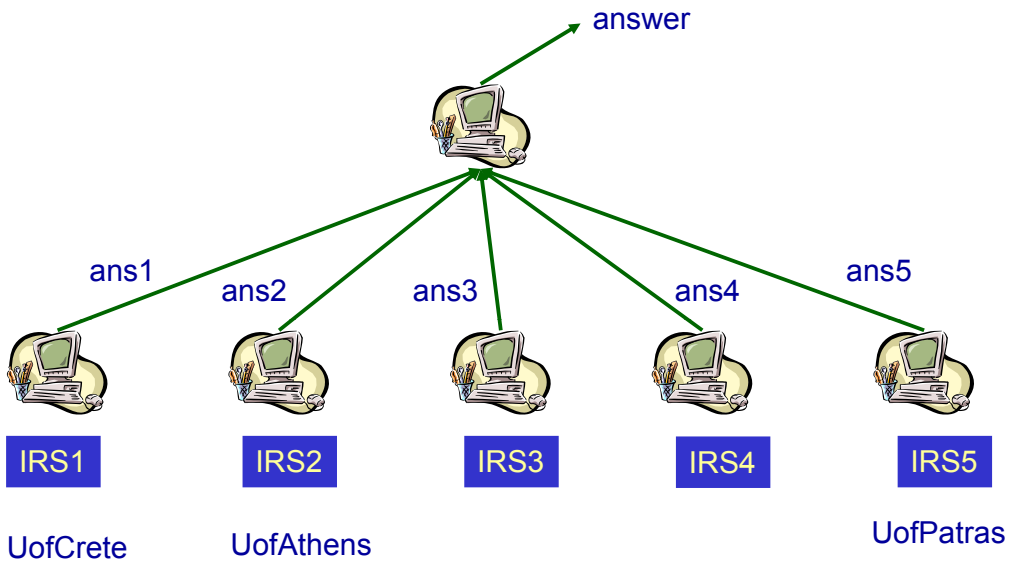
Κατανεμημένη Ανάκτηση Πληροφοριών



Κατανεμημένη Ανάκτηση Πληροφοριών

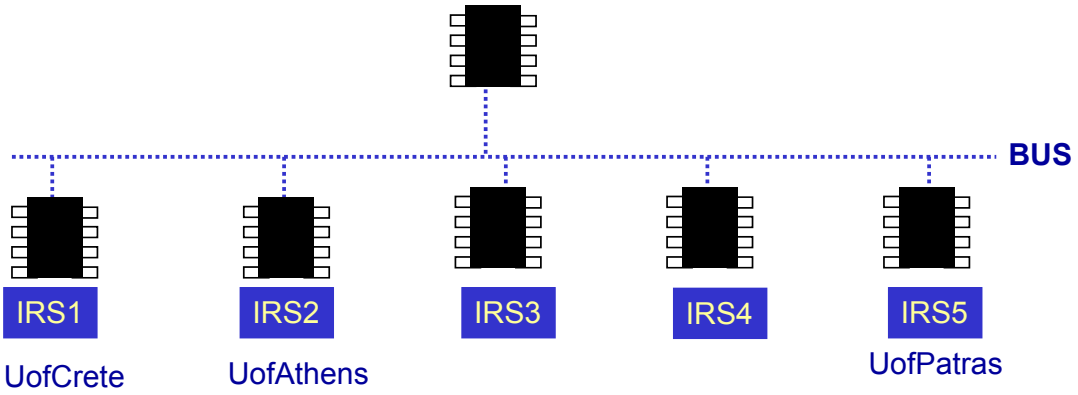


Κατανεμημένη Ανάκτηση Πληροφοριών



Ποια η Σχέση μεταξύ Παράλληλης και Κατανεμημένης Ανάκτησης Πληροφοριών;

- Η κατανεμημένη μοιάζει με την παράλληλη αρχιτεκτονική MIMD
- Διαφορές με την MIMD
  - το **κανάλι επικοινωνίας** μεταξύ των επεξεργαστών είναι πολύ πιο αργό
  - δεν έχουμε τους ίδιους επεξεργαστές (όπως σε μια παράλληλη μηχανή)
  - στην κατανεμημένη ο μεσίτης (broker) συχνά επιλέγει να χρησιμοποιήσει μόνο ένα υποσύνολο των υποκείμενων συστημάτων

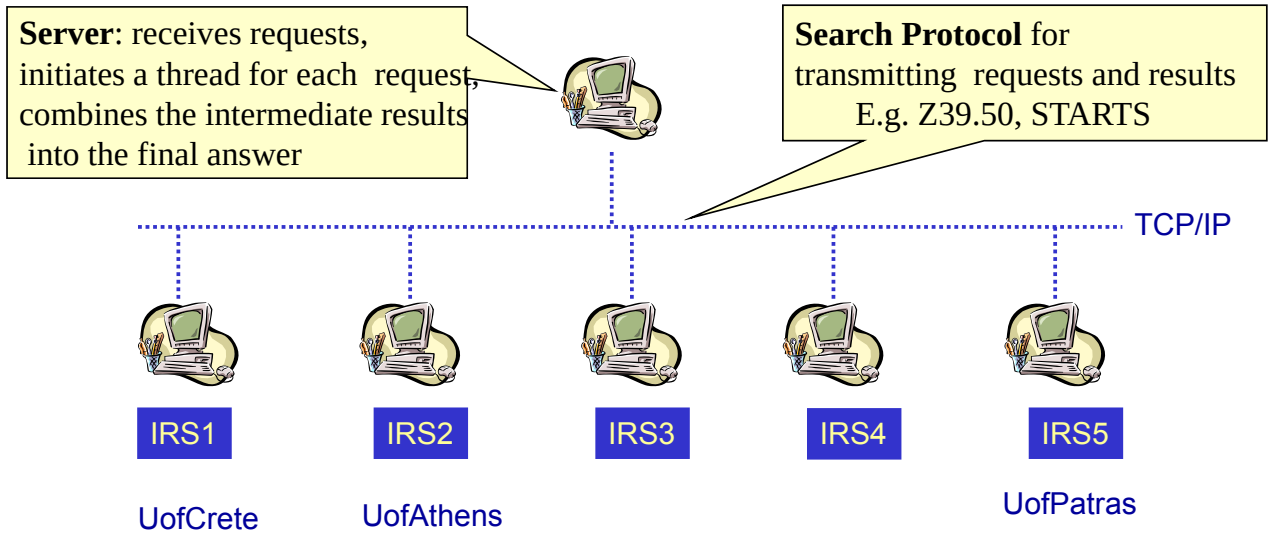


Σχέση μεταξύ Παράλληλης και Κατανεμημένης Αν. Πλ.

Ποια προσέγγιση του **Partitioned Parallel Processing** είναι κατάλληλη για την Κατανεμημένη Ανάκτηση;

- **document partitioning:** ενδείκνυται για την κατανεμημένη ανάκτηση
- **term partitioning:** δεν είναι πολύ καλή διότι απαιτεί περισσότερη επικοινωνία (για αυτό σπάνια υιοθετείται από ένα κατανεμημένο σύστημα

Κατανεμημένη Ανάκτηση: Σχεδιαστικά ζητήματα

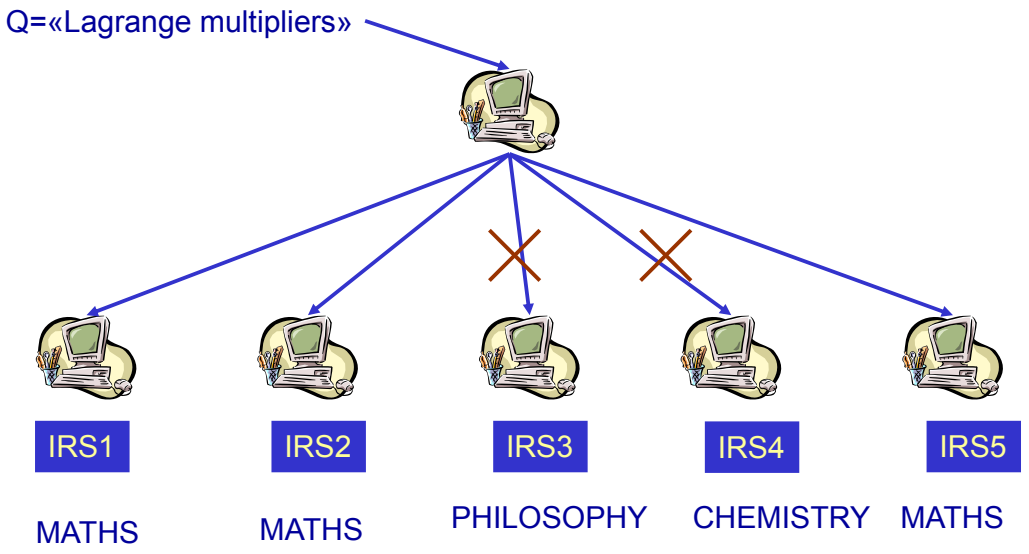


- **distribute** documents across servers
- **selection** of the servers to receive a particular request
- **combine** the results of multiple servers

Διαμερισμός Συλλογών (Collection Partitioning)

- Δεν τίθεται τέτοιο ζήτημα αν τα υποκείμενα συστήματα είναι ετερογενή
- Σενάρια για την περίπτωση που υπάρχει κεντρικός έλεγχος:
  - **Semantic**-based partition of collections to servers
    - Search Servers focusing on a particular subject area
      - E.g. Maths, Physics, etc
  - **Semantic**-based partition of documents to servers
    - π.χ. με χρήση ενός αλγορίθμου ομαδοποίησης (clustering)
  - **Replications** of collections to all servers
    - για βελτίωση throughput (μοιάζει με το multitasking)
    - όταν οι συλλογές δεν είναι μεγάλες
  - **Τυχαία διανομή** εγγράφων στους servers
    - για βελτίωση της απόδοσης στην περίπτωση που η συλλογή είναι πολύ μεγάλη

Επιλογή Πηγής (Source Selection)



Source Selection: Επιλογή των συλλογών που είναι πιθανόν να έχουν συναφή έγγραφα με την τρέχουσα επερώτηση

Για ποιο λόγο να κάνουμε Επιλογή Πηγής;

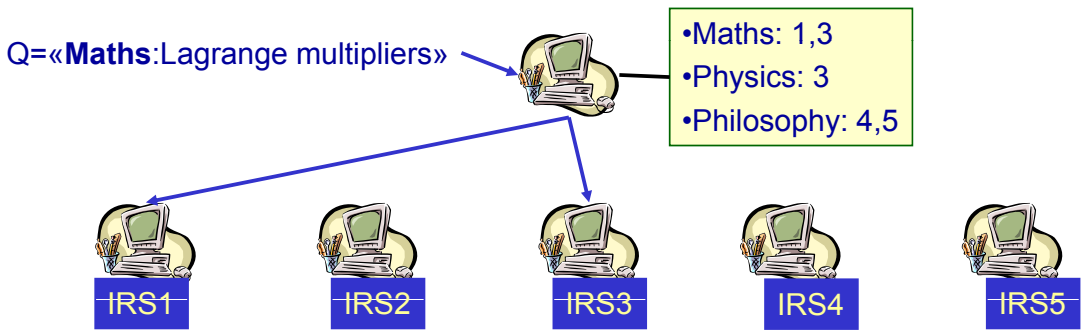
- Η αναζήτηση σε κάθε συλλογή μπορεί:
  - να είναι **ακριβή σε χρόνο** (αφού μπορεί να έχουμε εκατοντάδες συλλογές)
  - να είναι **ακριβή σε χρήμα** (η αναζήτηση μπορεί να έχει χρηματικό κόστος)
  - να καθορίσει την **αποτελεσματικότητα** (effectiveness) της ανάκτησης

Επιλογή Πηγής: Επιλογή **Όλων**

- Κατάλληλη κυρίως για διαμερισμό μιας μεγάλης συλλογής πάνω από ένα **τοπικό δίκτυο**
- Εύκολη ενοποίηση αποτελεσμάτων για το Boolean model
  - answer(q) = ans1(q) ∪ … ∪ ansk(q)
- Η ενοποίηση αποτελεσμάτων για τα στατιστικά μοντέλα είναι πιο δύσκολη (*το ζήτημα αυτό θα μελετηθεί παρακάτω*)

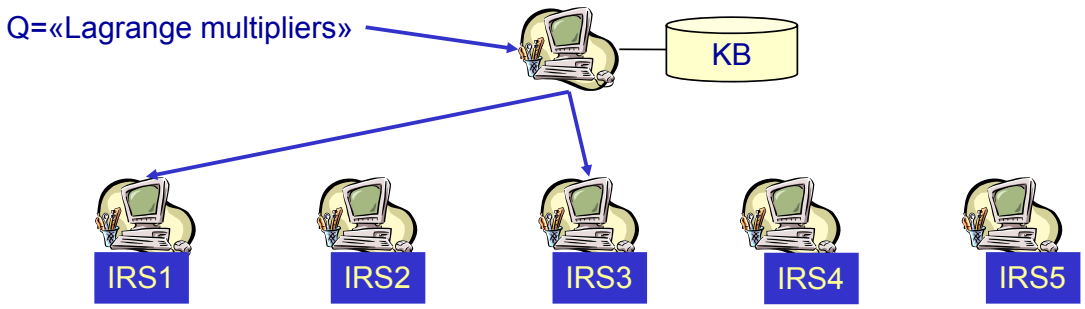
Επιλογή Πηγής: **Χειρονακτική Ομαδοποίηση** και Επιλογή

- Θεματική οργάνωση συλλογών (χειρονακτικώς)
  - πχ ***μαθηματικά, φυσική, ειδήσεις***, κλπ
  - προβλήματα
    - χρονοβόρα διαδικασία, ευάλωτη σε ασυνέπειες/παραλείψεις, δεν θα δουλέψει καλά για μη-συνηθισμένες επερωτήσεις
- Ο χρήστης επιλέγει τη θεματική κατηγορία



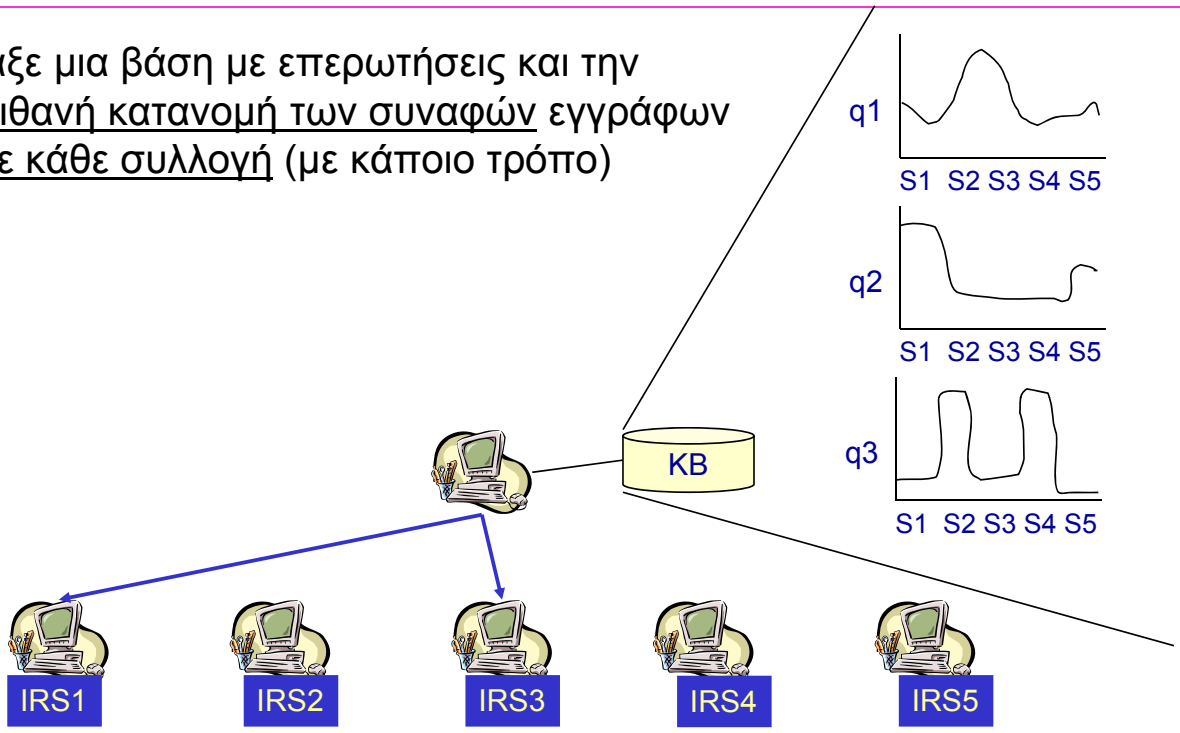
Επιλογή Πηγής βάσει **Κανόνων** (Rule-based)

- Τα περιεχόμενα κάθε συλλογής περιγράφονται σε μια **Βάση Γνώσης**
- Ένα Σύστημα Κανόνων επιλέγει τις πηγές για κάθε εισερχόμενη επερώτηση
- Αδυναμίες
  - κόστος συγγραφής κανόνων
  - ανάγκη συντήρησης των κανόνων (αν οι συλλογές είναι δυναμικές)



Επιλογή Πηγής: **Κατανομή Συναφών** Εγγράφων (Relevant Document Distribution (RDD))

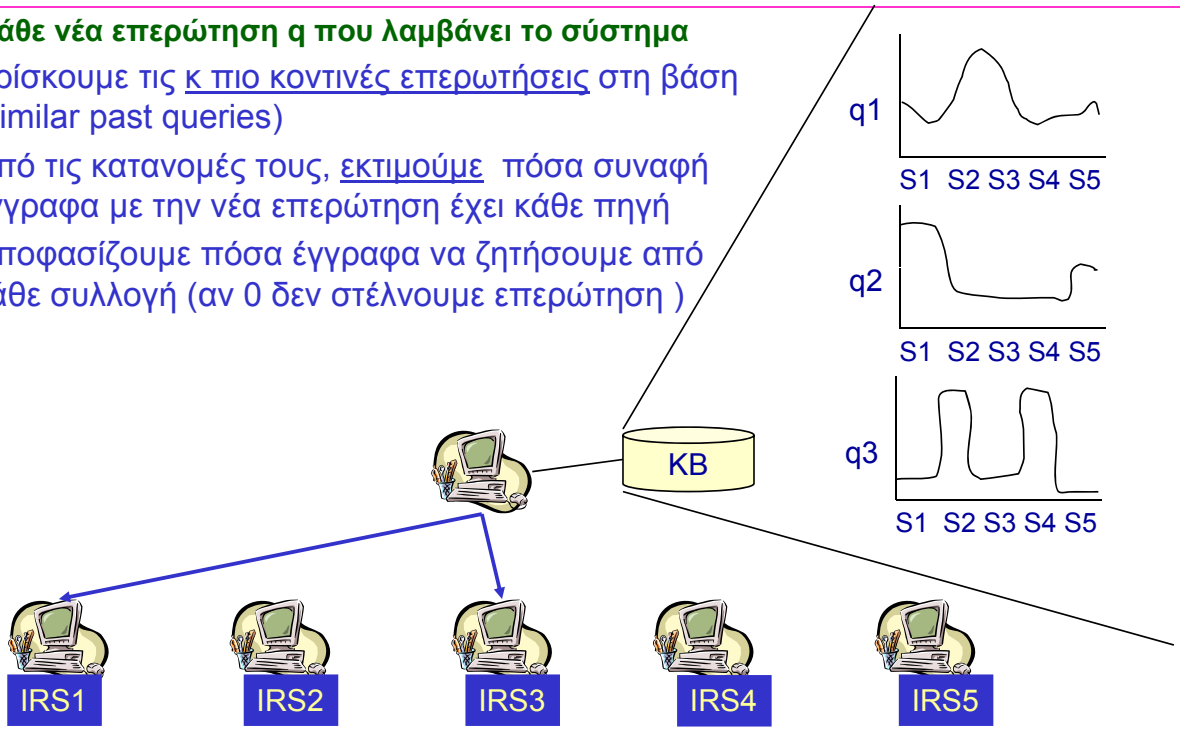
Φτιάξε μια βάση με επερωτήσεις και την πιθανή κατανομή των συναφών εγγράφων σε κάθε συλλογή (με κάποιο τρόπο)



Επιλογή Πηγής: **Κατανομή Συναφών** Εγγράφων (Relevant Document Distribution (RDD))

Για κάθε νέα επερώτηση q που λαμβάνει το σύστημα

- Βρίσκουμε τις κ πιο κοντινές επερωτήσεις στη βάση (similar past queries)
- Από τις κατανομές τους, εκτιμούμε πόσα συναφή έγγραφα με την νέα επερώτηση έχει κάθε πηγή
- Αποφασίζουμε πόσα έγγραφα να ζητήσουμε από κάθε συλλογή (αν 0 δεν στέλνουμε επερώτηση)

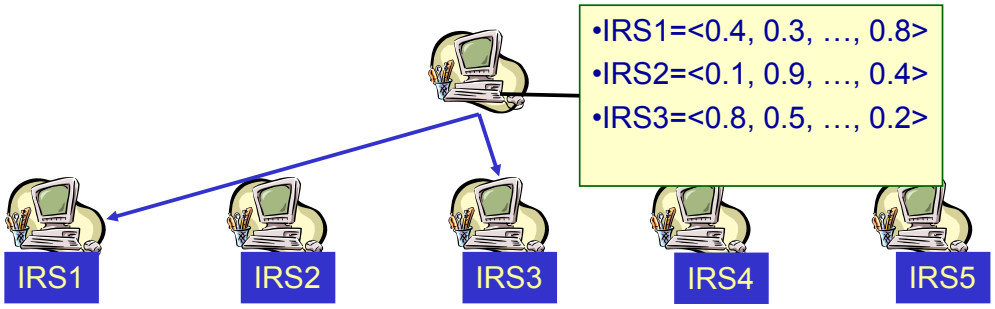


Επιλογή Πηγής: **Επερώτηση Βολιδοσκόπησης** (Query Probing)

- Στέλνουμε μια **επερώτηση βολιδοσκόπησης** σε κάθε συλλογή (που μπορεί να περιλαμβάνει μερικούς από τους όρους της επερώτησης)
  - κάθε συλλογή απαντά με στατιστικές πληροφορίες
    - πχ: μέγεθος συλλογής, πόσα έγγραφα έχουν τον κάθε όρο, πόσα έγγραφα έχουν όλους τους όρους της επερώτησης, κλπ
  - βάσει αυτών των στοιχείων επιλέγουμε την πηγή

- Υποθέσεις
  - η επεξεργασία των επερωτήσεων βολιδοσκόπησης είναι πολύ φθηνότερη
    - περιέχουν λίγους όρους, δεν χρειάζεται να υπολογίσουμε βαθμούς συνάφειας ή να διατάξουμε τα έγγραφα ως προς τη συνάφεια τους

Επιλογή Πηγής με **Διανύσματα Πηγών**



- Βλέπουμε **κάθε συλλογή** ως ένα **μεγάλο έγγραφο**
- Φτιάχνουμε ένα **διάνυσμα για κάθε συλλογή** (τύπου TF-IDF)
  - tfij: συνολικές εμφανίσεις του όρου i στη συλλογή j
  - idfi: log(N/ni), όπου N το πλήθος των συλλογών, και ni το πλήθος των συλλογών που έχουν τον όρο i
- Υπολογίζουμε το **βαθμό ομοιότητας κάθε νέας επερώτησης με το διάνυσμα κάθε συλλογής** (π.χ. ομοιότητα συνημίτονου)
- Διατάσσουμε τις συλλογές και επιλέγουμε τις κορυφαίες

Επιλογή Πηγής με **Διανύσματα Πηγών (II)**

- Μια αδυναμία:
  - Μπορεί ο βαθμός ομοιότητας με μία συλλογή να είναι μεγάλος, αλλά να μην υπάρχει κανένα έγγραφο εκεί με μεγάλο βαθμό συνάφειας
- Ένας τρόπος αντιμετώπισης:
  - Για κάθε συλλογή φτιάξε N/B διανύσματα, δηλαδή ένα διάνυσμα για κάθε B έγγραφα της συλλογής
  - Αν B=1 τότε ο server είναι σαν να έχει το ευρετήριο όλων των συστημάτων
  - Αν B=N τότε έχουμε ένα διάνυσμα για κάθε συλλογή

Επιλογή Πηγής: **GlOSS**
(**Gl**ossary of **S**ervers **S**erver)

- Estimate the number of potentially relevant documents in a collection C for a Boolean AND query Q as:

$$|C| \cdot \prod_{t \in Q} \frac{df_t}{|C|}$$

$$df_t : \text{number of docs in C that contain t}$$

- |C| the number of documents in the collection C
- Requires that for each collection *C we* have an entry in a centralized index
  - centralized index is small, easy to maintain

Επιλογή Πηγής: **gGlOSS** και **hGlOSS**

- **gGlOSS**
  - Extends the GlOSS approach to the vector space model
    - Each collection is represented by its centroid vector
    - Standard inner product similarity measure of query to each collection
    - Rank collections accordingly
- **hGlOSS** (hierarchical GlOSS)
  - Extends the gGlOSS approach to sets of gGlOSS indexes
  - Each gGlOSS index is represented by its centroid vector

Επιλογή Πηγής: Σύνοψη

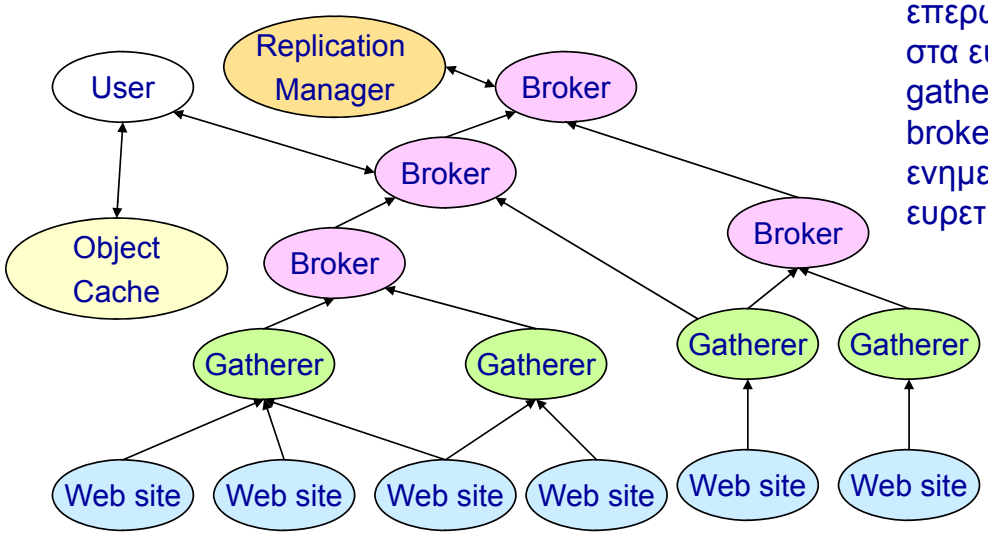
- Προσεγγίσεις για μικρό αριθμό συλλογών
  - Επιλογή Όλων
  - Χειρονακτική Ομαδοποίηση (και χειρονακτική επιλογή)
  - Επιλογή βάσει Κανόνων (Rule-based selection)
  - Κατανομή Συναφών Εγγράφων (Relevant document distribution (RDD))
  - Βολιδοσκόπηση Επερώτησης (Query Probing)
  - Διανύσματα Πηγών
- Προσεγγίσεις για μεγάλο αριθμό συλλογών
  - Διανύσματα Πηγών
  - GlOSS

HARVEST

A distributed architecture to gather and distribute data
- used by CIA, NASA, US National Academy of Sciences

Gatherers: ευρετηριάζουν >=1 Web Server περιοδικά

Brokers: απαντούν επερωτήσεις βασιζόμενοι στα ευρετήρια των gatherers ή άλλων brokers (και ενημερώνουν αυξητικά τα ευρετήρια τους)



On writing parallel/distributed programs (the case of Google)

- To reducing the complexity of writing parallel programs, Google, has adopted the **Map Reduce** technique
  - A typical first-week training assignment for a new programmer hired by Google is to write a software routine that uses **MapReduce** to count all occurrences of words in a set of Web documents.
  - In that case, the "**map**" would involve tallying all occurrences of each word on each page—not bothering to add them at this stage, just ticking off records for each one like hash marks on a sheet of scratch paper.
  - The programmer would then write a "**reduce"** function to do the math—in this case, taking the scratch paper data, the intermediate results, and producing a count for the number of times each word occurs on each page.
  - One example, from a Google developer presentation, shows how the phrase "to be or not to be" would move through this process.

MAP

key	TO	BE	OR	NOT	TO	BE
value	1	1	1	1	1	1

REDUCE

key	TO	BE	OR	NOT
value	2	2	1	1