Algoritmo genetico parallelo sul Single-Chip Cloud Computer

Tesi di Licenza

Scuola Superiore Sant'Anna

Relatore

Prof. *Giuseppe Lipari*

Scuola Superiore Sant'Anna

Candidato

Vincenzo Maffione

Tutor

Prof. *Paolo Ancilotti*

Scuola Superiore Sant'Anna

8 Luglio 2011

Obiettivi

1. Studio dell'architettura hardware del Single-Chip Cloud computer (SCC)
2. Studio di alcuni strumenti software predisposti alla programmazione sull'SCC
3. Sviluppo sull'SCC di un framework di ottimizzazione mono-obiettivo non vincolata basata su un algoritmo genetico parallelo

Tendenze architetturali

Architettura shared memory

La stragrande maggioranza dei sistemi multicore o multiprocessore disponibili oggi sul mercato si basa sull'architettura a memoria comune

Tendenze architetturali

Problemi

Inconvenienti dell'architettura a memoria comune:

- l'accesso alla memoria centrale è il collo di bottiglia del sistema
- a causa del caching, è necessario gestire la coerenza tramite opportuni protocolli

L'architettura a memoria comune è dunque poco scalabile.

Tendenze architetturali

Architettura a memoria distribuita

I problemi precedenti spingono la ricerca a dirigersi verso architetture a memoria distribuita:

- una memoria privata per ciascuna unità di calcolo
- comunicazione per scambio di messaggi tramite una rete di interconnessione

Intel SCC

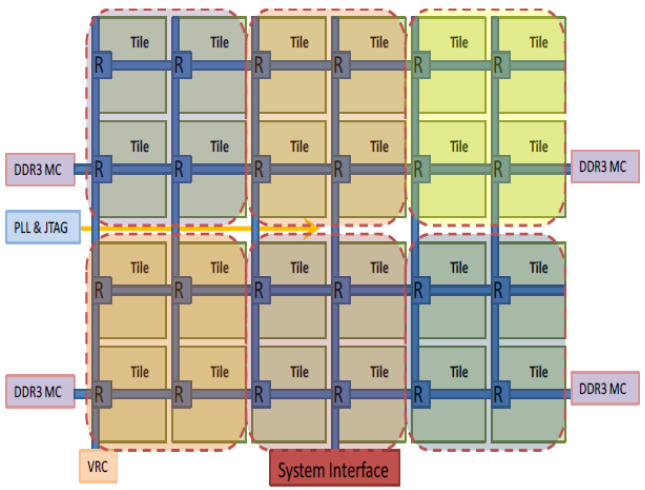
Nel 2010 Intel ha fatto partire un progetto di ricerca sulle architetture a scambio di messaggi, sviluppando il processore sperimentale *Single-Chip Cloud Computer*.

Dati salienti

- 48 core Pentium
- instruction set complesso (Intel Architecture)
- possibilità di caricare Linux su ciascun core

Architettura dell'SCC

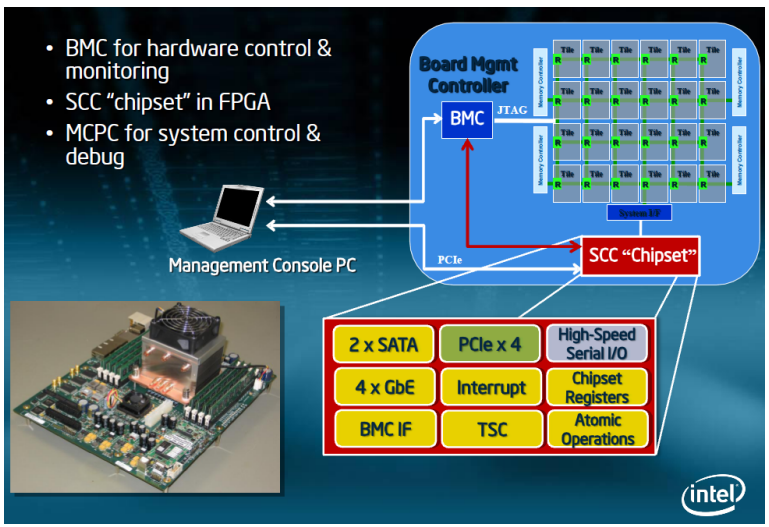
1. 24 *tiles* (mattonelle) che compongono la griglia
2. una rete mesh composta da 24 router con picco di banda sul taglio pari a 256 GB/s
3. 4 DDR3 memory controller integrati
4. supporto hardware per lo scambio dei messaggi



Piattaforma di sviluppo

Oltre all'SCC, comprende

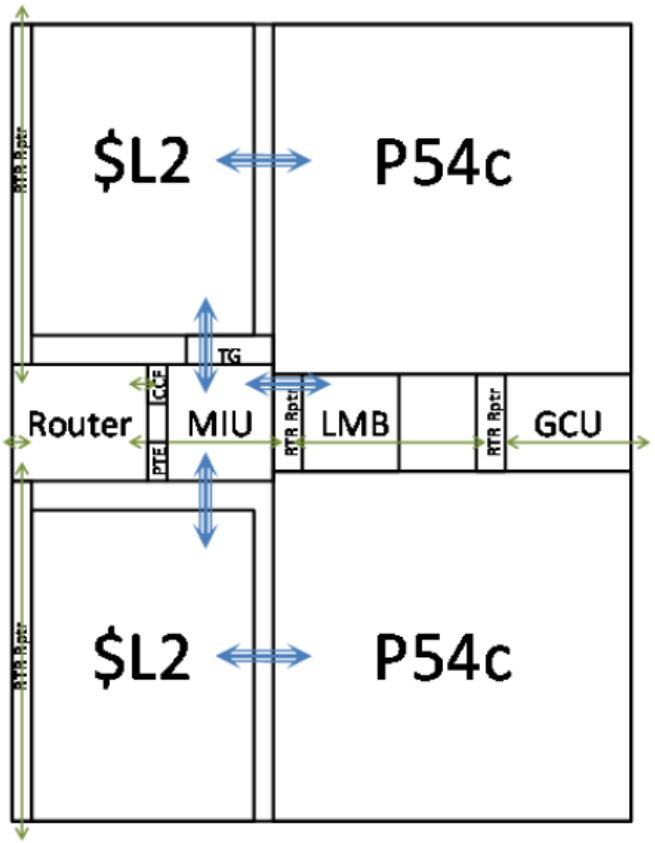
- un FPGA che fa da chipset per l'SCC
- un *Board Management Controller* per il controllo delle funzionalità critiche della piattaforma
- una workstation operativa (MCPC)



Architettura del tile

1. due IA core PC54C, con cache L1 interna e cache L2 esterna
2. un crossbar router a 5 porte, che interfaccia il tile con la mesh
3. una *mesh interface unit* (MIU) che gestisce tutti gli accessi alla memoria e le operazioni di scambio dei messaggi
4. una *memory lookup table* (LUT) che permette la traduzione degli indirizzi fisici del core in indirizzi di sistema (globali)
5. un message-passing buffer, che supporta lo scambio di messaggi.
6. ciruciterie di generazione e raccordo del clock (GCU e CCF)

Architettura del tile



Supporto allo scambio di messaggi

Message passing buffer

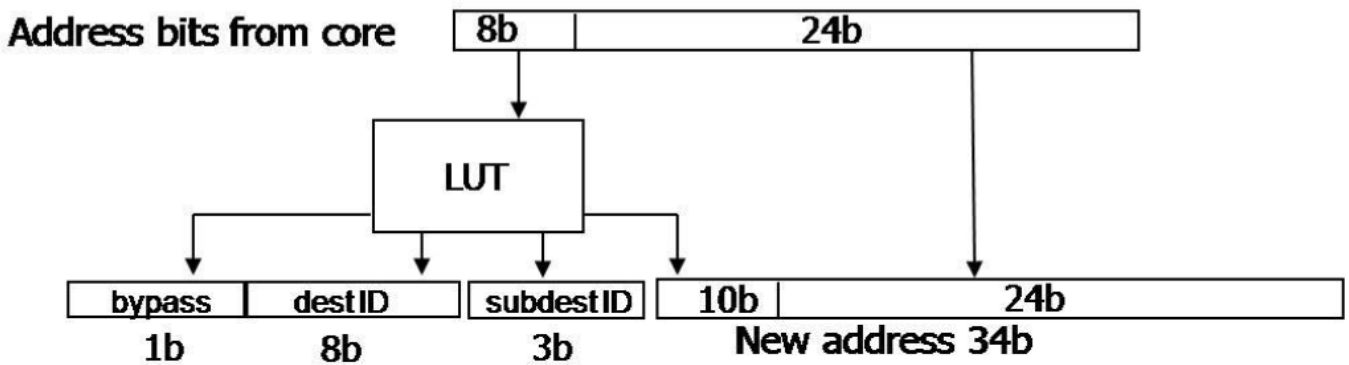
16 KB di SRAM accedibile da qualsiasi core, da utilizzarsi preferibilmente come memoria tampone durante lo scambio di messaggi

Modifiche principali al core P54C

- aggiunta di un nuovo tipo di memoria, MPBT (message passing buffer type)
- aggiunta dell'istruzione CL1INVMB per invalidare le linee di L1 cache di tipo MPBT
- aggiunta di un write combine buffer verso il bus di memoria, che agisce sui dati di tipo MPBT

Tabelle di lookup

Permettono di tradurre gli indirizzi fisici su 32 bit di un core in indirizzi di sistema a 46 bit.



Libreria RCCE

Caratteristiche

- libreria minimale e di basso livello che permette ai core dell'SCC di comunicare scambiandosi messaggi
- concepita inizialmente per lavorare sull'SCC in mancanza di sistema operativo
- primitive di invio e ricezione bloccanti
- non utilizza il meccanismo delle interruzioni
- molto efficiente (niente servizi di sistema, primitive bloccanti)
- due interfacce: una di più alto livello (*nongory*) e l'altra di più basso livello (*gory*)

Libreria RCCE

Modello offerto al programmatore (*nongory*)

- ad ogni core tra gli N coinvolti nell'elaborazione è assegnato un *rank* diverso (intero tra 0 ed N-1)
- le primitive `RCCE_send()` e `RCCE_receive()` permettono di inviare/ricevere un numero di byte arbitrario verso/da un altro core
- il programmatore deve attenersi al modello SPMD (*Single Program Multiple Data*)
- su ciascun core non possono essere lanciate contemporaneamente più applicazioni che utilizzano RCCE (o l'MPB)

Libreria RCCE

Implementazione (*nongory*)

- modello di allocazione simmetrica delle variabili nell'MPB
- due array di flag di sincronizzazione per ogni core, l'array `sent` e l'array `ready`, aventi ciascuno N elementi
- la sincronizzazione avviene in modo semplice con un protocollo basato sulle attese attive
- due implementazioni dei flag possibili, con diversi overhead spaziali e temporali

Algoritmo genetico

Definizione

É un algoritmo stocastico di ottimizzazione *population-based*.
L'evoluzione avviene per mezzo di

1. selezione
2. crossover (ricombinazione)
3. mutazione

Algoritmo genetico

Altre caratteristiche dell'algoritmo

- fitness scaling
- elite children
- inizializzazione della popolazione iniziale in modo casuale
- condizioni di terminazione (numero di iterazioni, varianza della popolazione)

Parallelizzazione dell'algoritmo genetico

Modelli di parallelizzazione

1. Master-slave (sincronizzato/asincrono)
2. Static subpopulation with migration
3. Overlapping subpopulation without migration
4. Massively parallel genetic algorithms

Parallelizzazione dell'algoritmo genetico

Il modello con sottopopolazioni e migrazioni è quello più adatto all'architettura ibrida dell'SCC.

Si alternano fasi di evoluzione isolata (algoritmo sequenziale) a fasi di migrazioni.

Parallelizzazione dell'algoritmo genetico

Estensioni necessarie per la parallelizzazione

- individuazione di uno schema di migrazione
- individuazione di un algoritmo di terminazione distribuita

Schema di migrazione

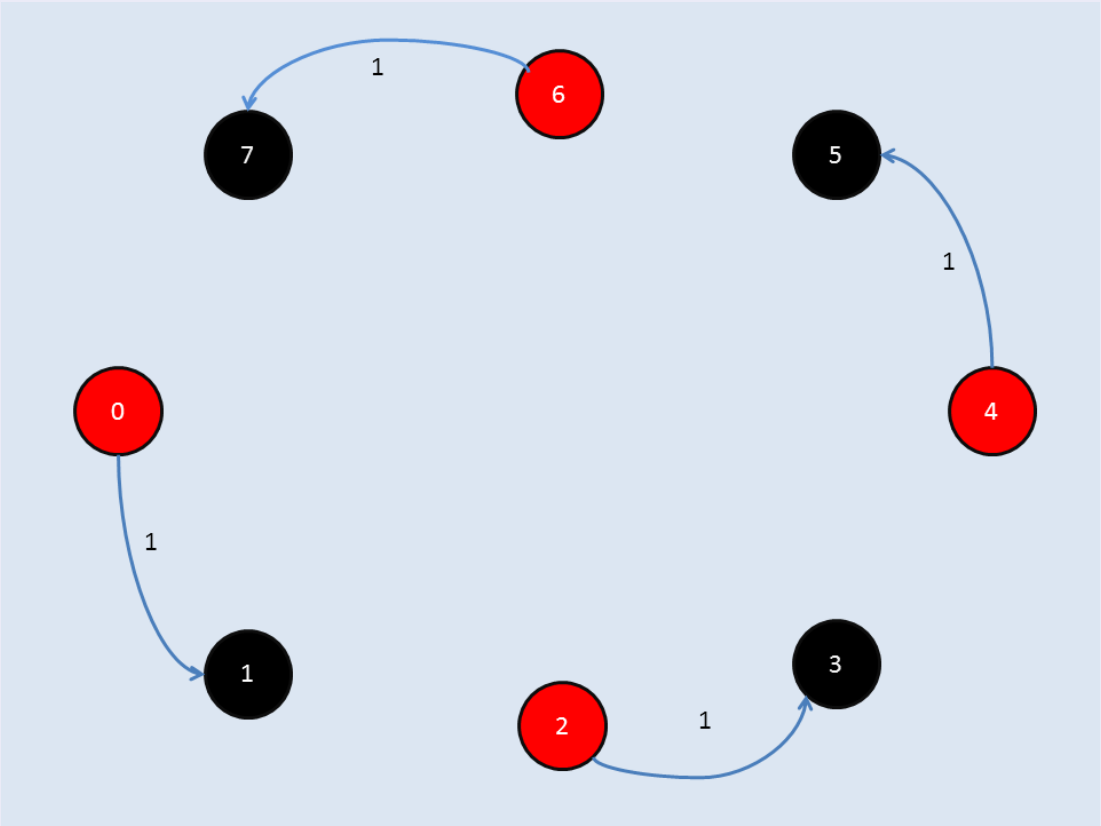
Specifica come il materiale genetico localmente dominante viene scambiato tra i core durante una fase di migrazione.

Schema a flusso circolare unidirezionale

- ciascun core ha un core precedente ed un successivo
- durante una fase di migrazione ciascun core riceve dal precedente e invia al successivo
- ciascun core ha un colore che determina l'ordine degli scambi
- due parametri da specificare: *migration* fraction e *migration period*
- migrazione in due o tre passi (primitive bloccanti)
- si evitano deadlock

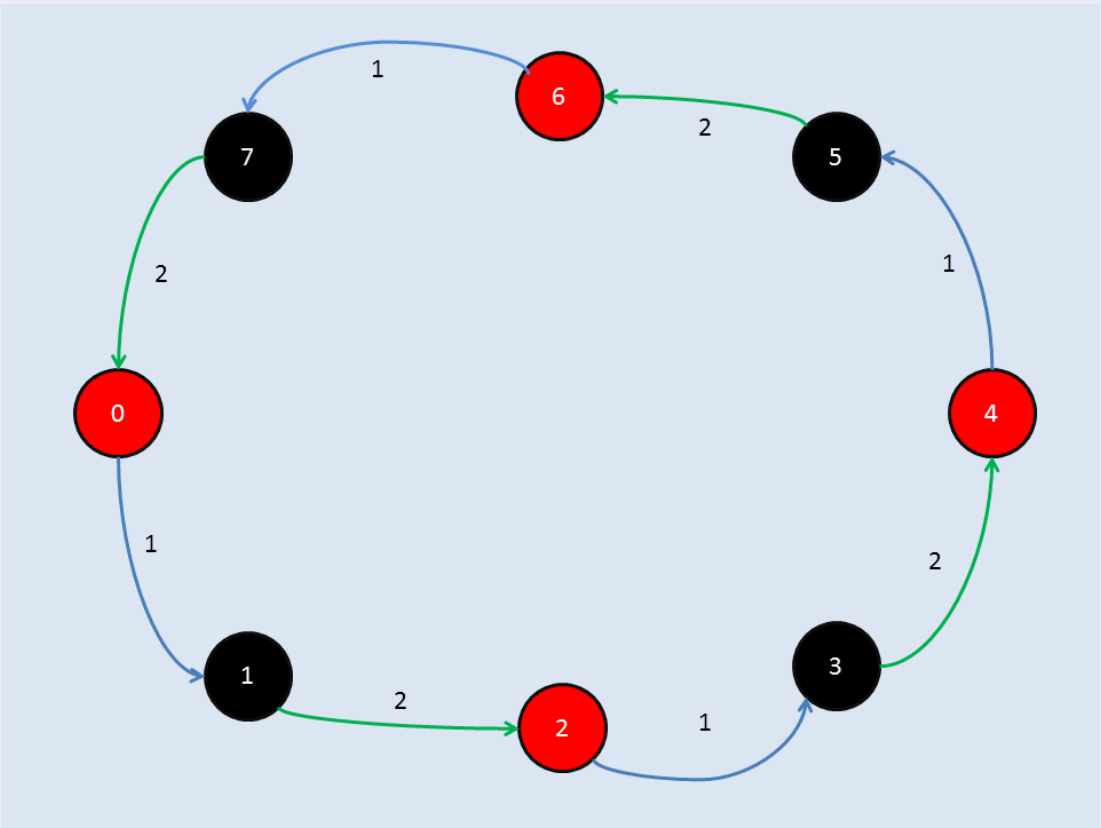
Schema di migrazione

Esempio



Schema di migrazione

Esempio (continua)



Schema di migrazione

Ricerca di cicli hamiltoniani bilanciati sull'SCC

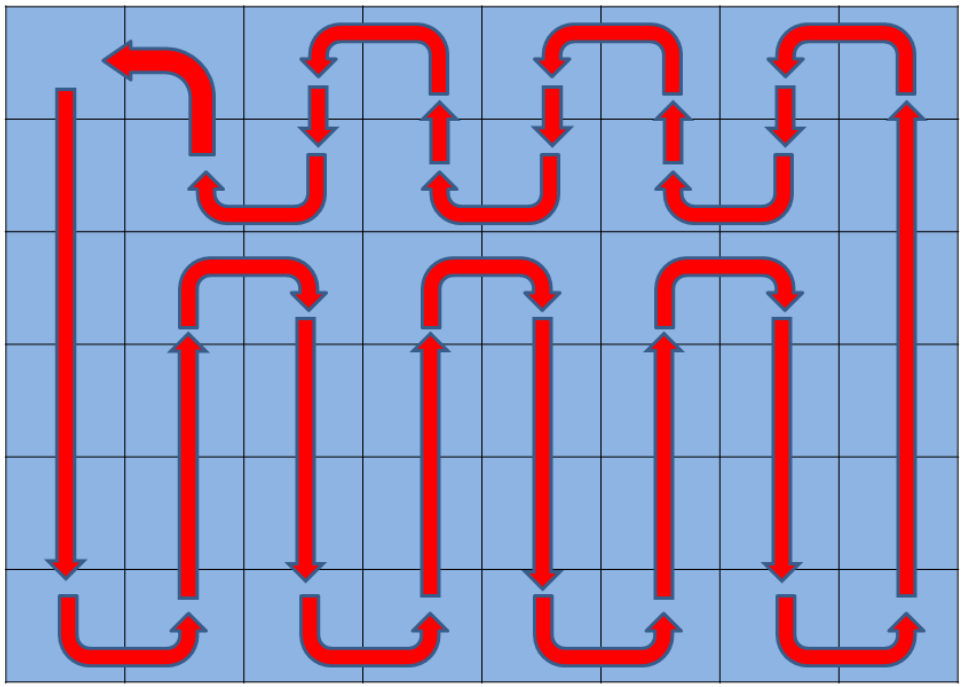
Problema semplice, in quanto il grafo è completamente connesso.

I core disponibili possono essere solamente un sottoinsieme di quelli totali. Si cercano cicli il più possibile *bilanciati*:

1. scelta di percorsi ottimi in casi particolarmente favorevoli
2. euristica da applicare nei restanti casi

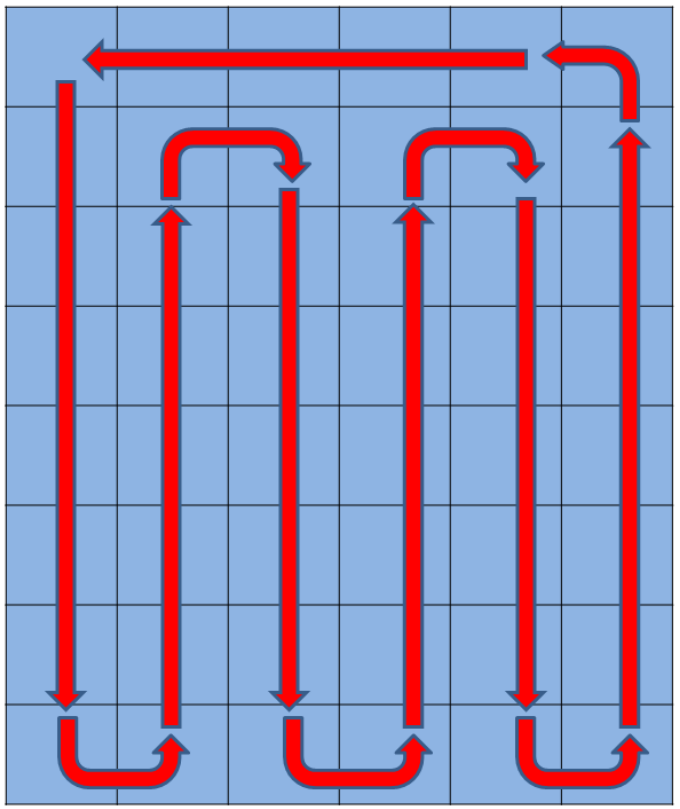
Schema di migrazione

Esempio ciclo ottimo



Schema di migrazione

Euristica da applicare nel caso generale



Algoritmo distribuito di terminazione

Problemi

- i core possono convergere in iterazioni diverse
- una fase di migrazione non può essere eseguita solo da un sottoinsieme di core
- si può terminare solo quando tutti i core sono arrivati a convergenza
- si vuole evitare di appesantire le comunicazioni

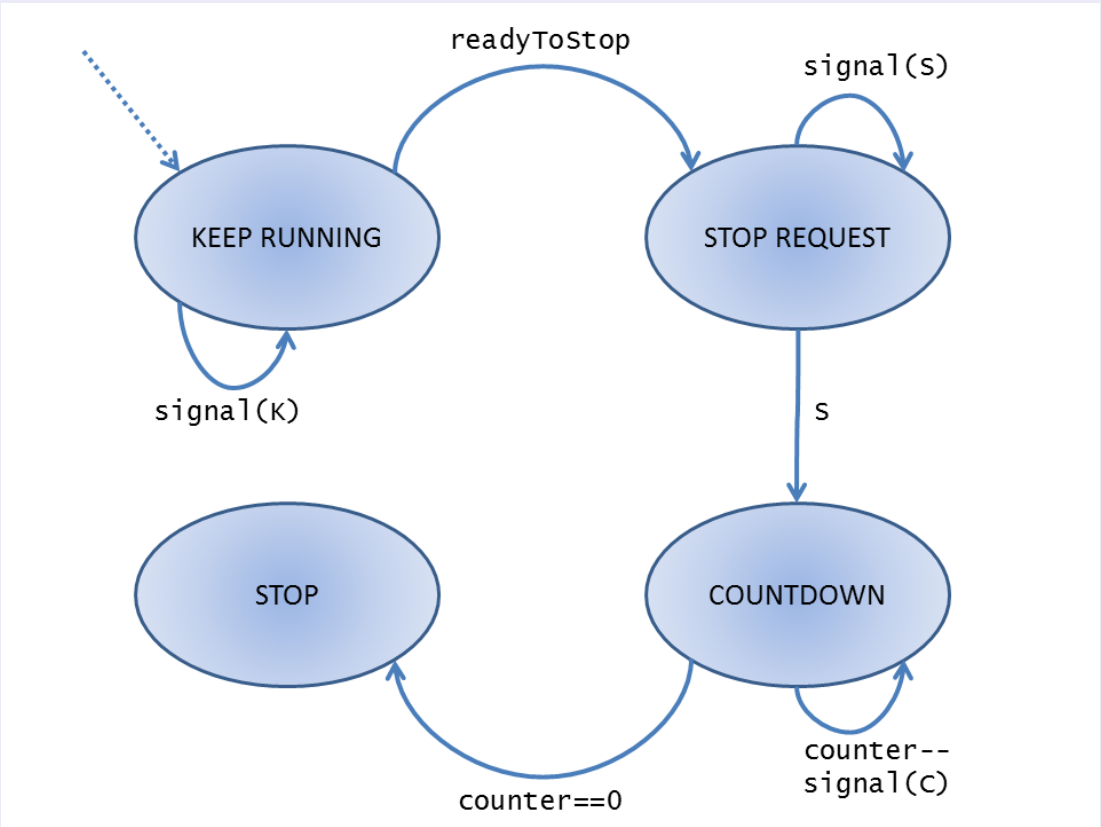
Algoritmo distribuito di terminazione

Soluzione

- protocollo di segnalazione che agisce solo durante le fasi di migrazione
- ogni nodo segnala la propria decisione di terminare al nodo successivo
- tutti i core terminano esattamente durante la prima fase di migrazione in cui la terminazione è possibile
- necessità di un nodo coordinatore
- implementazione tramite macchina a stati finiti

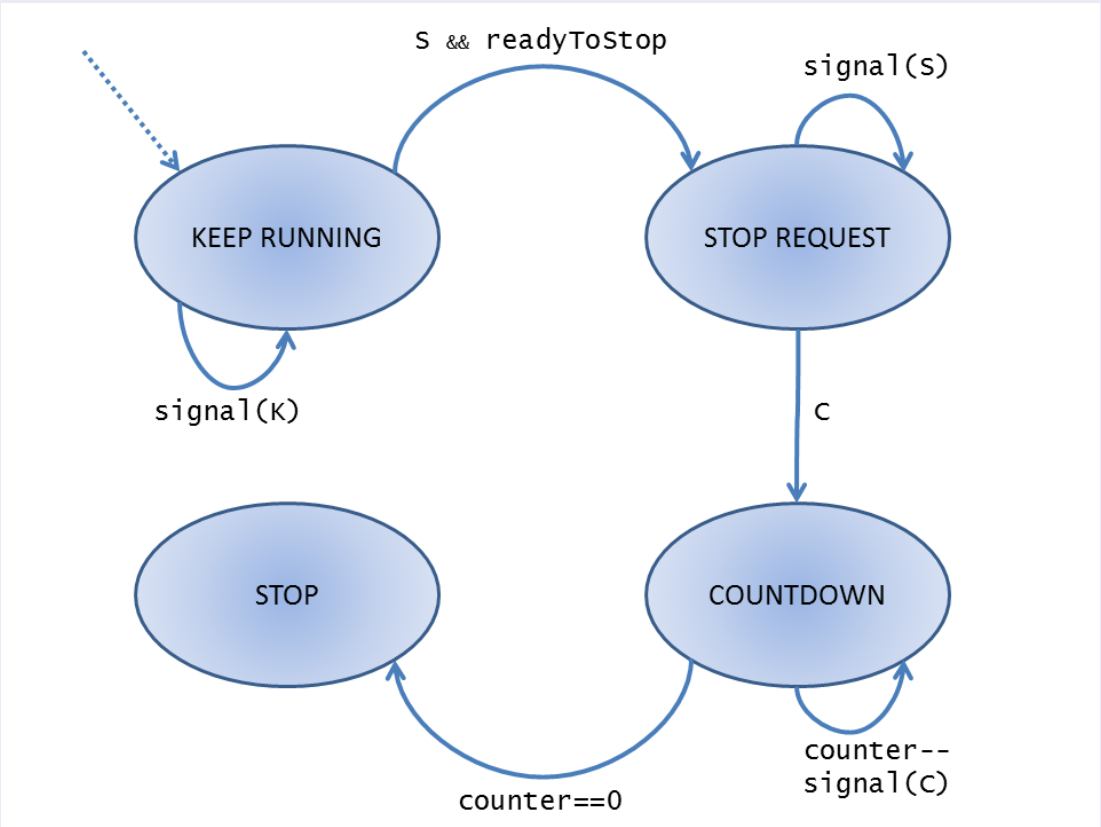
Algoritmo distribuito di terminazione

Macchina a stati per il nodo coordinatore



Algoritmo distribuito di terminazione

Macchina a stati per un nodo non coordinatore



Addestramento di una rete neurale

Problema test

Addestramento di una rete neurale feed-forward

- dataset con ingresso ed uscita unidimensionale, composto da 50 elementi
- minimizzazione della deviazione standard dell'errore sul dataset
- 4 neuroni nascosti con funzione di attivazione sigmoidale
- neurone nello strato di uscita con funzione di attivazione lineare
- il test ha solo scopo dimostrativo, e pertanto non sono state applicate tecniche di cross-validation
- il dominio di ricerca è $\mathbb{R}^{13}$

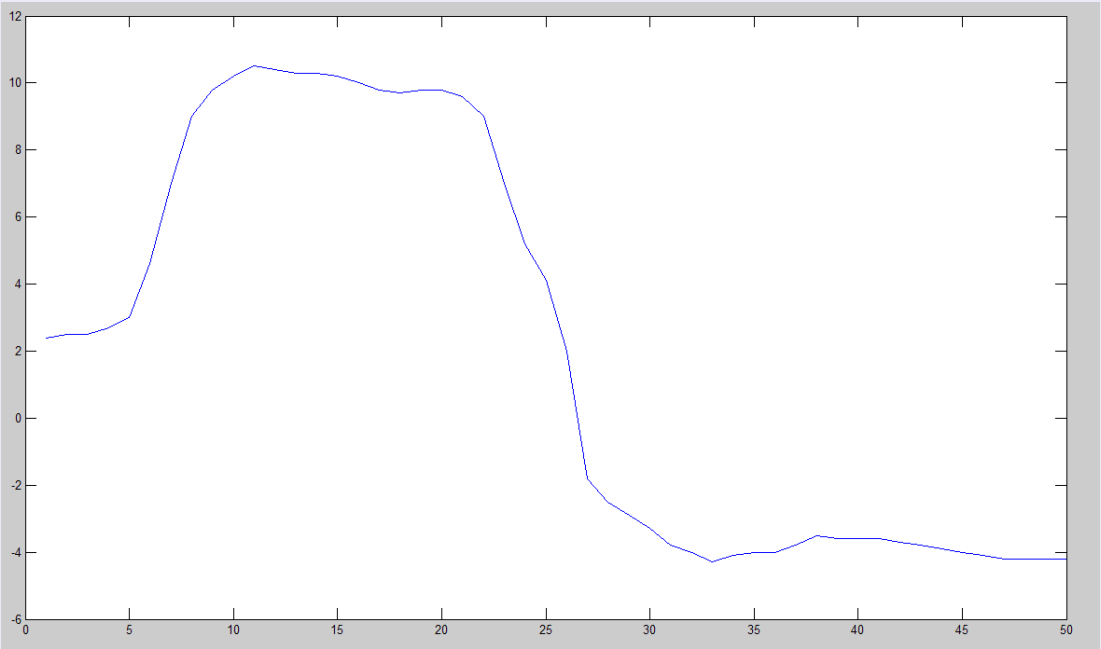
Addestramento di una rete neurale

Parametri dell'algoritmo genetico

- massimo numero di iterazioni: 5000
- crossover fraction: 0.8
- numero di elite children: 3
- migration fraction: 0.1
- migration period: 20 iterazioni
- shrink factor: 1.0

Addestramento di una rete neurale

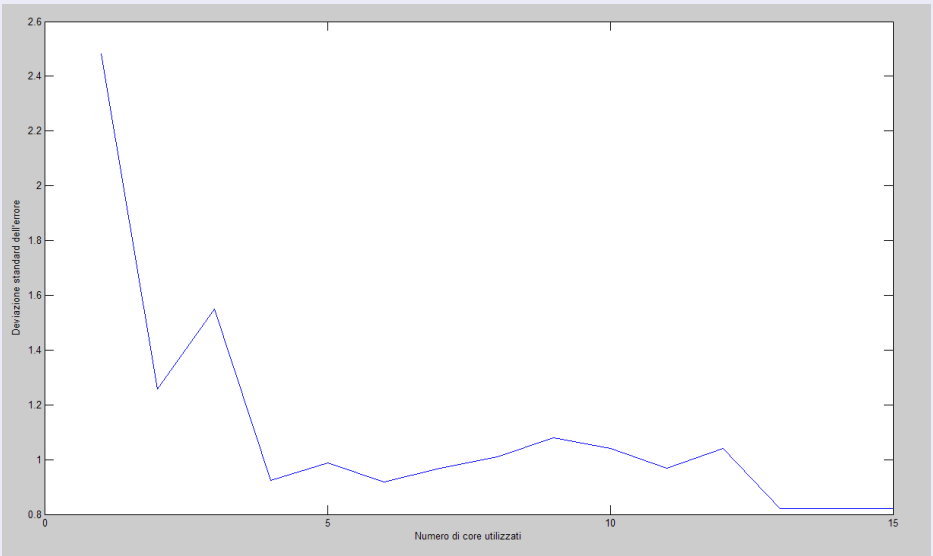
Dataset 1



Addestramento di una rete neurale

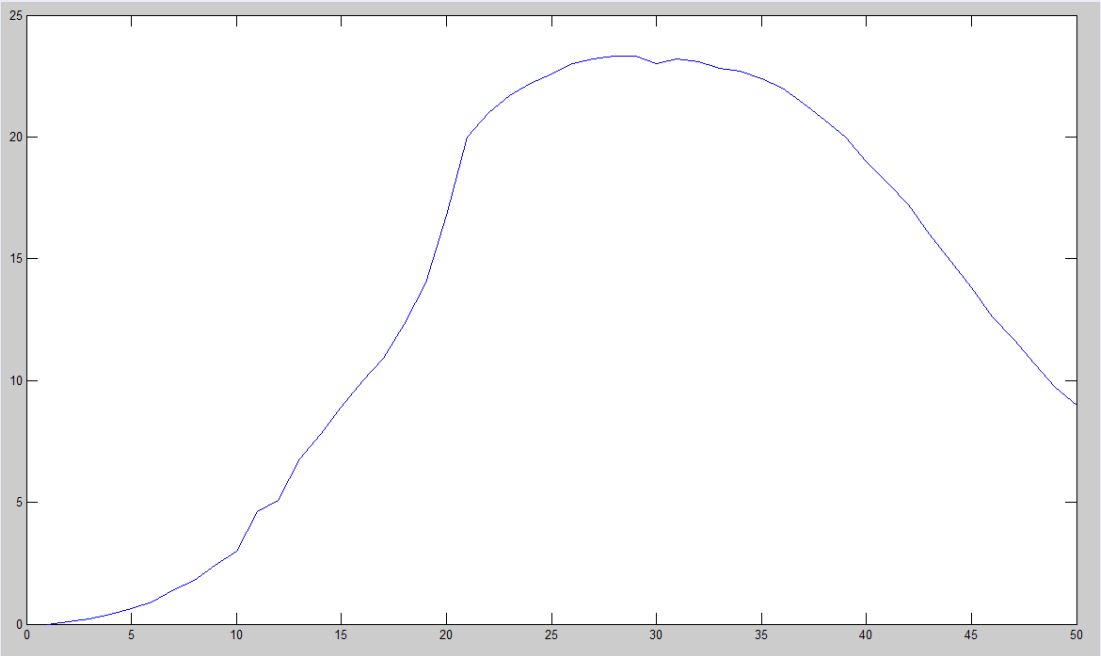
Risultati sul dataset 1

- intervallo di inizializzazione: [-10,10]
- dimensione della popolazione: 20



Addestramento di una rete neurale

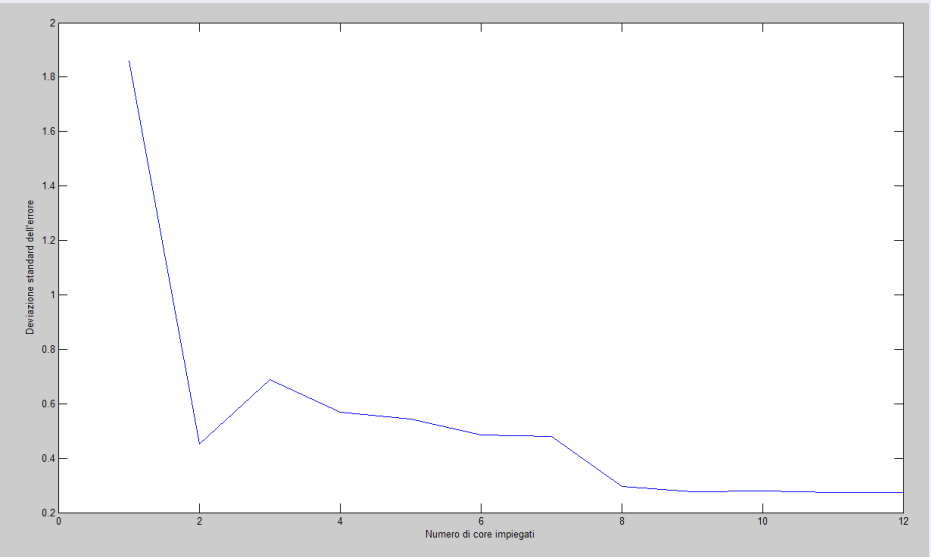
Dataset 2



Addestramento di una rete neurale

Risultati sul dataset 2

- intervallo di inizializzazione: [-30,30]
- dimensione della popolazione: 80



Addestramento di una rete neurale

Considerazioni sui risultati

- tempo di esecuzione indipendente dal numero di core impiegati
- l'intervallo di inizializzazione va dimensionato in base al numero di core e alla dimensione della popolazione
- la probabilità di convergere a buone soluzioni diminuisce se si ingrandiscono gli intervalli iniziali
- la stessa probabilità aumenta se si aumenta il numero di core e/o la dimensione della popolazione
- intervalli troppo piccoli rendono inutile l'utilizzo di tanti core
- necessità di trovare un buon compromesso