Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. και Μηχανικών Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Παράλληλος προγραμματισμός:
Σχεδίαση και υλοποίηση παράλληλων προγραμμάτων

Συστήματα Παράλληλης Επεξεργασίας

9ο Εξάμηνο

National Technical University of Athens
CSLab

Σύνοψη παρουσίασης

- Παράλληλες υπολογιστικές πλατφόρμες
  - PRAM: Η ιδανική παράλληλη πλατφόρμα
  - Η ταξινόμηση του Flynn
  - Συστήματα κοινής μνήμης
  - Συστήματα κατανεμημένης μνήμης
- Ανάλυση παράλληλων προγραμμάτων
  - Μετρικές αξιολόγησης επίδοσης
  - Ο νόμος του Amdahl
  - Μοντελοποίηση παράλληλων προγραμμάτων
- **Οργάνωση πρόσβασης στα δεδομένα**
- **Παράλληλα προγραμματιστικά μοντέλα**
  - **Κοινού χώρου διευθύνσεων**
  - **Ανταλλαγής μηνυμάτων**

Σύνοψη παρουσίασης

- Σχεδίαση και υλοποίηση παράλληλων προγραμμάτων
  - Παραλληλισμός σε επίπεδο εργασίας (task parallelism)
  - Παραλληλοποίηση σε επίπεδο δεδομένων (data parallelism)
  - Παραλληλοποίηση σε επίπεδο βρόχου (loop parallelism)
  - Παραλληλοποίηση σε επίπεδο συνάρτησης (function parallelism)
- Γλώσσες και εργαλεία
  - POSIX threads, MPI, OpenMP, Cilk, Cuda, Γλώσσες PGAS
- Αλληλεπίδραση με το υλικό
  - Συστήματα κοινής μνήμης
  - Συστήματα κατανεμημένης μνήμης και υβριδικά

Σχεδιασμός και υλοποίηση παράλληλων προγραμμάτων

- **Στόχος:** Η μετατροπή ενός σειριακού αλγορίθμου σε παράλληλο
  - Κατανομή υπολογισμών σε υπολογιστικές εργασίες (tasks)
  - Ορισμός ορθής σειράς εκτέλεσης (χρονοδρομολόγηση)
  - Διαμοιρασμός των δεδομένων - Οργάνωση πρόσβασης στα δεδομένα (συγχρονισμός / επικοινωνία)
  - Ανάθεση εργασιών (απεικόνιση) σε οντότητες εκτέλεσης (processes, threads)
- Δεν υπάρχει αυστηρά ορισμένη μεθοδολογία για το σχεδιασμό και την υλοποίηση παράλληλων προγραμμάτων
- “It’s more art than science”
- Ιδανικά θα θέλαμε ο σχεδιασμός και η υλοποίηση να λαμβάνουν χώρα ανεξάρτητα. Στην πράξη υπάρχει αλληλεπίδραση, π.χ. τον σχεδιασμό επηρεάζουν:
  - Η αρχιτεκτονική της πλατφόρμας εκτέλεσης
  - Τα υποστηριζόμενα προγραμματιστικά μοντέλα
  - Οι δυνατότητες του περιβάλλοντος υλοποίησης

Σχεδιασμός και υλοποίηση παράλληλων προγραμμάτων

- Μετρικές αξιολόγησης (τόσο για το σχεδιασμό όσο και για την υλοποίηση):
  - **Επίδοση** του παραγόμενου κώδικα
    - Επιτάχυνση
    - Αποδοτικότητα
    - Κλιμακωσιμότητα
  - **Παραγωγικότητα** (code productivity)
    - Χαμηλός χρόνος υλοποίησης
    - Μεταφερσιμότητα (portability)
    - Ευκολία στη συντήρηση (maintainability)
  - **Κατανάλωση ενέργειας / ισχύος**
    - Αν ένας παράλληλος αλγόριθμος έχει περισσότερη δουλειά (work σε PRAM ορολογία) επιφέρει μεγαλύτερη κατανάλωση ενέργειας
    - Η μη γραμμική κλιμακωσιμότητα (που είναι και η τυπική περίπτωση) έχει ενεργειακό κόστος.
      - Στην τυπική περίπτωση η πιο ενεργειακά αποδοτική εκτέλεση είναι η σειριακή!

Δομή της παρουσίασης

- Ορισμοί
  - Εργασία (task)
  - Εργάτης (process ή thread)
  - Επεξεργαστής (processor)
  - Δεδομένα (data)
- Οργάνωση πρόσβασης στα δεδομένα
  - Μοιραζόμενα
  - Κατανεμημένα
  - Αντιγραμμένα
- Προγραμματιστικά μοντέλα
  - Κοινού χώρου διευθύνσεων
  - Ανταλλαγής μηνυμάτων

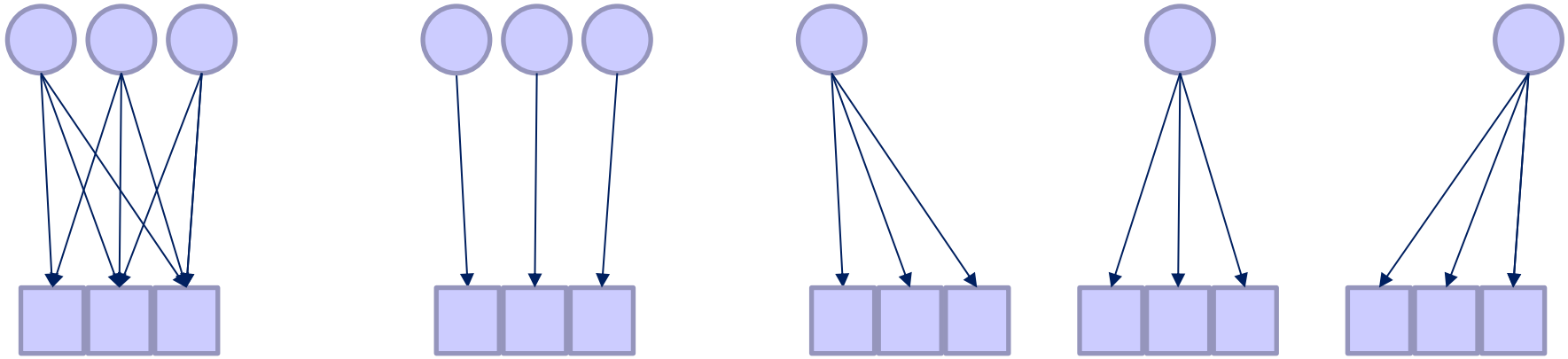
Ορισμοί

- Εργασία (**task**): Ένα υποσύνολο της συνολικής δουλειάς που πρέπει να κάνει ένας αλγόριθμος ή υπολογιστικός πυρήνας, π.χ. ένας πολλαπλασιασμός ή ένα σύνολο πράξεων
  - Το ορίζει ο σχεδιαστής / προγραμματιστής
  - Μπορείτε να το σκέφτεστε σαν ένα υποσύνολο του κώδικα με τα σχετικά δεδομένα (μοιάζει με ένα function)
- Εργάτης, οντότητα εκτέλεσης (**process or thread**): Η μονάδα λογισμικού (συνήθως σε όρους Λειτουργικού Συστήματος) που εκτελεί μία ή περισσότερες εργασίες
- Επεξεργαστής (**processor, CPU, CPU core**): Η μονάδα υλικού στην οποία εκτελείται το process/thread.
- Δεδομένα (**data**): Τα δεδομένα στα οποία επενεργεί μία εργασία
- Αναλογία:
  - Task – process – data- CPU core
  - Εργασία – εργάτης – υλικά - εργαστήριο

Οργάνωση πρόσβασης στα δεδομένα

- **Χαρακτηρισμός δεδομένων (data access patterns)**
- Τα δεδομένα του προβλήματος μπορούν να είναι:
  - *Μοιραζόμενα (shared data)*:
    - Τα δεδομένα είναι προσβάσιμα από όλα τα (ή από πολλά) tasks
    - Μπορεί να υποστηριχθεί μόνο από προγραμματιστικά μοντέλα κοινού χώρου διευθύνσεων (βλ. συνέχεια)
    - Αποτελεί έναν πολύ βολικό τρόπο πρόσβασης
    - Χρειάζεται ιδιαίτερη προσοχή για τον εντοπισμό race conditions (**ανάγκη συγχρονισμού**)
  - *Κατανεμημένα* (*distributed data*):
    - Τα δεδομένα κατανέμονται ανάμεσα στα tasks
    - Αποτελεί τη βασική προσέγγιση στα προγραμματιστικά μοντέλα ανταλλαγής μηνυμάτων (βλ. συνέχεια)
    - Επιβάλλει επιπλέον προγραμματιστικό κόστος
    - Οδηγεί σε **ανάγκη για επικοινωνία**
  - *Αντιγραμμένα* (*replicated data*):
    - Τα δεδομένα αντιγράφονται ανάμεσα στα tasks
    - Κατάλληλο σχήμα για αντιγραφή μικρών read-only δομών δεδομένων
    - Απαιτεί σχήμα ενημέρωσης των αντιγράφων στην περίπτωση που γίνονται εγγραφές

Οργάνωση πρόσβασης στα δεδομένα



Μοιραζόμενα (shared)

Κατανεμημένα (distributed)

Αντιγραμμένα (replicated)

task

data element

Οργάνωση πρόσβασης στα δεδομένα

	READ	WRITE
Μοιραζόμενα (shared)	Η απλούστερη περίπτωση, ο προγραμματιστής δεν χρειάζεται να λάβει κάποια ειδική μέριμνα	Χρειάζεται προσοχή για τον εντοπισμό πιθανών race conditions ώστε να προστεθούν κατάλληλες δομές συγχρονισμού (π.χ. locks)
Κατανεμημένα (distributed)	Χρειάζεται απόφαση για τον τρόπο κατανομής των δεδομένων. Αν παραπάνω από μία εργασίες χρειάζονται τα δεδομένα απαιτείται επικοινωνία, οπότε και είναι σκόπιμη η εξέταση του ενδεχομένου αντιγραφής των δεδομένων εξαρχής.	Πιθανόν απαιτείται επικοινωνία για τα δεδομένα που γράφει μία εργασία και τα χρειάζεται μία άλλη (βλ. εξίσωση θερμότητας, game of life, κλπ).
Αντιγραμμένα (replicated)	Αρκετά απλή και βολική περίπτωση, οδηγεί όμως σε σπατάλη χώρου μνήμης (και άρα μπορεί να την εξαντλήσει)	Απαιτείται σχήμα που θα εξάγει συνεπές αποτέλεσμα από τις τιμές των αντιγραμμένων δεδομένων (βλ. reduction σε counter).

Οργάνωση πρόσβασης στα δεδομένα

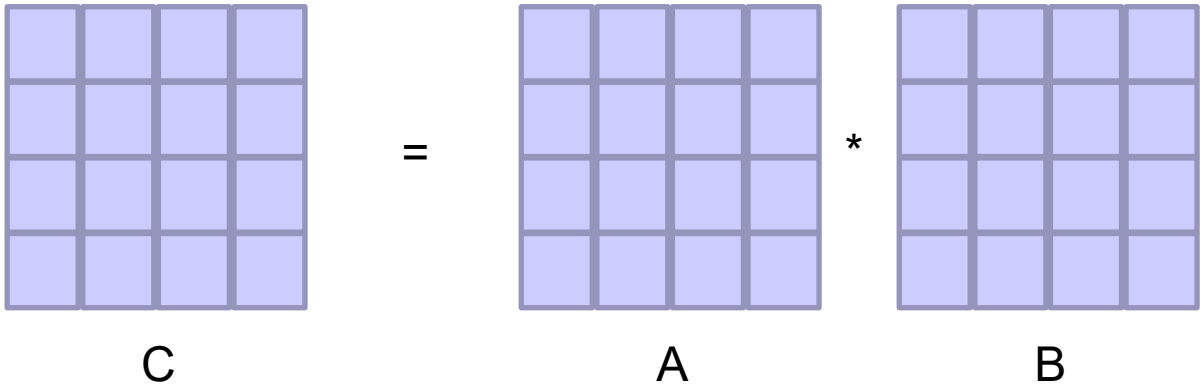
- Ο χαρακτηρισμός των δεδομένων δημιουργεί ανάγκες για **επικοινωνία** και **συγχρονισμό**
- **Επικοινωνία:**
  - Κατανεμημένα δεδομένα ή αντιγραμμένα δεδομένα
  - 1 task χρειάζεται να διαβάσει δεδομένα που κατέχει ένα άλλο task
    - Είτε εξαιτίας του γράφου εξαρτήσεων (βλ συνέχεια, παράχθηκαν κατά την εκτέλεση σε άλλο task)
    - Είτε επειδή βρίσκονται αποθηκευμένα σε άλλο task (π.χ. από την αρχική κατανομή)
  - Σημείο-προς-σημείο και συλλογική
  - Καθορισμός ποια δεδομένα πρέπει να αποσταλούν, σε ποιες εργασίες και πότε

Οργάνωση πρόσβασης στα δεδομένα

- Ο χαρακτηρισμός των δεδομένων δημιουργεί ανάγκες για **επικοινωνία** και **συγχρονισμό**
- **Συγχρονισμός:**
  - Απαιτείται είτε λόγω του γράφου εξαρτήσεων (βλ. συνέχεια - **σειριοποίηση**) είτε λόγω **καταστάσεων συναγωνισμού** (race conditions)
  - **Σειριοποίηση**:
    - **Μηχανισμοί:** Barriers, condition variables, semaphores
    - Τους εισάγει ο προγραμματιστής απευθείας ή το αναλαμβάνει το σύστημα χρόνου εκτέλεσης (runtime system).
  - Εντοπισμός **καταστάσεων συναγωνισμού** και εισαγωγή κατάλληλου σχήματος ελέγχου ταυτόχρονης πρόσβασης (concurrency control)
    - Αναγκαία συνθήκη για δημιουργία κατάστασης συναγωνισμού: εγγραφές σε μοιραζόμενα δεδομένα
    - Ικανή συνθήκη για δημιουργία κατάστασης συναγωνισμού: προκύπτει από την υλοποίηση (π.χ. ανάγκη για ατομική εκτέλεση >1 εντολών)
  - **Αμοιβαίος αποκλεισμός:**
    - **Μηχανισμοί:** Critical section, locks, readers-writers locks, atomic operations

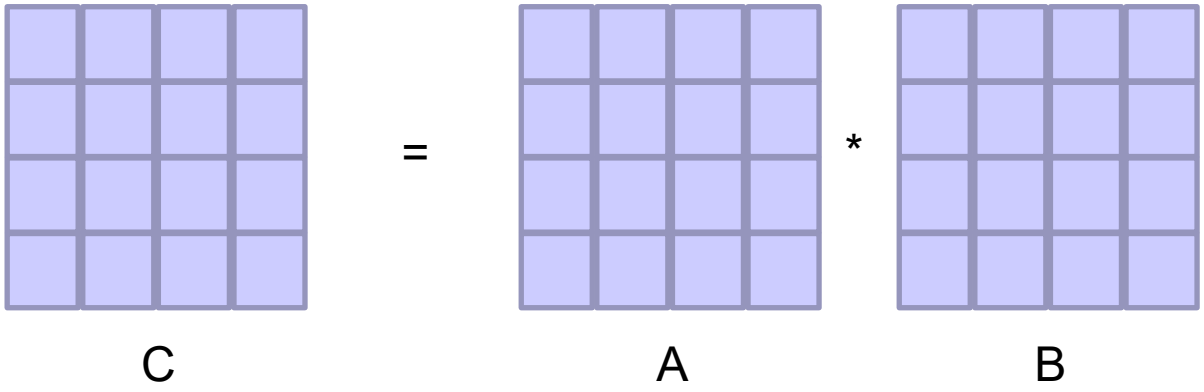
Οργάνωση πρόσβασης στα δεδομένα

- **Παράδειγμα:** πολλαπλασιασμός πινάκων



Οργάνωση πρόσβασης στα δεδομένα

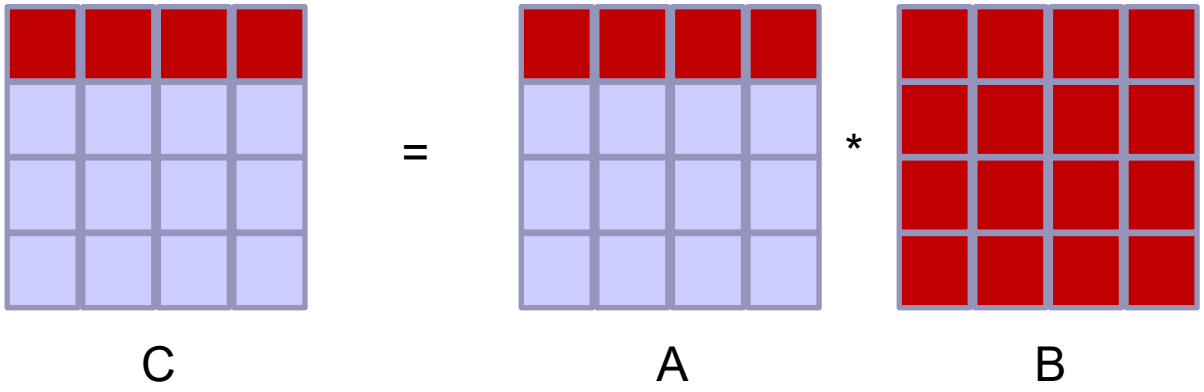
- **Παράδειγμα:** πολλαπλασιασμός πινάκων



- 1 task = υπολογισμός του C_{ij}
- Έστω σύστημα που υποστηρίζει μοιραζόμενα δεδομένα
  - C, A, B μοιραζόμενα
  - Δεν υπάρχει race condition στην εγγραφή του C
  - Όλα καλά ☺

Οργάνωση πρόσβασης στα δεδομένα

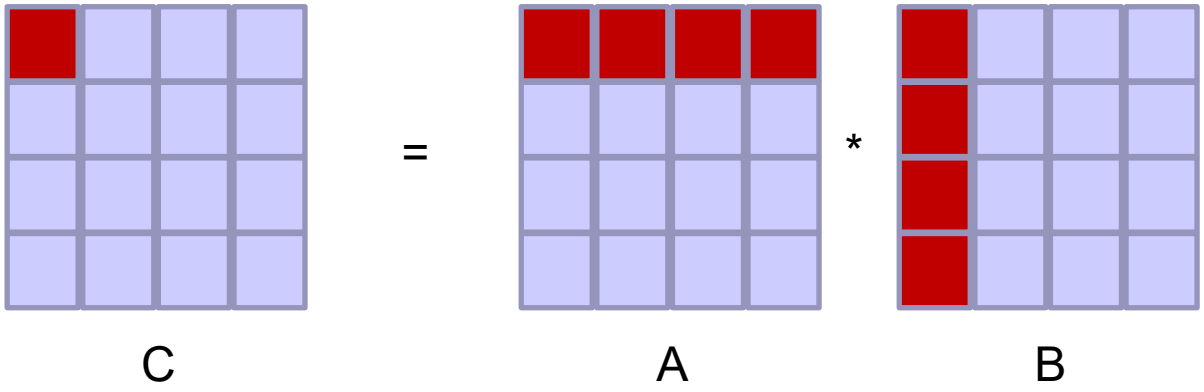
- **Παράδειγμα:** πολλαπλασιασμός πινάκων



- 1 task = υπολογισμός μιας γραμμής του C
- Έστω σύστημα που ΔΕΝ υποστηρίζει μοιραζόμενα δεδομένα
  - C, A distributed κατά γραμμές
  - B αντιγραμμένο
  - Όλα καλά, εκτός αν: α) ο B δεν χωράει στη μνήμη και β) το task είναι πολύ μεγάλο (π.χ. έχουμε περισσότερους επεξεργαστές από tasks)

Οργάνωση πρόσβασης στα δεδομένα

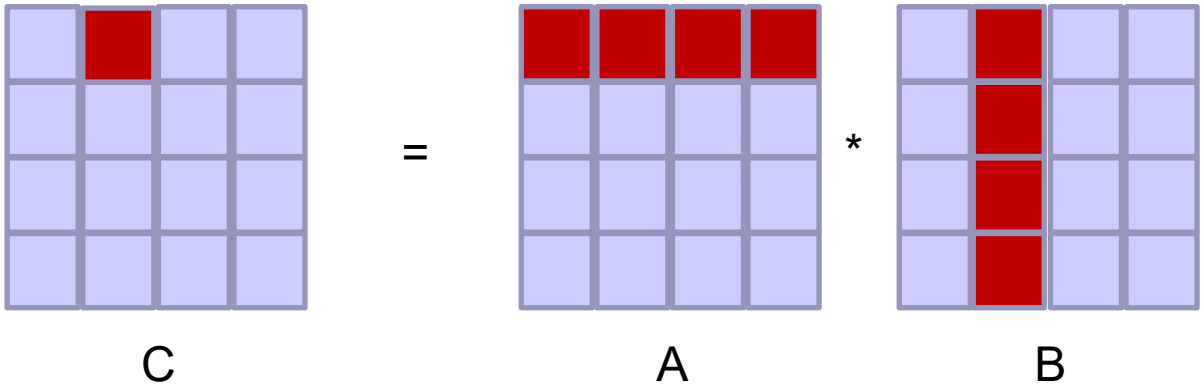
- **Παράδειγμα:** πολλαπλασιασμός πινάκων



- 1 task = υπολογισμός του C_{ij}
- Έστω σύστημα που ΔΕΝ υποστηρίζει μοιραζόμενα δεδομένα
  - C distributed κατά στοιχείο
  - A distributed και copied ανά γραμμή, B distributed και copied ανά στήλη
  - Όλα καλά, εκτός αν οι γραμμές/στήλες A και B δεν χωρούν στη μνήμη

Οργάνωση πρόσβασης στα δεδομένα

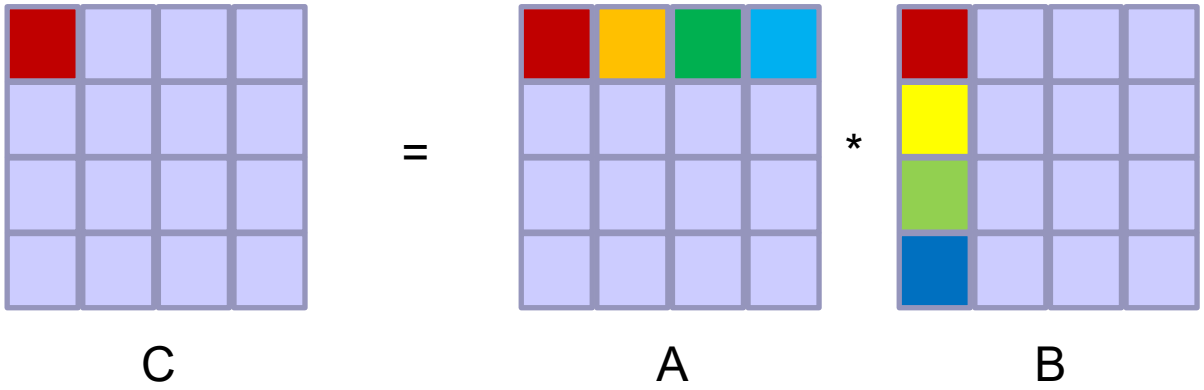
- **Παράδειγμα:** πολλαπλασιασμός πινάκων



- 1 task = υπολογισμός του C_{ij}
- Έστω σύστημα που ΔΕΝ υποστηρίζει μοιραζόμενα δεδομένα
  - C distributed κατά στοιχείο
  - A distributed και copied ανά γραμμή, B distributed και copied ανά στήλη
  - Όλα καλά, εκτός αν οι γραμμές/στήλες A και B δεν χωρούν στη μνήμη

Οργάνωση πρόσβασης στα δεδομένα

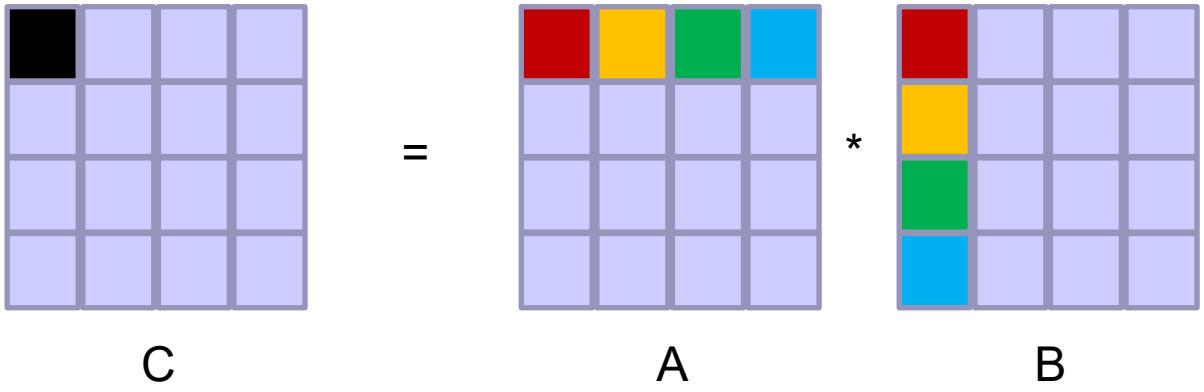
- **Παράδειγμα:** πολλαπλασιασμός πινάκων



- 1 task = υπολογισμός του C_{ij}
- Έστω σύστημα που ΔΕΝ υποστηρίζει μοιραζόμενα δεδομένα
  - Η πιο γενική περίπτωση:
    - C, A, B distributed κατά στοιχείο
  - Χρειάζεται επικοινωνία για ανάγνωση του στοιχείου από την εργασία που το κατέχει

Οργάνωση πρόσβασης στα δεδομένα

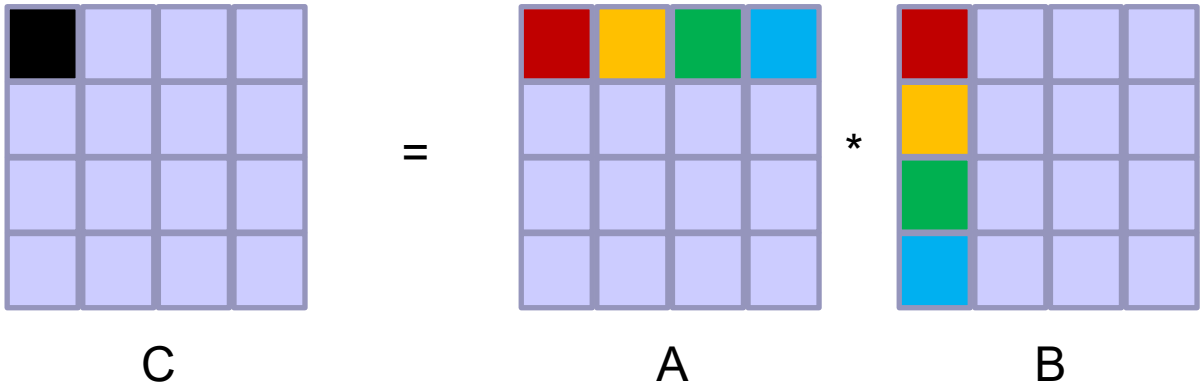
- **Παράδειγμα:** πολλαπλασιασμός πινάκων



- 1 task = υπολογισμός του $A_{ik}*B_{kj}$
- Έστω σύστημα που υποστηρίζει μοιραζόμενα δεδομένα
  - C, A, B μοιραζόμενα
  - Χρειάζεται συγχρονισμός για το σωστό υπολογισμό του C_{ij}

Οργάνωση πρόσβασης στα δεδομένα

- **Παράδειγμα:** πολλαπλασιασμός πινάκων



- 1 task = υπολογισμός του $A_{ik}*B_{kj}$
- Έστω σύστημα που υποστηρίζει μοιραζόμενα δεδομένα
  - C αντιγραμμένο, A, B μοιραζόμενα
  - Χρειάζεται συνδυασμός αποτελεσμάτων (reduction) για το σωστό υπολογισμό του C_{ij}

Προγραμματιστικά μοντέλα

- Κοινού χώρου διευθύνσεων
  - Υποστηρίζει κοινά δεδομένα ανάμεσα στα νήματα
  - Επιταχύνει τον προγραμματισμό
  - Μπορεί να οδηγήσει σε δύσκολα ανιχνεύσιμα race conditions
  - Δεν μπορεί να υλοποιηθεί αποδοτικά σε πλατφόρμες που δεν παρέχουν πρόσβαση σε κοινή μνήμη στο υλικό
    - Κατάλληλο για συστήματα κοινής μνήμης
    - Περιορισμένος συνολικός αριθμός νημάτων
- Ανταλλαγής μηνυμάτων
  - Κάθε διεργασία (νήμα) βλέπει μόνο τα δικά της δεδομένα
  - Τα δεδομένα μπορεί να είναι μόνο distributed ή replicated
  - Οδηγεί σε χρονοβόρο προγραμματισμό ακόμα και για απλά προγράμματα (fragmented κώδικας)
  - Μπορεί να υποστηριχθεί από πολύ μεγάλης κλίμακας συστήματα
- Υβριδικό: συνδυασμός των παραπάνω 2

Αρχιτεκτονικές και Προγραμματιστικά Μοντέλα από την οπτική του προγραμματιστή

		Αρχιτεκτονική	
		Κοινής μνήμης (shared memory)	Κατανεμημένης μνήμης (distributed memory)
Προγραμματιστικό μοντέλο	Κοινός χώρος διευθύνσεων (shared address space)	+ Ευκολία υλοποίησης + Προγραμματιστική ευκολία + Υψηλή επίδοση	+ Προγραμματιστική ευκολία - Δυσκολία υλοποίησης - Χαμηλή επίδοση
	Ανταλλαγή μηνυμάτων (message-passing)	+ Ευκολία υλοποίησης + Υψηλή επίδοση - Προγραμματιστική δυσκολία	+ Ευκολία υλοποίησης + Υψηλή επίδοση - Προγραμματιστική δυσκολία

Ερωτήσεις;