HY463 - Συστήματα Ανάκτησης Πληροφοριών
Information Retrieval (IR) Systems

Parallel and Distributed IR
Παράλληλη και Κατανεμημένη ΑΠ

Γιάννης Τζίτζικας

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.

Παράλληλα Συστήματα

- **Παράλληλη Επεξεργασία**: η ταυτόχρονη λειτουργία πολλών πόρων (επεξεργαστών ή μονάδων I/O) για την επίλυση ενός προβλήματος.
- Η χρήση πολλών πόρων μπορεί να βελτιώσει σημαντικά το χρόνο επεξεργασίας ενός ερωτήματος, διότι η κάθε μονάδα συνεισφέρει στην επίλυση ενός μέρους του αρχικού προβλήματος.

Παράλληλα Συστήματα

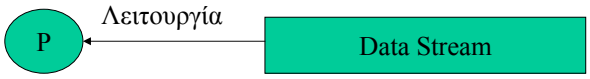
- Παραλληλισμός μπορεί να εφαρμοστεί σε όλους τους πόρους ενός συστήματος.
- Συνήθως αναφερόμαστε σε :
  - παραλληλισμό επεξεργαστών (processor ή CPU parallelism)
  - παραλληλισμό συστήματος εισόδου-εξόδου (I/O parallelism)

Παράλληλος Προγραμματισμός

- Παράλληλος Προγραμματισμός: Η ταυτόχρονη χρήση πολλών επεξεργαστών για την επίλυση ενός προβλήματος
- Ταξινόμηση 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)

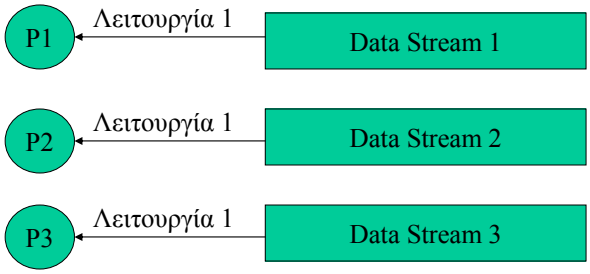
SISD

- Αποτελεί την κλασική μηχανή von Neumann, όπου έχουμε μόνο έναν επεξεργαστή ο οποίος εκτελεί μία ακολουθία εντολών σε ένα stream δεδομένων.



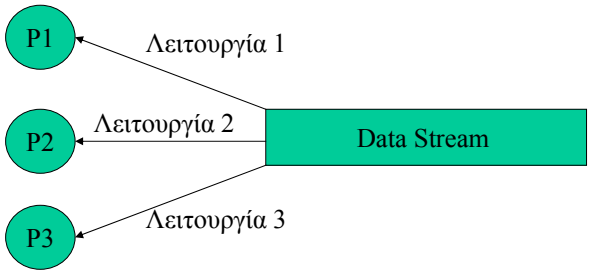
SIMD

- Αποτελείται από *N* επεξεργαστές οι οποίοι επεξεργάζονται *N* διαφορετικά δεδομένα εφαρμόζοντας την ίδια λειτουργία.
- Οι επεξεργαστές είναι συγχρονισμένοι ώστε να εκτελούν την ίδια λειτουργία στα δεδομένα.
- Παράδειγμα SIMD συστήματος: Thinking Machines CM-2.
- Κάθε επεξεργαστής μπορεί να έχει τη δική του μνήμη ή όλοι επεξεργαστές μπορούν να μοιράζονται την ίδια μνήμη.



MISD

- *N* επεξεργαστές οι οποίοι μπορούν να εκτελούν διαφορετικές λειτουργίες στα ίδια δεδομένα.
- Όλοι οι επεξεργαστές μοιράζονται την ίδια μνήμη.
- Τα συστήματα MISD δεν είναι ευρέως διαδεδομένα.
- Παράδειγμα: Systolic Arrays

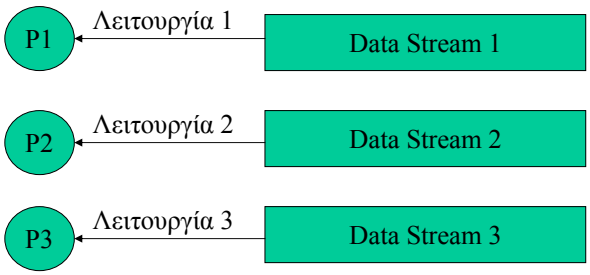


MIMD

- Είναι η πιο γενική και πιο διαδεδομένη παράλληλη αρχιτεκτονική συστημάτων.
- Έχουμε *N* επεξεργαστές και *N* δεδομένα, και κάθε επεξεργαστής μπορεί να εκτελεί διαφορετική λειτουργία.
- Κάθε επεξεργαστής μπορεί να έχει τη δική του μνήμη ή όλοι να μοιράζονται την ίδια.
- Παραδείγματα: Sun HPC Server, IBM SP2

Ανάλογα με το βαθμό επικοινωνίας των επεξεργαστών μεταξύ τους διακρίνουμε:

- Tightly-coupled συστήματα.
- Loosely-coupled συστήματα.



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:

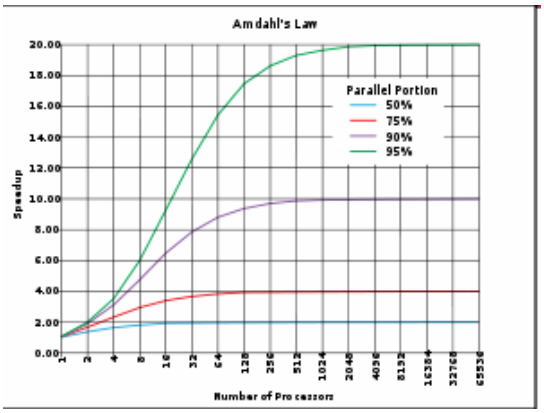
- ένα πρόβλημα μπορεί να μην αναλύεται σε N ανεξάρτητα υποπροβλήματα
- επιπλέον κόστος ελέγχου (scheduling, συγχρονισμός)
- το πρόβλημα μπορεί να περιλαμβάνει ένα *εγγενώς σειριακό υποπρόβλημα*

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:

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



Μέτρα Απόδοσης Παράλληλων Προγραμμάτων
(Parallel Program Performance Measures)

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

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

Μέτρα Απόδοσης

- **Speedup** = χρόνος σειριακού αλγορίθμου προς το χρόνο παράλληλου αλγορίθμου (θέλουμε γραμμικότητα)

Αν το πρόβλημα (δεδομένα) μεγαλώνουν:

- **Scaleup** = χρόνος επίλυσης μικρού προβλήματος σε μικρό σύστημα προς το χρόνο επίλυσης μεγάλου προβλήματος σε μεγάλο σύστημα (θέλουμε σταθερότητα)
- **Sizeup** = χρόνος επίλυσης μικρού προβλήματος σε μεγάλο σύστημα προς το χρόνο επίλυσης μεγάλου προβλήματος σε μεγάλο σύστημα (θέλουμε γραμμικότητα)

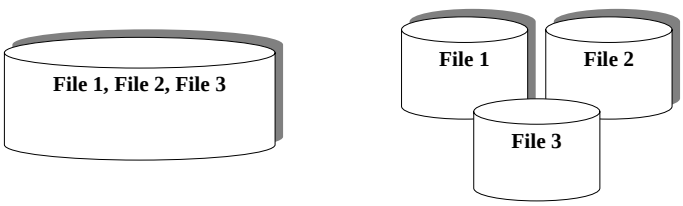
Μέτρα Απόδοσης

Μέτρα απόδοσης από την πλευρά κάθε εργασίας:

- Ρυθμός εξόδου ή ολοκλήρωσης (throughput): ο αριθμός των εργασιών που μπορούν να ολοκληρωθούν σε συγκεκριμένο χρονικό διάστημα
- Χρόνος απόκρισης (response time): ο χρόνος που απαιτείται για να ολοκληρωθεί *μία* εργασία από τη στιγμή που θα υποβληθεί
- Ολικός χρόνος (total time) vs Response (or parallel) time: αθροιστικά ο συνολοκός χρόνος για μία εργασία

Παραλληλισμός I/O

- Κατά τη διάρκεια επεξεργασίας ενός ερωτήματος πολύς χρόνος δαπανάται για λειτουργίες I/O.
- Θέλουμε οι λειτουργίες αυτές να εκτελούνται γρηγορότερα.
- Ο τρόπος με τον οποίο διαμοιράζονται τα δεδομένα στους δίσκους παίζει σημαντικό ρόλο στην επίδοση του συστήματος.
- Ο διαμοιρασμός (declustering) μπορεί να πραγματοποιηθεί:
  - από το υλικό (π.χ. I/O controller)
  - από το λογισμικό (π.χ. λειτουργικό σύστημα, database)



Παραλληλισμός I/O

Ο παραλληλισμός στοχεύει:

- Κατανομή του φόρτου πολλών μικρών αιτήσεων I/O με στόχο την αύξηση του *throughput*.
- Παραλληλισμός μεγάλων αιτήσεων I/O με στόχο τη μείωση του *χρόνου απόκρισης* (response time).

Χρήση RAID (Redundant Arrays of Inexpensive Disks) - Redundant Array of Independent Disks

increased data reliability or increased input/output performance

Παραλληλισμός I/O

Data striping: the data is segmented into equal-sized partitions that are distributed over multiple disks

Striping unit: size of a partition

Usually, using a round robin algorithm: if the disk array consists of D disks, partition i is written onto disk i mod D

Round robin => large requests of the size of many contiguous blocks involve all disks => process the request by all disks in parallel => increase the transfer rate to the aggregated bandwidth of the D disks

Παραλληλισμός I/O

Multiple disks may lower overall reliability

Increase reliability through redundancy

- Where to store redundant information?
  - on a few check disks
  - distribute it uniformly across all disks
- How to compute redundant information
  - parity scheme: an extra check disk contains information that can be used to recover from failure of any one disk in the array

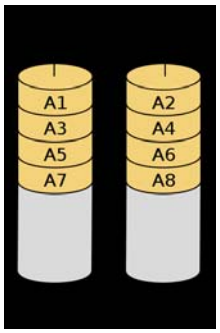
Consider the first bit in each disk

Let i of the D data bits be one => first bit on the check disk is 1 if i is odd, 0 otherwise

RAID Levels

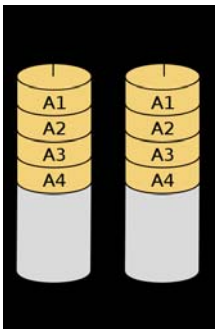
RAID LEVEL 0

- Κάθε δίσκος αποθηκεύει τα δικά του δεδομένα. Δεν υπάρχει επανάληψη.
- Σε περίπτωση που έχουμε failure σε έναν δίσκο τα περιεχόμενα καταστρέφονται.



RAID LEVEL 1

- Mirroring (redundancy)
- Κάθε δίσκος έχει και έναν αντίστοιχο ο οποίος περιέχει αντίγραφο των δεδομένων
- No stripping
- Parallel reads of the same disk block!



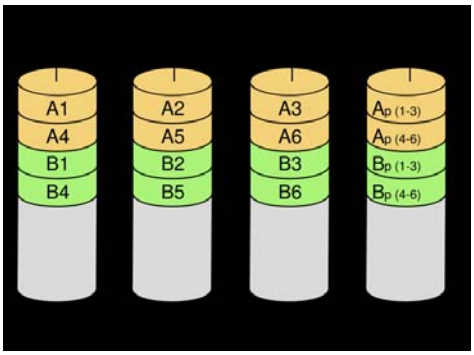
RAID Levels

RAID LEVEL 2

- Memory-style error correcting codes
- Stripping unit: single bit
- Για κάθε byte δεδομένων αποθηκεύεται και ένα parity bit, ώστε να μπορεί να γίνει ανακατασκευή δεδομένων σε περίπτωση που έχουμε failure

RAID LEVEL 3

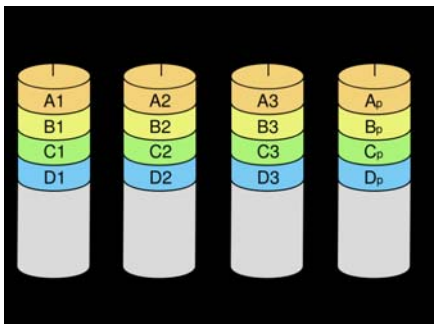
- Bit-interleaved parity
- Σε αντίθεση με το level 2 χρησιμοποιεί έναν δίσκο για parity.
- Μειονέκτημα: σε κάθε I/O συμμετέχουν όλοι οι δίσκοι.



RAID Levels

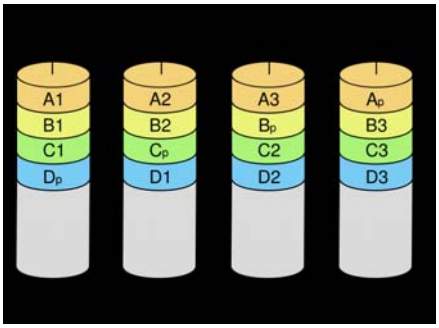
RAID LEVEL 4

- Block-interleaved parity
- Stripping at the block level
- Σε κάθε δίσκο αποθηκεύεται block δεδομένων και χρησιμοποιείται το parity για κάθε block



RAID LEVEL 5

- Block-interleaved distributed parity
- Τα parity δεδομένα αποθηκεύονται σε όλους τους δίσκους, μαζί με τα δεδομένα.



Διαμοιρασμός Δεδομένων

Declustering, Partitioning: Μέθοδοι διαμοιρασμού των δεδομένων στους διαθέσιμους δίσκους.

Ο τρόπος διαμοιρασμού των δεδομένων έχει άμεση σχέση με την επίδοση ενός ερωτήματος.

Στόχοι:

- Να ενεργοποιούνται όσο το δυνατό *λιγότεροι δίσκοι*.
- Οι δίσκοι που ενεργοποιούνται να εκτελούν *ισοδύναμες εργασίες*, ώστε ο χρόνος επεξεργασίας κάθε δίσκου να είναι περίπου ο ίδιος.

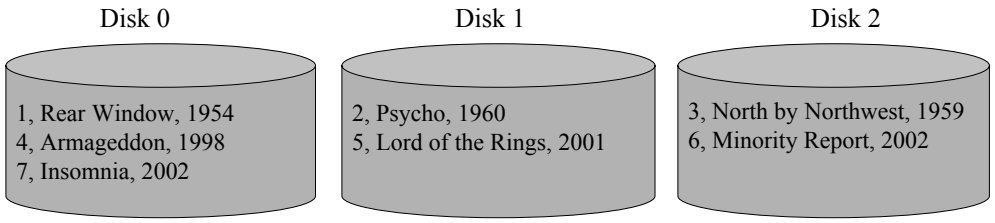
Υπάρχουν τρεις διαφορετικές τεχνικές διαμοιρασμού των δεδομένων ενός πίνακα σε ένα σύνολο δίσκων:

- Round-Robin
- Hash Partitioning
- Range Partitioning

Διαμοιρασμός Δεδομένων

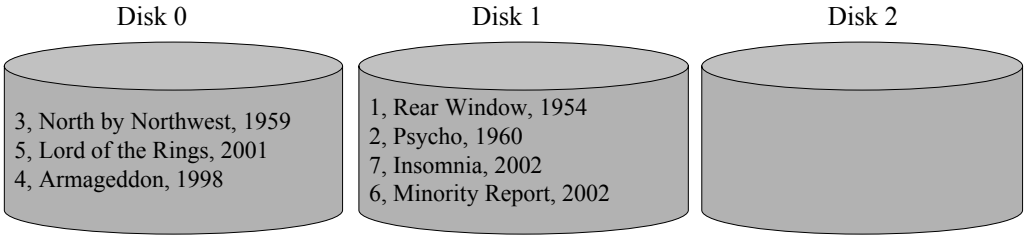
ID	Τίτλος Ταινίας	Χρονιά
1	Rear Window	1954
2	Psycho	1960
3	North by Northwest	1959
4	Armageddon	1998
5	Lord of the Rings	2001
6	Minority Report	2002
7	Insomnia	2002

Round-Robin



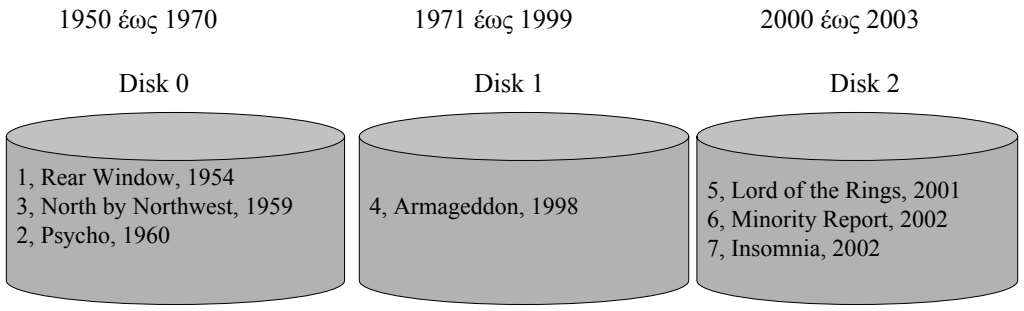
Hash Partitioning

Χρησιμοποιούμε τη συνάρτηση κατακερματισμού: [Χρονιά] MOD 3



Range Partitioning

Ο πίνακας χωρίζεται σε περιοχές (ranges) και κάθε περιοχή αποθηκεύεται σε ξεχωριστό δίσκο.



Σύγκριση

Ερωτήματα:

SELECT * FROM movies (relation scanning)

SELECT * FROM movies
WHERE title="Armageddon" (point query)

SELECT * FROM movies
WHERE year BETWEEN (1970,2000) (range query)

Να συζητηθεί η απόδοση των μεθόδων διαμοιρασμού για τα ερωτήματα.

Διαμοιρασμός Δομών

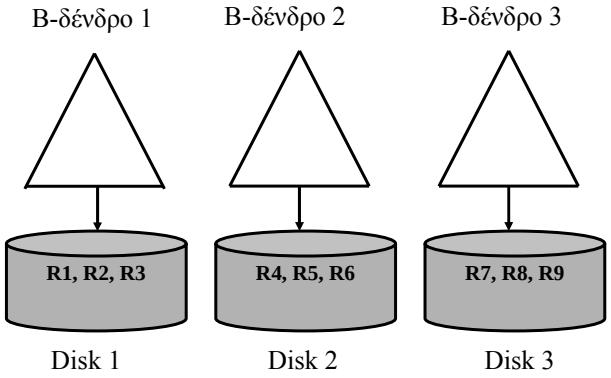
Σε συστήματα πολλών δίσκων υπάρχει η ανάγκη να διαμοιράσουμε τις δομές δεδομένων (π.χ. Β-δένδρα).

Τεχνικές που χρησιμοποιούνται:

- Διαμοιρασμός των εγγραφών
- Υπερ-σελίδες
- Διαμοιρασμός σελίδων
- Multi-Disk B-trees

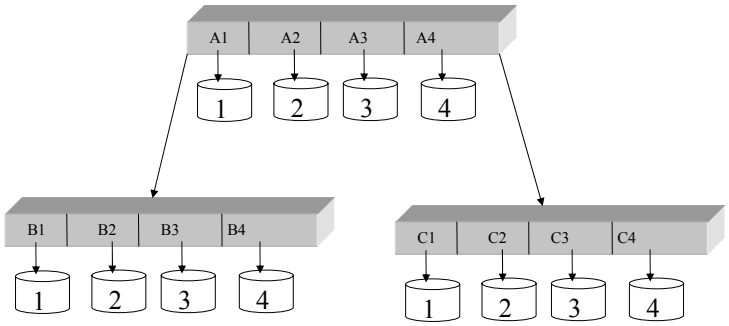
Διαμοιρασμός Εγγραφών

- Εφαρμόζεται ένας κανόνας διαμοιρασμού των εγγραφών στους δίσκους (round-robin, hashing, range partitioning).
- Για κάθε δίσκο υπάρχει **ένα ξεχωριστό** Β-δένδρο το οποίο δεικτοδοτεί τις αντίστοιχες εγγραφές.



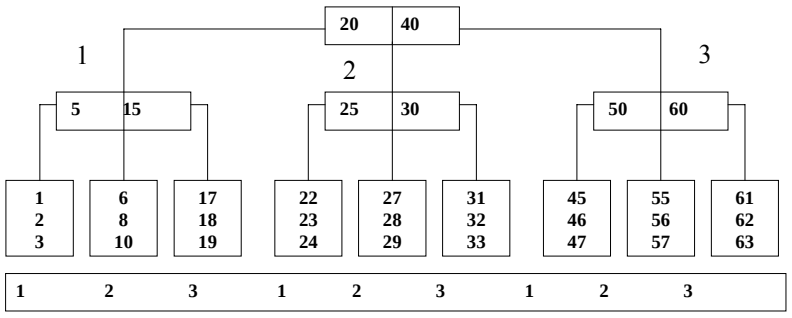
Υπερ-σελίδες

- Κάθε κόμβος του δένδρου θεωρείται ότι αποτελείται από *N* τμήματα, όπου *N* ο αριθμός των δίσκων. Κάθε τμήμα αποθηκεύεται σε ξεχωριστό δίσκο.

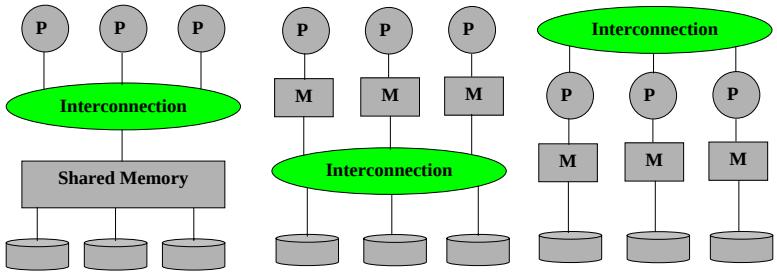


Multi-Disk B-trees

- Οι κόμβοι του δένδρου κατανέμονται στους δίσκους έτσι ώστε, δύο σελίδες που έχουν μεγάλη πιθανότητα να ζητηθούν μαζί σε ένα ερώτημα αποθηκεύονται σε διαφορετικούς δίσκους.
- *N* αριθμός δίσκων: Όταν δημιουργείται μία νέα σελίδα *P*, αποθηκεύεται στο δίσκο ο οποίος δεν περιέχει τις *N*-1 ή *N*-2 γειτονικές σελίδες με την *P*.



Παράλληλες Βάσεις Δεδομένων



shared everything shared disk shared nothing

Παράλληλοι Αλγόριθμοι

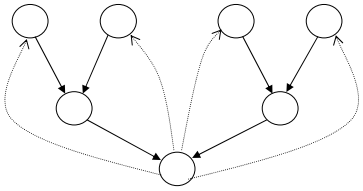
- Βασικοί στόχοι ενός παράλληλου αλγορίθμου είναι:
  - η διάσπαση του αρχικού προβλήματος σε υποπροβλήματα
  - η αντιστοίχιση υποπροβλημάτων σε επεξεργαστές.

- Σε πολλές περιπτώσεις απαιτείται να έχουμε μία εκτίμηση σχετικά με το χρόνο που απαιτείται για την επίλυση ενός υποπροβλήματος.
- Η αντιστοίχιση υποπροβλημάτων σε επεξεργαστές μπορεί να μη γίνεται στατικά αλλά δυναμικά, ανάλογα με τις ανάγκες που προκύπτουν κατά την πορεία απάντησης του ερωτήματος (static vs dynamic assignment)

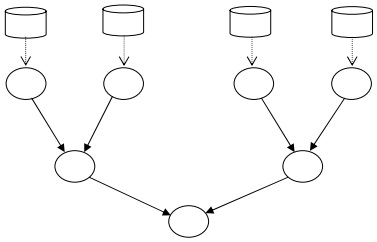
- Στη συνέχεια περιγράφουμε δύο παράλληλους αλγορίθμους οι οποίοι λύνουν το πρόβλημα της ταξινόμησης χρησιμοποιώντας *N* επεξεργαστές.
  - Parallel Merge-Sort
  - Partitioned-Based Parallel Sorting

Parallel Merge-Sort

Backend Sort: Το αρχείο προς ταξινόμηση διαμοιράζεται στους επεξεργαστές και μετά την επεξεργασία, το ταξινομημένο αρχείο επιστρέφει στον host.



Distributed Sort: αρχικά το αρχείο είναι ήδη διαμοιρασμένο στους επεξεργαστές. Στο τέλος της ταξινόμησης είτε εγγράφεται στο δίσκο είτε αποστέλλεται στον host.



Parallel Merge-Sort

- Ο αλγόριθμος θεωρεί ότι οι επεξεργαστές σχηματίζουν μία δενδρική δομή.
- Κάθε επεξεργαστής που βρίσκεται σε «φύλλο» έχει ένα δίσκο και μπορεί να εκτελέσει ταξινόμηση ανεξάρτητα από τους άλλους.
- Διακρίνουμε δύο φάσεις:
  - Φάση ταξινόμησης (sort phase)
  - Φάση συγχώνευσης (merge phase)

- Κατά τη φάση της ταξινόμησης - κάθε επεξεργαστής στο επίπεδο των φύλλων ταξινομεί τα δεδομένα που του αντιστοιχούν.
- Κατά τη φάση της συγχώνευσης - τα ταξινομημένα τμήματα που έχουν δημιουργηθεί από την προηγούμενη φάση συγχωνεύονται για να δώσουν μεγαλύτερα ταξινομημένα τμήματα.

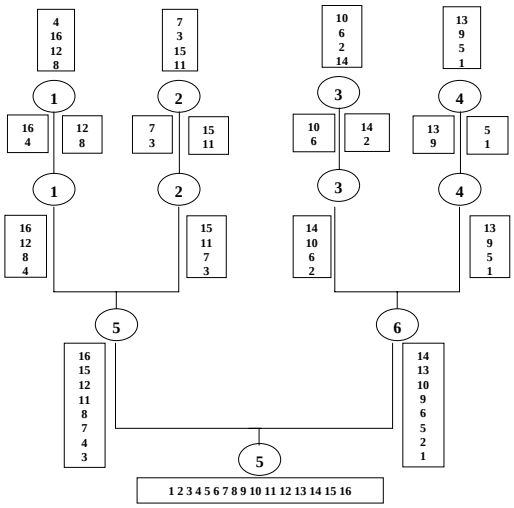
Η διαδικασία της συγχώνευσης συνεχίζεται μέχρι να ταξινομηθεί όλο το αρχείο.

Parallel Merge-Sort

Υπάρχουν δύο τρόποι με τους οποίους μπορεί να γίνει η συγχώνευση:

- Με χρήση pipelining μεταξύ των επεξεργαστών διαφορετικών επιπέδων.
- Με χρήση ταυτόχρονης συγχώνευσης στους επεξεργαστές του ίδιου επιπέδου.

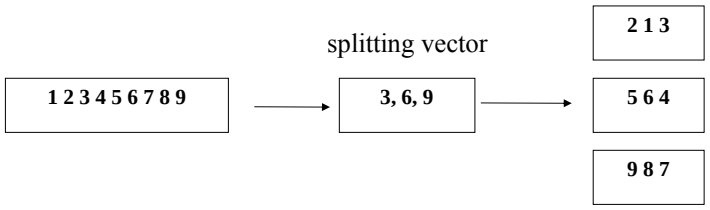
Parallel Merge-Sort



Partitioned-Based Sorting

Σε κάθε επεξεργαστή αντιστοιχεί ένα εύρος τιμών το οποίο καλείται να ταξινομήσει.

- Η εύρεση του εύρους για κάθε επεξεργαστή καθορίζει και την απόδοση του αλγορίθμου.
- Η μέθοδος προσπαθεί για ομοιόμορφη κατανομή σε όλους τους επεξεργαστές, ώστε να μην υπάρχει bottleneck.



Partitioned-Based Sorting

N επεξεργαστές *P*1 … *PN*, και κάθε ένας έχει ένα σύνολο δεδομένων *D*i.

- Κάθε *P*i πραγματοποιεί τυχαία δειγματοληψία στο *D*i και αποστέλλει το δείγμα στον συντονιστή (coordinating processor).
- Ο συντονιστής με βάση τα δείγματα καθορίζει ένα splitting vector το οποίο καθορίζει το εύρος τιμών για κάθε επεξεργαστή.
- Το splitting vector αποστέλλεται στους επεξεργαστές και ο κάθε ένας διανέμει τα δεδομένα στους αντίστοιχους επεξεργαστές.
- Κάθε επεξεργαστής ταξινομεί τα δεδομένα του και εγγράφει τα αποτελέσματα στον τοπικό δίσκο.

Η δειγματοληψία κοστίζει.

Μεγάλο δείγμα καλή προσέγγιση των δεδομένων αλλά μεγάλο κόστος υπολογισμού και μετάδοσης.

Μικρό δείγμα μικρό κόστος υπολογισμού και μετάδοσης αλλά κίνδυνος για κακή εκτίμηση.

Ανάκτηση Πληροφοριών και
Παράλληλη Επεξεργασία

Προσεγγίσεις

(Α) Σχεδιασμός **νέων** τεχνικών ΑΠ που να είναι κατάλληλες για παράλληλη επεξεργασία

(Β) **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:

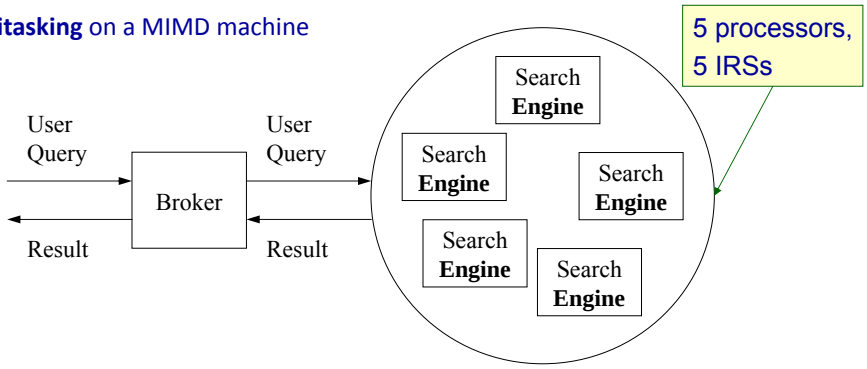
- Parallel multitasking;

Κάθε επεξεργαστής εκτελεί ένα ξεχωριστό έργο (task) ανεξάρτητα από τους άλλους επεξεργαστές.

- Partitioned parallel processing.

MIMD Architectures: **Parallel Multitasking**

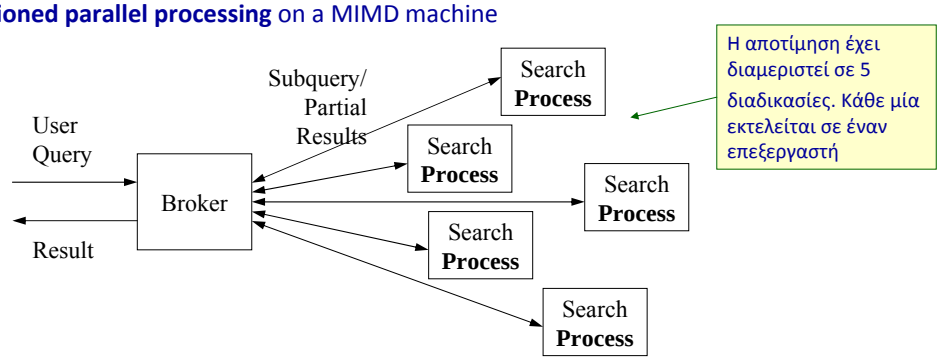
Parallel multitasking on a MIMD machine



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

MIMD Architectures: **Partitioned Parallel Processing**

Partitioned parallel processing on a MIMD machine

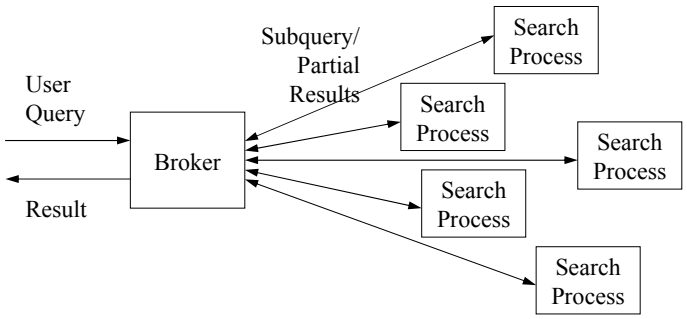


εδώ ο χρόνος αποτίμησης μιας επερώτησης είναι μικρότερος

- 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;

MIMD Architectures: Partitioned Parallel Processing
Πώς να διαμερίσουμε τα δεδομένα σε P επεξεργαστές;

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	$w_{1,N}$	$w_{2,N}$	. . .	$w_{i,N}$	. . .	$w_{t,N}$

MIMD Architectures: Partitioned Parallel Processing
Πώς να διαμερίσουμε τα δεδομένα σε P επεξεργαστές;

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

MIMD Architectures: Partitioned Parallel Processing

Πώς να διαμερίσουμε τα δεδομένα σε P επεξεργαστές;

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

MIMD Architectures: Partitioned Parallel Processing

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

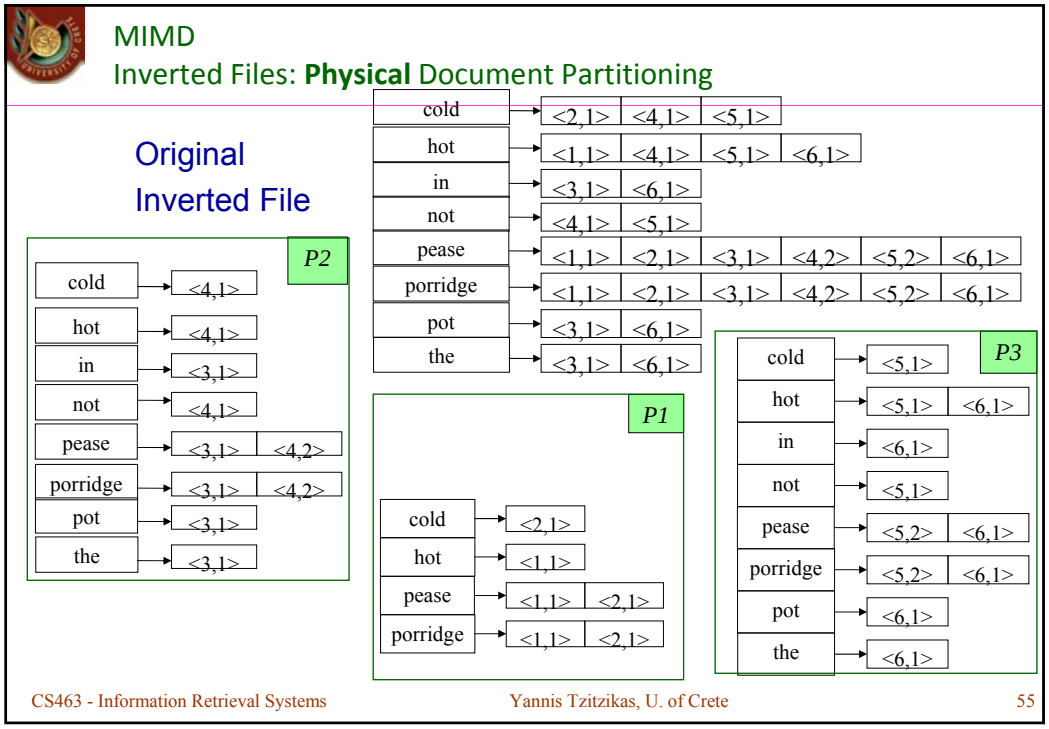
cold	<2,1>
hot	<1,1>
pease	<1,1> <2,1>
porridge	<1,1> <2,1>

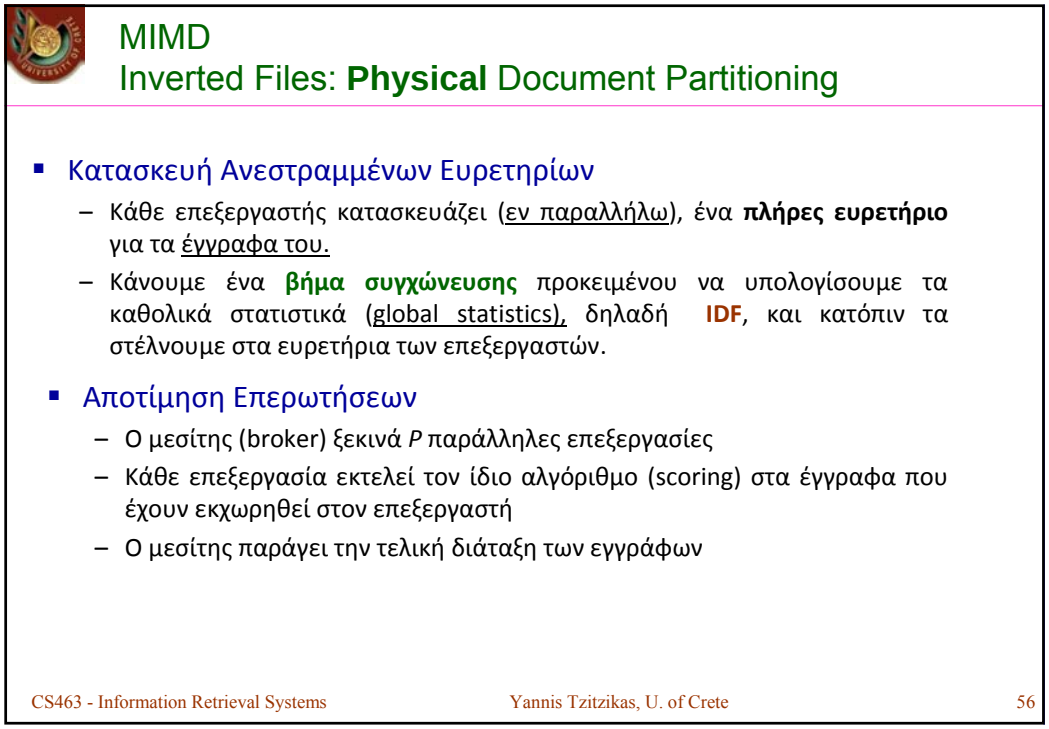
P2

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>

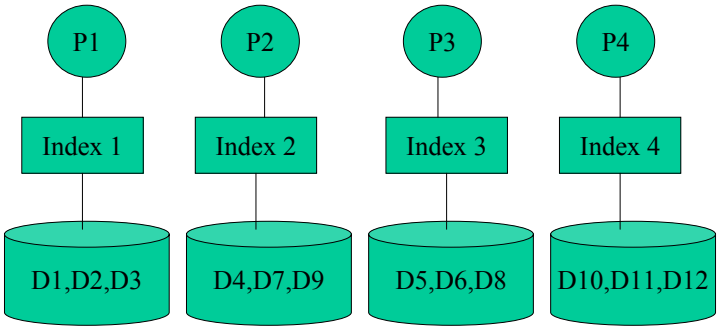
P3

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
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>
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>
P1
cold <2,1>
hot <1,1>
pease <1,1> <2,1>
porridge <1,1> <2,1>
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>
CS463 - Information Retrieval Systems
Yannis Tzitzikas, U. of Crete
55


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
CS463 - Information Retrieval Systems
Yannis Tzitzikas, U. of Crete
56

Φυσικός Διαμοιρασμός



MIMD Architectures: Partitioned Parallel Processing

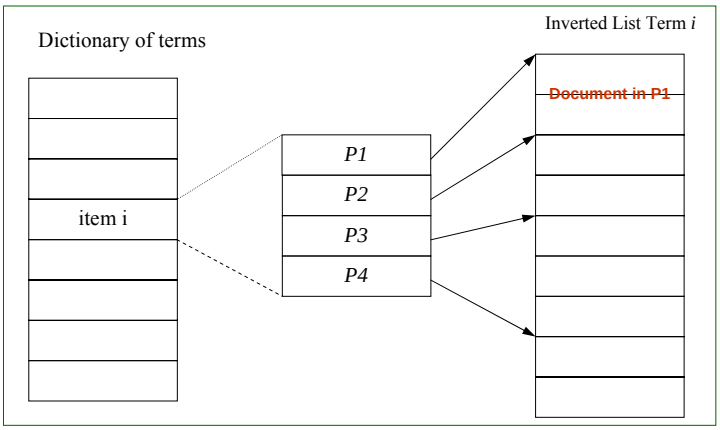
Document and Term Partitioning **for Inverted Files**

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

MIMD
Inverted Files: **Logical** Document Partitioning

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

Κάθε θέση στο λεξικό περιέχει ***P* pointers** (*P*: αριθμός επεξεργαστών). Ο *j*-οστός pointer δεικτοδοτεί τα έγγραφα που βρίσκονται στις λίστες εμφάνισης και πρέπει να επεξεργαστούν από τον επεξεργαστή *Pj*.

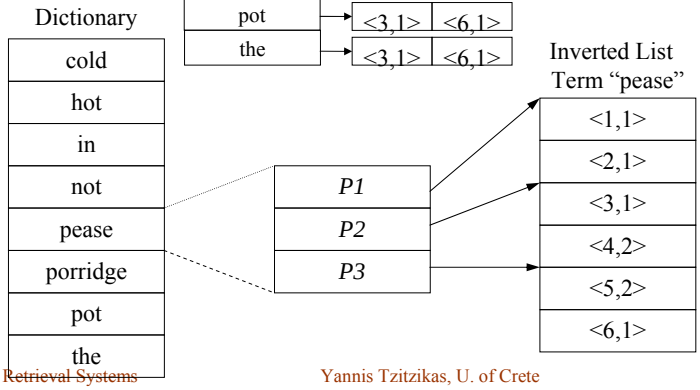


Logical Document Partitioning

Original Inverted File

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>

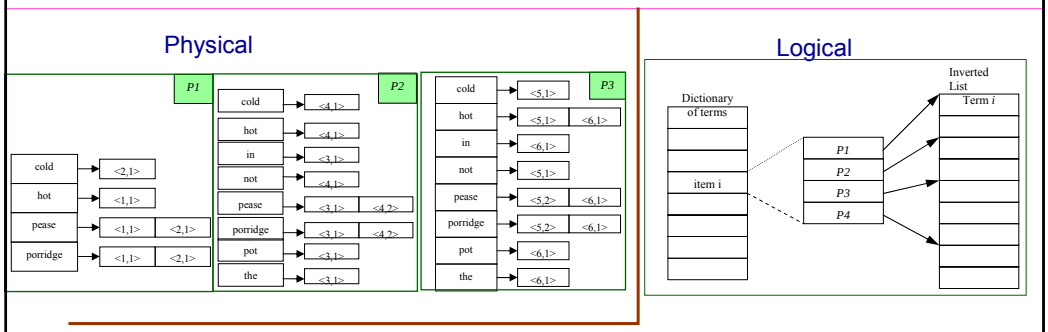
Extended Dictionary



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 προηγούμενη διάλεξη: 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)).

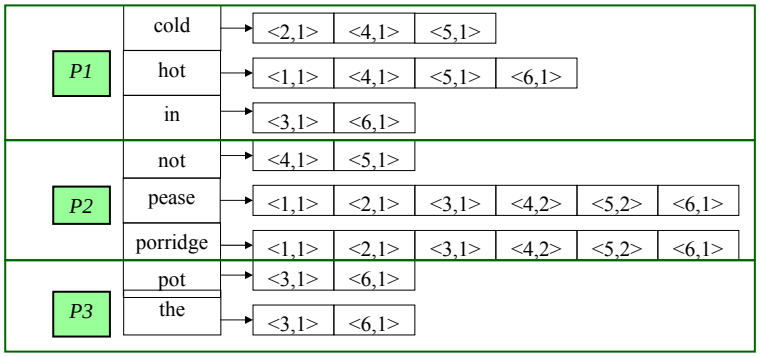
MIMD Architectures: Partitioned Parallel Processing

Document and Term Partitioning **for Inverted Files**

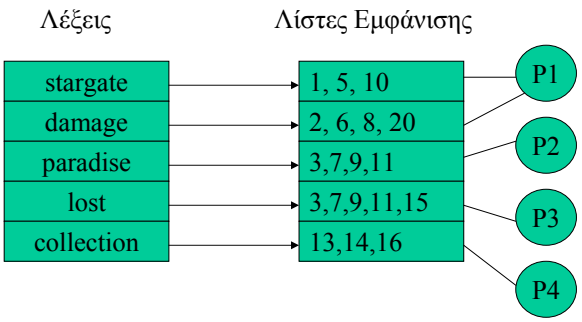
- Document Partitioning
  - **Physical** Document Partitioning
  - **Logical** Document Partitioning
- **Term Partitioning**

MIMD
Inverted Files:**Term** Partitioning

Term Partitioning



Διαμοιρασμός Όρων



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 **συνδυάζοντας τα μερικά**, και παράγει την τελική απάντηση

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

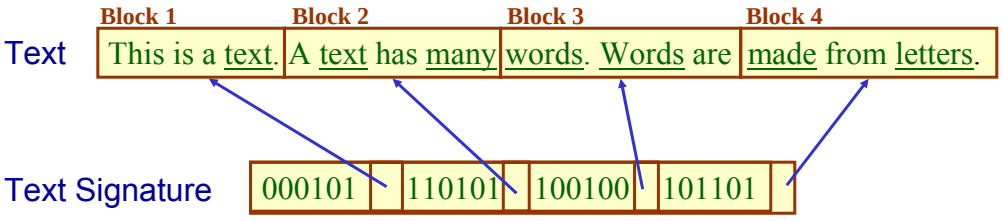
- Οργάνωση ευρετηρίου σε μία MIMD μηχανή:
  - Document partitioning (physical or logical);
  - Term partitioning.
- Document partitioning
  - simpler inverted index construction and maintenance than term partitioning;
  - performs better when term distributions in the documents and queries are more skewed
- 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