

# Memoria condivisa distribuita (DSM)

# Sommario

- Introduzione e motivazioni
- Modelli di DSM
- Problema della coerenza

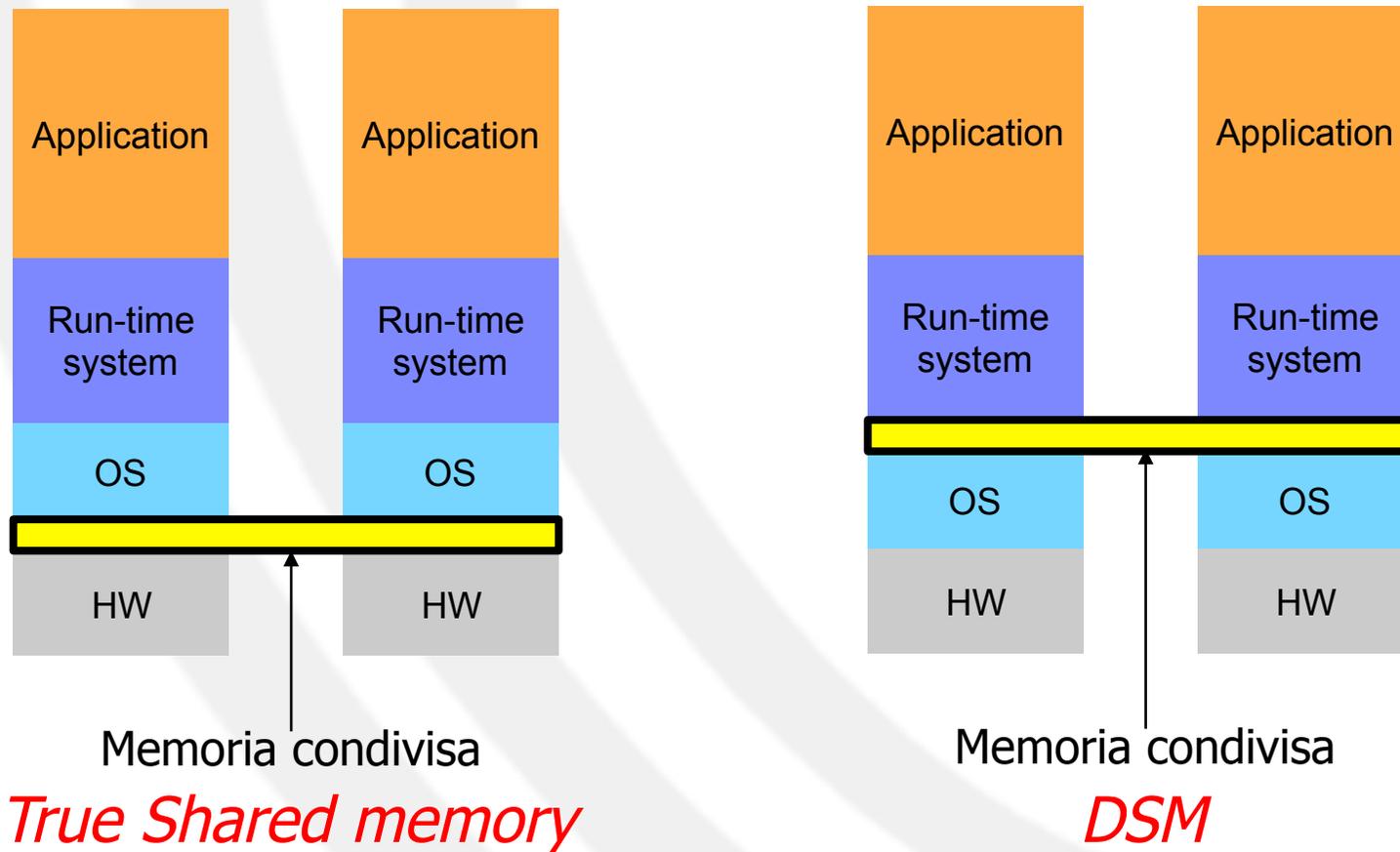
# Introduzione

- Multicomputer
  - Facili da costruire
  - Difficili da programmare
- Multiprocessor
  - Difficili da costruire
  - Facili da programmare
- Vorremmo qualcosa facile da costruire e facile da programmare
  - Distributed Shared Memory

# Introduzione

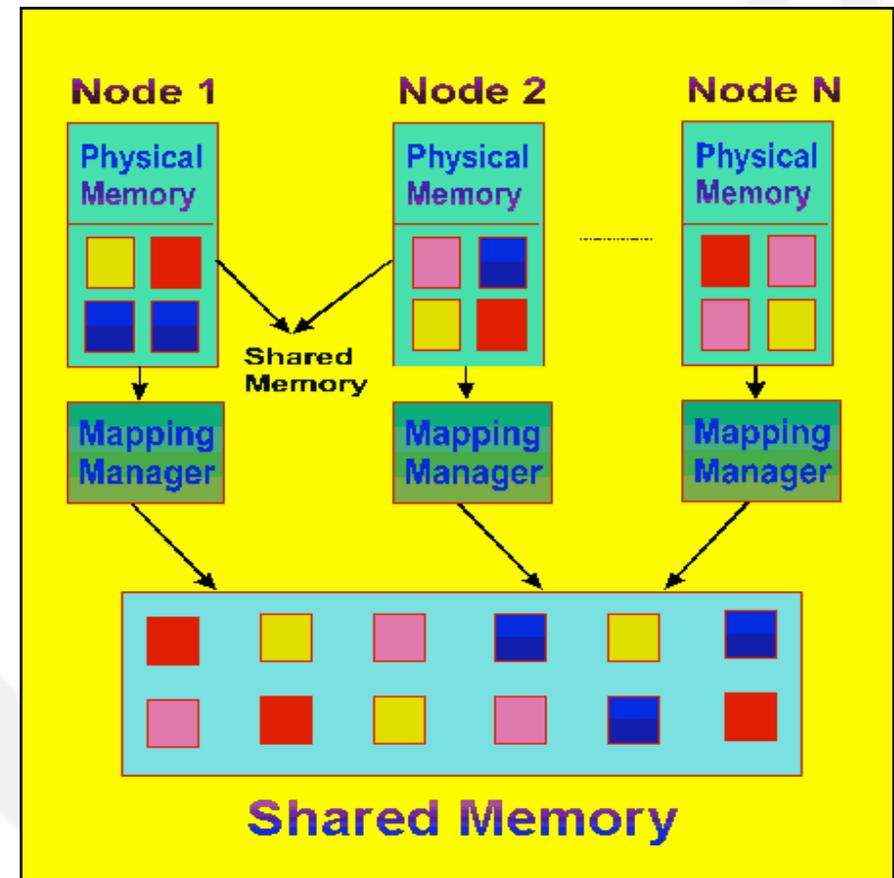
- Distributed shared memory [Li 86, Li 89]
  - Realizzazione dello schema a memoria condivisa in ambiente distribuito
  - Ottenibile con “modeste” penalità nelle prestazioni
    - In particolare in sistemi debolmente accoppiati
- Motivazioni
  - Astrazione più semplice dello scambio di messaggi
  - Sfrutta la località dei riferimenti a memoria
  - Scalabile (no bus comune)
- DSM vs. scambio di messaggi
  - Una buona implementazione di message-passing è (in genere) più efficiente di DSM!

# Introduzione



# Schema concettuale

- Spazio di indirizzamento suddiviso in pagine
- Ogni host possiede un certo numero di pagine locali
- Accessi a pagine remote (ma apparentemente locali) gestite dal *Mapping Manager*
  - Strato SW che mappa accessi allo spazio condiviso in accessi alla memoria fisica
- Simile al concetto di memoria virtuale nei S.O. tradizionali



## DSM: problemi

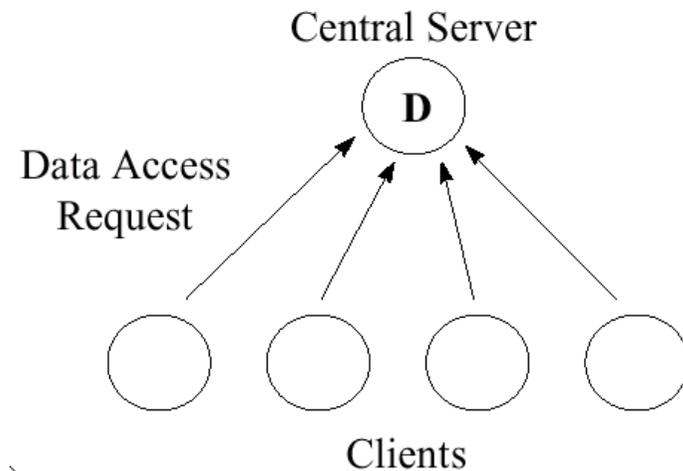
- Performance basse
- Tutti gli studi su DSM mirano a
  - Diminuire il traffico di rete
  - Ridurre la latenza tra il momento in cui si richiede una pagina e il momento in cui la pagina diventa disponibile

# DSM – Approcci

- Due dimensioni:
  - Uso di migrazione di dati
  - Uso di repliche di dati
- 4 possibilità:
  - Server centrale “di memoria” (central server)
  - Migrazione dei dati (data migration)
  - Replica in lettura (read replication)
  - Replica completa (full replication)

# Server di memoria

- Un server centrale mantiene tutte le pagine dati
  - Ritorna i dati in caso di lettura
  - Effettua le modifiche in caso di scrittura
- Importante evitare scritture duplicate
  - Ottenibile “numerando” le richieste
- I compiti del server possono essere suddivisi su più server tramite partizionamento statico



# Server di memoria

- **Vantaggi**
  - Facile da implementare
  - Meccanismi di sincronizzazione relativamente consolidati
- **Svantaggi**
  - La maggior parte dei riferimenti a dati sono remoti
  - Soggetto a colli di bottiglia

# Migrazione dei dati

- Concetto
  - I dati risiedono dove vengono usati più spesso
  - Quando viene fatto riferimento a dati “off-site”, questi vengono migrati presso il sito richiedente
    - Granularità tipica: blocco o pagina
- Un solo processo alla volta può accedere ad un dato condiviso
  - Protocollo “lettore singolo/scrittore singolo”

# Migrazione dei dati

| Client   | Host remoto                            |
|--|--|
| Se il blocco non è locale, determina la sua locazione e invia la richiesta |  |
|  | Riceve la richiesta ed invia il blocco |
| Riceve la risposta ed accede ai dati                                       |  |

- Come individuare i blocchi?
  - Processo server tiene traccia della locazione dei dati (directory)
  - Broadcast
  - Informazioni locali presso i vari nodi

# Migrazione dei dati

- Vantaggi
  - In caso di alta località di riferimento, la maggior parte degli accessi è locale
    - Infatti trasferisco blocchi/pagine di memoria, non singoli dati
  - Sincronizzazione relativamente facile
- Svantaggi
  - Rischio di thrashing inter-nodo
    - Blocchi migrati tra nodi diversi
    - Soluzione tramite timeout (garantire un minimo numero di accessi locali prima della migrazione)
  - Possibili colli di bottiglia nei server

# Approcci basati su replicazione

- Nell'approccio basato su migrazione solo le thread di un singolo host possono accedere allo stesso blocco
  - Copia unica di un blocco
- Replicare i dati permette di avere più accessi simultanei sui dati stessi, riducendo il costo medio delle operazioni di accesso
- Due diversi gradi di replicazione:
  - In lettura (read replication)
  - Totale (full replication)

# Read replication

- Memorizzazione di copie multiple di blocchi in sola lettura
  - Protocollo “lettori multipli/scrittore singolo”
  - Schema conveniente se numero di read  $\gg$  numero di write
- In caso di accesso in scrittura a un blocco tutte le altre copie vengono invalidate (o aggiornate)

# Read replication

- Ogni blocco ha un “possessore” che lo memorizza in modo permanente
  - e questa è l'unica copia modificabile
- Necessaria individuazione delle copie di un blocco
  - Soluzioni
    - Il nodo possessore tiene una traccia dei nodi che ne hanno una copia
    - Uso di una lista distribuita dei nodi che ne hanno una copia

# Read replication

- Lettura

| Client   | Host remoto                                   |
|--|---|
| Se il blocco non è locale, determina la sua locazione e invia la richiesta     |   |
|  | Riceve la richiesta ed invia copia del blocco |
| Riceve il blocco; modifica i permessi della copia in scrittura come READ-ONLY; |   |
| Accesso ai dati  |   |

# Read replication

- Scrittura

| Client   | Host remoto  |
|--|--|
| Se il blocco non è locale, determina la sua locazione e invia la richiesta |  |
|  | Il possessore del blocco riceve la richiesta ed invia copia del blocco   |
| Riceve il blocco; invalida tutte le altre copie                            |  |
|  | Chi possiede una copia del blocco riceve invalidazione e invalida blocco |
| Accesso ai dati  |  |

# Read replication

- Vantaggi
  - Lettura tipicamente ad elevata località
  - Sincronizzazione relativamente facile
  - Consistenza per costruzione
- Svantaggi
  - Richiede invalidazione di copie
  - Possibile carico elevato per i possessori di pagine “popolari”
  - Scritture locali solo per il possessore della pagina

# Full replication

- Estensione del read replication
- Più nodi possono accedere in lettura e scrittura a blocchi condivisi
  - Protocollo “lettori multipli/scrittori multipli”
- Richiede meccanismi di consistenza!
  - Simile al caso di file replicati

## Full replication – Sequencer

- Un modo per garantire consistenza è quello di *sequenzializzare* le scritture
  - Utilizzo di un sequencer (processo in esecuzione su uno degli host)
  - Quando un processo cerca di scrivere nella memoria condivisa, la modifica viene inviata al sequencer
    - Assegna un numero progressivo alla modifica e lo comunica agli altri nodi via multicast
  - Ogni processo processa scritture in ordine progressivo (senza salti)

# Full replication – Sequencer

| Client                                    | Sequencer  | Host remoto                                 |
|---|--|---|
| In caso di scrittura, invia il dato       |  |   |
|   | Riceve il dato, incrementa il contatore e lo invia agli altri processi |   |
| Riceve l'ACK e aggiorna la memoria locale |  | Riceve il dato e aggiorna la memoria locale |

# Full replication

- **Vantaggi**
  - Accessi tipicamente locali
  - Colli di bottiglia meno probabili (broadcast scritte)
- **Svantaggi**
  - Sincronizzazione complessa
  - Sequencer può essere collo di bottiglia

# Consistenza della memoria

- Se più host possono scrivere sorgono problemi di consistenza della memoria
  - Esistono copie multiple di un valore
  - Problema in caso di scrittura e lettura successiva
- Una memoria è consistente (*coherent*) se il valore ritornato da una lettura è sempre il valore che il programmatore si aspetta

# Meccanismi di consistenza

- Necessari meccanismi di consistenza
- Modelli di consistenza
  - Consistenza stretta (*strict consistency*)
  - Consistenza sequenziale (*sequential consistency*)
  - Consistenza causale (*causal consistency*)
  - Consistenza debole (*weak consistency*)
  - ...

## Consistenza stretta

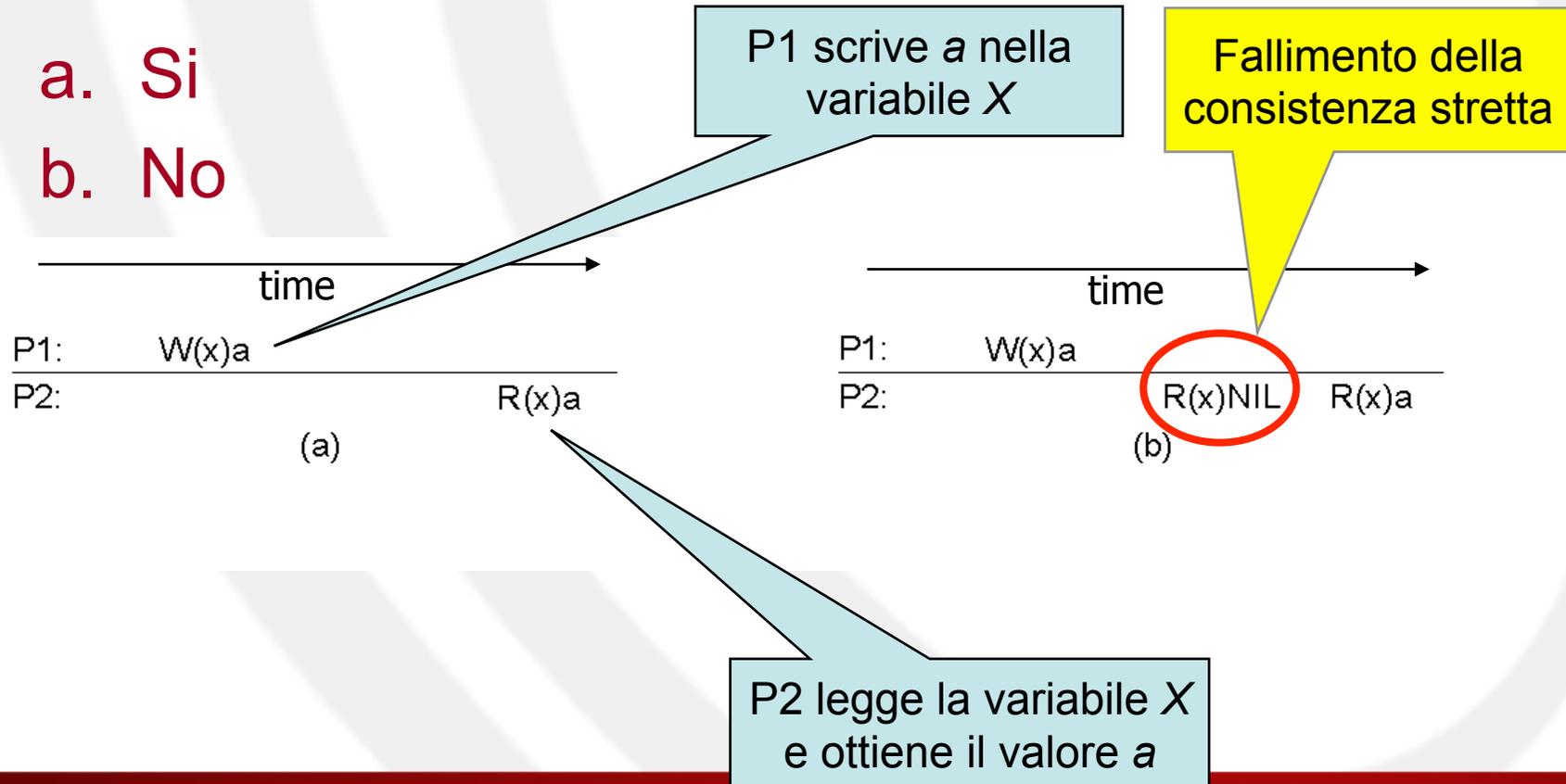
- Come se ci fosse una copia unica
- Ogni lettura di X ritorna il valore memorizzato dall'ultima scrittura di X
- Modello ideale impossibile da implementare
  - Esperienza mostra che i programmatori possono fare a meno della consistenza stretta
    - Sezioni critiche, mutua esclusione, semafori, ...
    - Programmi paralleli scritti bene devono essere indipendenti dalla velocità relativa dei processori e dall'interleaving delle istruzioni

# Consistenza stretta: esempio

- Consistenza stretta

a. Si

b. No



# Consistenza sequenziale

- Il risultato di ogni esecuzione è identico al caso in cui le operazioni siano eseguite dai processi in “qualche” ordine sequenziale [Lamport 79]
  - Riferimenti a memoria viste nello stesso ordine da tutti i processi
- Le operazioni di un singolo processo devono apparire nell'ordine specificato dal programma

# Consistenza sequenziale

## Esempio

- a. Consistenza sequenziale
- b. Consistenza non-sequenziale

|     |       |       |       |
|-----|-------|-------|-------|
| P1: | W(x)a |       |       |
| P2: | W(x)b |       |       |
| P3: |       | R(x)b | R(x)a |
| P4: |       | R(x)b | R(x)a |

(a)

|     |       |       |       |
|-----|-------|-------|-------|
| P1: | W(x)a |       |       |
| P2: | W(x)b |       |       |
| P3: |       | R(x)b | R(x)a |
| P4: |       | R(x)a | R(x)b |

P3 e P4 leggono e vedono un ordine diverso

(b)

# Consistenza causale

## [Hutto&Ahamad 90]

- Scritture correlate causalmente devono essere viste nello stesso ordine da tutti i processi
- Scritture concorrenti possono essere viste in ordini diversi da diversi processori
- Bisogna tenere traccia della dipendenza delle operazioni
  - Richiede overhead

# Consistenza causale: esempio

|     |       |       |             |
|-----|-------|-------|-------------|
| P1: | W(x)a |       | W(x)c       |
| P2: | R(x)a | W(x)b |             |
| P3: | R(x)a |       | R(x)c R(x)b |
| P4: | R(x)a |       | R(x)b R(x)c |

Concorrenti.  
Processi possono vederle in ordini differenti

Consistente causalmente, ma non sequenzialmente

# Consistenza causale: esempio

- a. Consistenza non causale
- b. Consistenza causale

|     |       |       |             |
|-----|-------|-------|-------------|
| P1: | W(x)a |       |             |
| P2: |       | R(x)a | W(x)b       |
| P3: |       |       | R(x)b R(x)a |
| P4: |       | R(x)a | R(x)b       |

(a)

Causalmente correlate

|     |       |       |             |
|-----|-------|-------|-------------|
| P1: | W(x)a |       |             |
| P2: |       |       | W(x)b       |
| P3: |       |       | R(x)b R(x)a |
| P4: |       | R(x)a | R(x)b       |

(b)

Non causalmente correlate

# Consistenza debole [Dubois 88]

- Utilizza variabili di sincronizzazione
  - Quando una sincronizzazione viene effettuata, tutte le scritture precedenti alla sincronizzazione vengono esportate verso le altre macchine e tutte le scritture di queste ultime vengono importate
- Accesso alle variabili di sincronizzazione consistente in modo sequenziale (comunicato in broadcast, accesso atomico)
- Nessun accesso a una variabile di sincronizzazione è permesso fino a che tutte le scritture precedenti non sono state completate da altre parti
- Nessun accesso ai dati (R/W) è permesso fino a che tutti gli accessi precedenti a variabili di sincronizzazione sono stati effettuati
  - Sincronizzazione permette di avere ultimi dati aggiornati

# Consistenza debole: esempio

- Sequenza valida di eventi per la consistenza debole

|           |       |       |   |
|-----------|-------|-------|---|
| P1: W(x)a | W(x)b | S     |   |
| P2:       | R(x)a | R(x)b | S |
| P3:       | R(x)b | R(x)a | S |

(a)

P2 e P3 leggono prima della sincronizzazione. L'ordine può essere diverso



# Riassunto Modelli Consistenza

| <i>Consistenza</i> | <i>Descrizione</i>   |
|--------------------|--|
| Stretta            | Importa l'ordine assoluto di tempo degli accessi condivisi   |
| Sequenziale        | Tutti i processi vedono gli accessi condivisi nello stesso ordine. Gli accessi non sono ordinati nel tempo |
| Causale            | Tutti i processi vedono accessi condivisi correlati causalmente nello stesso ordine                        |

(a) Modelli di consistenza senza operazioni di sincronizzazione

| <i>Consistenza</i> | <i>Descrizione</i>  |
|--------------------|---|
| Weak               | Dati condivisi consistenti solo dopo una sincronizzazione   |
| Release            | Dati condivisi consistenti solo in uscita da una sezione critica  |
| Entry              | Dati condivisi relativi ad una regione critica sono resi consistenti all'ingresso della sezione critica |

(b) Modelli di consistenza con operazioni di sincronizzazione