UNIVERSITÀ di VERONA

Dipartimento di INFORMATICA

PARCO Parallel Computing Lab

Modelli di programmazione parallela

Panoramica

- Esistono diversi modelli di programmazione parallela di uso comune:
  - Memoria condivisa
  - Discussioni
  - Passaggio del messaggio
  - Dati paralleli
  - Ibrido
- I modelli di programmazione parallela esistono come astrazione al di sopra delle architetture hardware e di memoria.

Panoramica

- Anche se potrebbe non sembrare evidente, questi modelli NON sono specifici per un particolare tipo di macchina o architettura di memoria. In effetti, ognuno di questi modelli può (teoricamente) essere implementato su qualsiasi hardware sottostante. Due esempi:
  - Memoria condivisamodello su a*memoria distribuita*macchina:
    - Approccio ALLCACHE di Kendall Square Research (KSR). La memoria della macchina era distribuita fisicamente, ma appariva all'utente come un'unica memoria condivisa (spazio di indirizzi globale). Genericamente questo approccio viene definito "memoria virtuale condivisa".
  - Passaggio del messaggiomodello su a*memoria condivisa*macchina:
    - MPI su SGI Origin. SGI Origin utilizzava il tipo CC-NUMA di architettura di memoria condivisa, in cui ogni attività ha accesso diretto alla memoria globale. Tuttavia, la capacità di inviare e ricevere messaggi con MPI, come avviene comunemente su una rete di macchine a memoria distribuita, non solo è implementata ma è utilizzata molto comunemente.

Panoramica

- Quale modello utilizzare è spesso *una combinazione* di ciò che è disponibile e scelta personale. *Non esiste un modello "migliore".*, anche se esistono certamente implementazioni migliori di alcuni modelli rispetto ad altri.
- Le sezioni seguenti descrivono ciascuno dei modelli sopra menzionati e discutere anche alcune delle loro effettive implementazioni

Modello di memoria condivisa

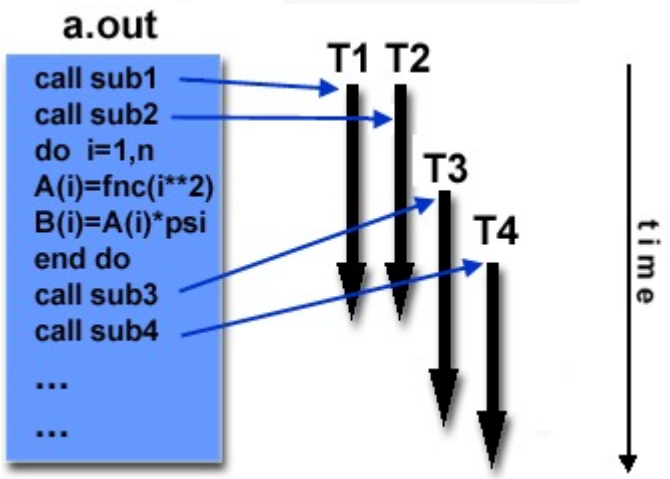
- Nel modello di programmazione a memoria condivisa, le attività condividono uno spazio di indirizzi comune, che leggono e scrivono in modo asincrono.
- Vari *meccanismi come serrature/semafori* può essere utilizzato per controllare l'accesso alla memoria condivisa.
- Un vantaggio di questo modello dal punto di vista del programmatore è che manca la nozione di "proprietà" dei dati
  - Non è necessario specificare esplicitamente la comunicazione dei dati tra le attività.
  - Lo sviluppo del programma può spesso essere semplificato.
- Uno svantaggio importante in termini di prestazioni è che diventa di più *difficile comprendere e gestire la località dei dati.*
  - Mantenere i dati locali rispetto al processore che li utilizza preserva gli accessi alla memoria, gli aggiornamenti della cache e il traffico del bus che si verifica quando più processori utilizzano gli stessi dati.
  - Sfortunatamente, il controllo della località dei dati è difficile da comprendere e va oltre il controllo dell'utente medio.

Modello di memoria condivisa: implementazione

- Sulle piattaforme di memoria condivisa, i compilatori nativi traducono *variabili del programma utente* in *indirizzi di memoria effettivi*, che sono globali.
- Attualmente non esistono implementazioni comuni della piattaforma di memoria distribuita.
  - Tuttavia, come accennato in precedenza nella sezione Panoramica, l'approccio KSR ALLCACHE forniva una visualizzazione della memoria condivisa dei dati anche se la memoria fisica della macchina era distribuita.

Modello con fili

- Nel modello thread della programmazione parallela, un singolo processo può avere più percorsi di esecuzione simultanei.
- Forse l'analogia più semplice che può essere utilizzata per descrivere i thread è il concetto di un singolo programma che include un numero di subroutine:



- Il programma principale**a.out**è pianificato per l'esecuzione dal sistema operativo nativo. a.out carica e acquisisce tutte le risorse utente e di sistema necessarie per l'esecuzione.
- a.out esegue del lavoro seriale e quindi crea una serie di attività (thread) che possono essere pianificate ed eseguite contemporaneamente dal sistema operativo.
- Ogni thread ha dati locali, ma condivide anche tutte le risorse di a.out. Ciò consente di risparmiare il sovraccarico associato alla replica delle risorse di un programma per ciascun thread. Ogni thread beneficia anche di una vista globale della memoria perché condivide lo spazio di memoria di a.out.
- Il lavoro di un thread può essere meglio descritto come una subroutine all'interno del main programma. Qualsiasi thread può eseguire qualsiasi subroutine contemporaneamente agli altri thread.
- I thread comunicano tra loroattraverso la memoria globale(in aggiornamento indirizzi).Ciò richiede la sincronizzazionecostrutti per garantire che più di un thread non aggiorni lo stesso indirizzo globale in qualsiasi momento.
- I thread possono andare e venire, ma a.out rimane presente per fornire le risorse condivise necessarie fino al completamento dell'applicazione.

Modello dei thread: implementazione

- I thread sono comunemente associati ad architetture di memoria condivisa e sistemi operativi.
- Dal punto di vista della programmazione, le implementazioni dei thread comprendono comunemente:
  - Una libreria di subroutine chiamate dal codice sorgente parallelo
  - Un insieme di direttive del compilatore incorporate nel codice sorgente seriale o parallelo
- In entrambi i casi, il programmatore è responsabile della determinazione di tutto il parallelismo.
- Le implementazioni threaded non sono nuove nell'informatica.
  - Storicamente, i fornitori di hardware hanno implementato le proprie versioni proprietarie dei thread.
  - Queste implementazioni differivano sostanzialmente l'una dall'altra rendendo difficile per i programmatori sviluppare applicazioni threaded portatili.
- Gli sforzi di standardizzazione non correlati hanno portato a due implementazioni molto diverse dei thread:
  - ***Discussioni POSIX***E
  - ***OpenMP***.

Fili positivi

- Basato su libreria;
  - richiede una codifica parallela
- Specificato dallo standard IEEE POSIX 1003.1c (1995).
- Solo linguaggio C
- Comunemente indicato come ***Pthread***.
- La maggior parte dei fornitori di hardware ora offre Pthread in aggiunta alle implementazioni dei thread proprietari.
- Parallelismo molto esplicito;
  - richiede una significativa attenzione da parte del programmatore ai dettagli.

OpenMP

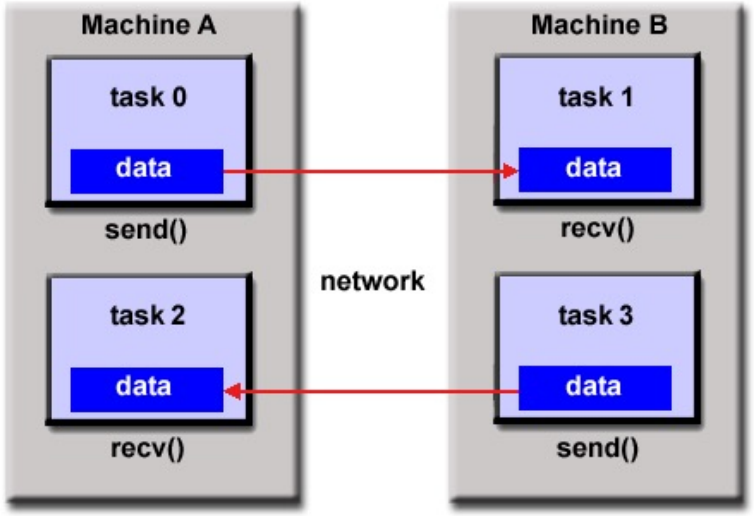
- Basato sulla direttiva del compilatore;
  - può utilizzare il codice seriale
- Definito e approvato congiuntamente da un gruppo di importanti fornitori di hardware e software per computer.
  - L'API Fortran OpenMP è stata rilasciata il 28 ottobre 1997.
  - L'API C/C++ è stata rilasciata alla fine del 1998.
- Portabile/multipiattaforma, comprese le piattaforme Unix e Windows NT
- Disponibile nelle implementazioni C/C++ e Fortran
- Può essere molto facile e semplice da usare: prevede il "parallelismo incrementale"

OpenMP

- Microsoft dispone di una propria implementazione per i thread, che non è correlata allo standard UNIX POSIX o OpenMP.
- Maggiori informazioni:
  - Tutorial sui thread POSIX: computing.llnl.gov/tutorials/pthreads
  - Esercitazione su OpenMP: computing.llnl.gov/tutorials/openMP

Modello di passaggio dei messaggi

- Il modello di trasmissione dei messaggi dimostra le seguenti caratteristiche:



- Una serie di compiti che *utilizzare la propria memoria locale* durante il calcolo. Più attività possono risiedere sulla stessa macchina fisica e su un numero arbitrario di macchine.
- Le attività scambiano dati attraverso le comunicazioni *inviando e ricevendo messaggi*.
- Il trasferimento dei dati di solito richiede l'esecuzione di operazioni cooperative da parte di ciascun processo. Ad esempio, un'operazione di invio deve avere un'operazione di ricezione corrispondente.

Modello di passaggio dei messaggi: implementazione

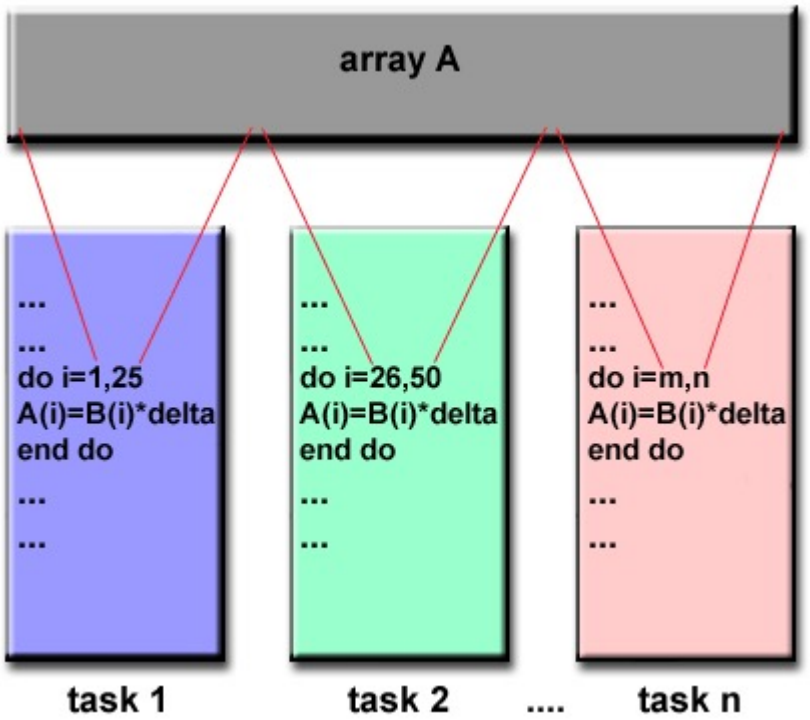
- Dal punto di vista della programmazione, le implementazioni dello scambio di messaggi comprendono comunemente una libreria di subroutine incorporate nel codice sorgente.
  - Il programmatore è responsabile della determinazione di tutto il parallelismo.
- Storicamente, a partire dagli anni '80 sono disponibili numerose librerie per lo scambio di messaggi.
  - Queste implementazioni differivano sostanzialmente l'una dall'altra rendendo difficile per i programmatori sviluppare applicazioni portatili.
- Nel 1992 è stato formato il Forum MPI con l'obiettivo primario di stabilire un'interfaccia standard per le implementazioni dello scambio di messaggi.

Modello di passaggio dei messaggi: implementazione

- Parte 1 del**Interfaccia per il passaggio dei messaggi (MPI)**è stata rilasciata nel 1994. La Parte 2 (MPI-2) è stata rilasciata nel 1996. Entrambe le specifiche MPI sono disponibili sul Web all'indirizzohttp://www-unix.mcs.anl.gov/mpi/ .
- MPI è ora lo standard industriale "de facto" per lo scambio di messaggi, sostituendo praticamente tutte le altre implementazioni di scambio di messaggi utilizzate per il lavoro di produzione.
  - La maggior parte, se non tutte, le piattaforme di calcolo parallelo più diffuse offrono almeno un'implementazione di MPI. Alcuni offrono un'implementazione completa di MPI-2.
- Per le architetture a memoria condivisa, le implementazioni MPI solitamente non utilizzano una rete per le comunicazioni delle attività. Usano la memoria condivisa (copie della memoria) per motivi di prestazioni.
- Maggiori informazioni:
  - Esercitazione sull'MPI:computing.llnl.gov/tutorials/mpi

Modello parallelo dei dati

- Il modello parallelo dei dati dimostra le seguenti caratteristiche:



- La maggior parte del lavoro parallelo si concentra sull'esecuzione di operazioni su un set di dati. Il set di dati è generalmente organizzato in a*struttura comune*, ad esempio un array o un cubo.
- Una serie di attività lavora collettivamente sulla stessa struttura dati, tuttavia,*ogni attività funziona su una partizione diversa della stessa struttura dati*.
- Le attività eseguono la stessa operazione sulla loro partizione di lavoro, ad esempio "aggiungi 4 a ogni elemento dell'array".

Modello parallelo dei dati

- Nelle architetture a memoria condivisa, tutte le attività possono avere accesso alla struttura dei dati attraverso la memoria globale. Nelle architetture di memoria distribuita la struttura dei dati è suddivisa e risiede come "pezzi" nella memoria locale di ciascun task.

Modello parallelo dei dati: implementazione

- La programmazione con il modello parallelo dei dati viene solitamente eseguita da *scrivere un programma con costrutti paralleli di dati*. I costrutti possono essere chiamate a una libreria di subroutine parallele di dati o direttive del compilatore*riconosciuto da un compilatore parallelo di dati*.
- **Fortran 90 e 95 (F90, F95):**Estensioni standard ISO/ANSI a Fortran 77.
  - Contiene tutto ciò che è in Fortran 77
  - Nuovo formato del codice sorgente; aggiunte al set di caratteri
  - Aggiunte alla struttura e ai comandi del programma
  - Aggiunte di variabili: metodi e argomenti
  - Aggiunti puntatori e allocazione dinamica della memoria
  - Aggiunta l'elaborazione degli array (array trattati come oggetti).
  - Aggiunte funzioni ricorsive e nuove intrinseche
  - Molte altre novità
  - Sono disponibili implementazioni per le piattaforme parallele più comuni

Modello parallelo dei dati: implementazione

- **Fortran ad alte prestazioni (HPF):**Estensioni a Fortran 90 per supportare la programmazione parallela dei dati.
  - Contiene tutto in Fortran 90
  - Aggiunte direttive per indicare al compilatore come distribuire i dati
  - Aggiunte asserzioni che possono migliorare l'ottimizzazione del codice generato
  - Aggiunti costrutti paralleli di dati (ora parte di Fortran 95)
  - Sono disponibili implementazioni per le piattaforme parallele più comuni
- **Direttive del compilatore:**Consentire al programmatore di specificare la distribuzione e l'allineamento dei dati. Le implementazioni Fortran sono disponibili per le piattaforme parallele più comuni
- Le implementazioni di memoria distribuita di questo modello di solito prevedono che il compilatore converta il programma in codice standard con chiamate a una libreria di passaggio di messaggi (solitamente MPI) per distribuire i dati a tutti i processi. Tutto lo scambio di messaggi avviene in modo invisibile al programmatore

Altri modelli

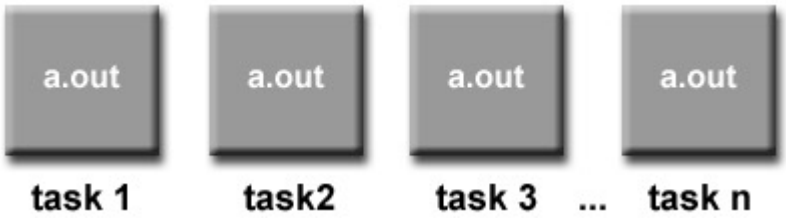
- Esistono certamente altri modelli di programmazione parallela oltre a quelli menzionati in precedenza, e continueranno ad evolversi insieme al mondo in continua evoluzione dell'hardware e del software dei computer. Qui vengono menzionati solo tre dei più comuni:
  - Ibrido
  - SPMD
  - MPMD

Altri modelli: Ibrido

- In questo modello vengono combinati due o più modelli di programmazione parallela.
- Un esempio comune di modello ibrido:
  - *combinazione del modello di passaggio dei messaggi (MPI) con il modello di thread (thread POSIX) o il modello di memoria condivisa (OpenMP)*. Questo modello ibrido si presta bene all'ambiente hardware sempre più comune delle macchine SMP collegate in rete.
- Un altro esempio comune di modello ibrido:
  - *combinando i dati in parallelo con lo scambio di messaggi*. Come menzionato in precedenza nella sezione del modello parallelo dei dati, implementazioni parallele dei dati (F90, HPF) sulla memoria distribuita
    le architetture utilizzano effettivamente il passaggio di messaggi per trasmettere dati tra attività, in modo trasparente per il programmatore.

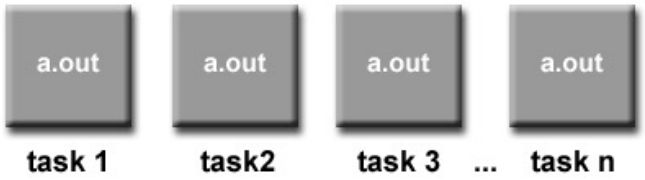
Altri modelli: Programma Singolo Dati multipli (SPMD)

- SPMD è in realtà un modello di programmazione di "alto livello" che può essere costruito su qualsiasi combinazione dei modelli di programmazione parallela precedentemente menzionati.
- *Un singolo programma viene eseguito da tutti i task contemporaneamente*.
- In qualsiasi momento, le attività possono essere in esecuzione *le stesse o diverse istruzioni all'interno dello stesso programma*



Altri modelli: Programma Singolo Dati multipli (SPMD)

- *SPMD rispetto a SIMD:*
  - Nell'SPMD, più processori autonomi eseguono simultaneamente lo stesso programma in punti indipendenti, anziché in un unico puntopasso di marcia QuelloSIMD impone su dati diversi
  - Con SPMD, le attività possono essere eseguite per scopi generaliCPU ; SIMD richiede processori vettoriali per manipolare i flussi di dati
  - SPMD è una sottocategoria di MIMD
- I programmi SPMD di solito hanno la logica necessaria programmata al loro interno per consentire a diversi compiti di ramificarsi o eseguire in modo condizionato solo quelle parti del programma per cui sono progettati. Cioè, le attività non devono necessariamente eseguire l'intero programma, forse solo una parte di esso.
- Tutte le attività possono utilizzare dati diversi



Altri modelli: Programma multiplo Dati multipli (MPMD)

- Come SPMD, MPMD è in realtà un modello di programmazione di "alto livello" che può essere costruito su qualsiasi combinazione dei modelli di programmazione parallela precedentemente menzionati.
- Applicazioni MPMD in genere *avere più file oggetto eseguibili* (programmi). Mentre l'applicazione viene eseguita in parallelo, ogni attività può essere eseguita *lo stesso programma o uno diverso* come altri compiti.
- Tutte le attività possono utilizzare dati diversi