Supporto a Meccanismi di Comunicazione per Architetture Many-Core

Laurea Triennale in Informatica

Candidato:
Federico Mariti

Relatore:
prof. Marco Vanneschi

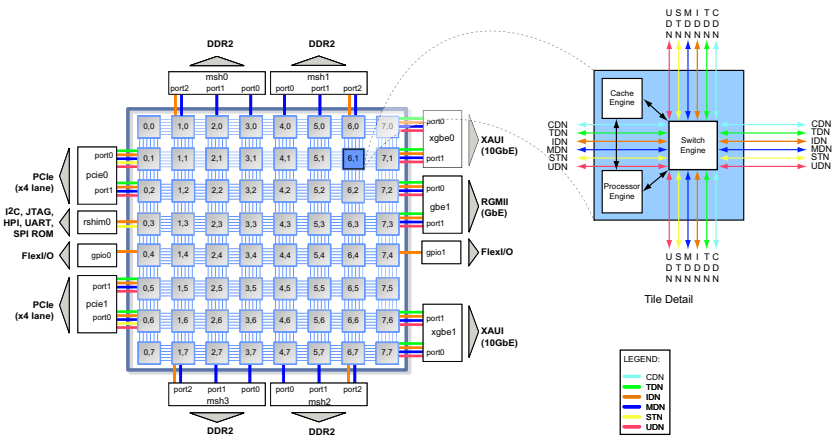
21 giugno 2013

Il lavoro svolto: obiettivi e motivazioni

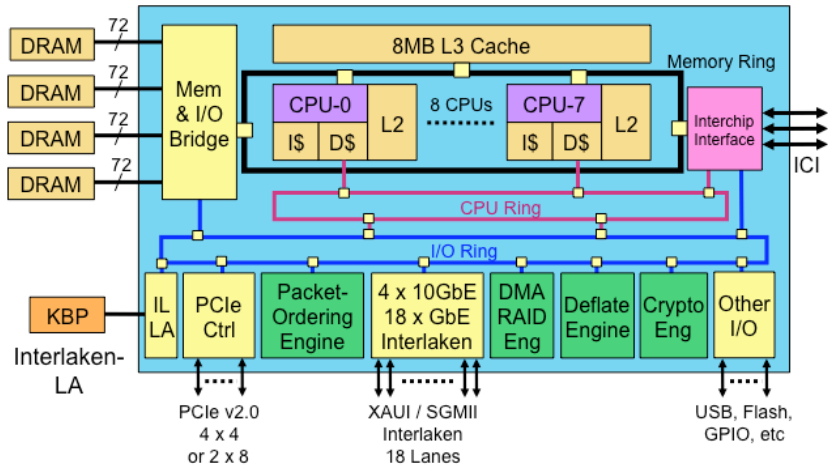
- Progettazione efficiente di un supporto alla comunicazione di processi in architetture Chip Many-Core (CMP)
- Nuovo approccio: utilizzo della Struttura di Interconnessione tra core messa a disposizione dall'architettura, piuttosto che della memoria condivisa (SM)
- L'obiettivo è la riduzione degli overhead presenti nelle tipiche implementazioni su SM:
  - latenza di sincronizzazione a strutture dati condivise in memoria
  - latenza per la gestione della coerenza della cache
- Altri vantaggi:
  - disaccoppiamento comunicazione e accesso alle informazioni in memoria condivisa
  - è possibile una forma parziale di sovrapposizione del tempo di comunicazione al tempo di calcolo, anche in assenza di un processore di comunicazione

- Nuove macchine CMP con elevato numero di core realizzano reti di interconnessione:
  - scalabili con il numero di core,
  - replicate e dedicate a scopi disgiunti,
  - una rete viene resa disponibile all'utente per comunicazioni inter-core.

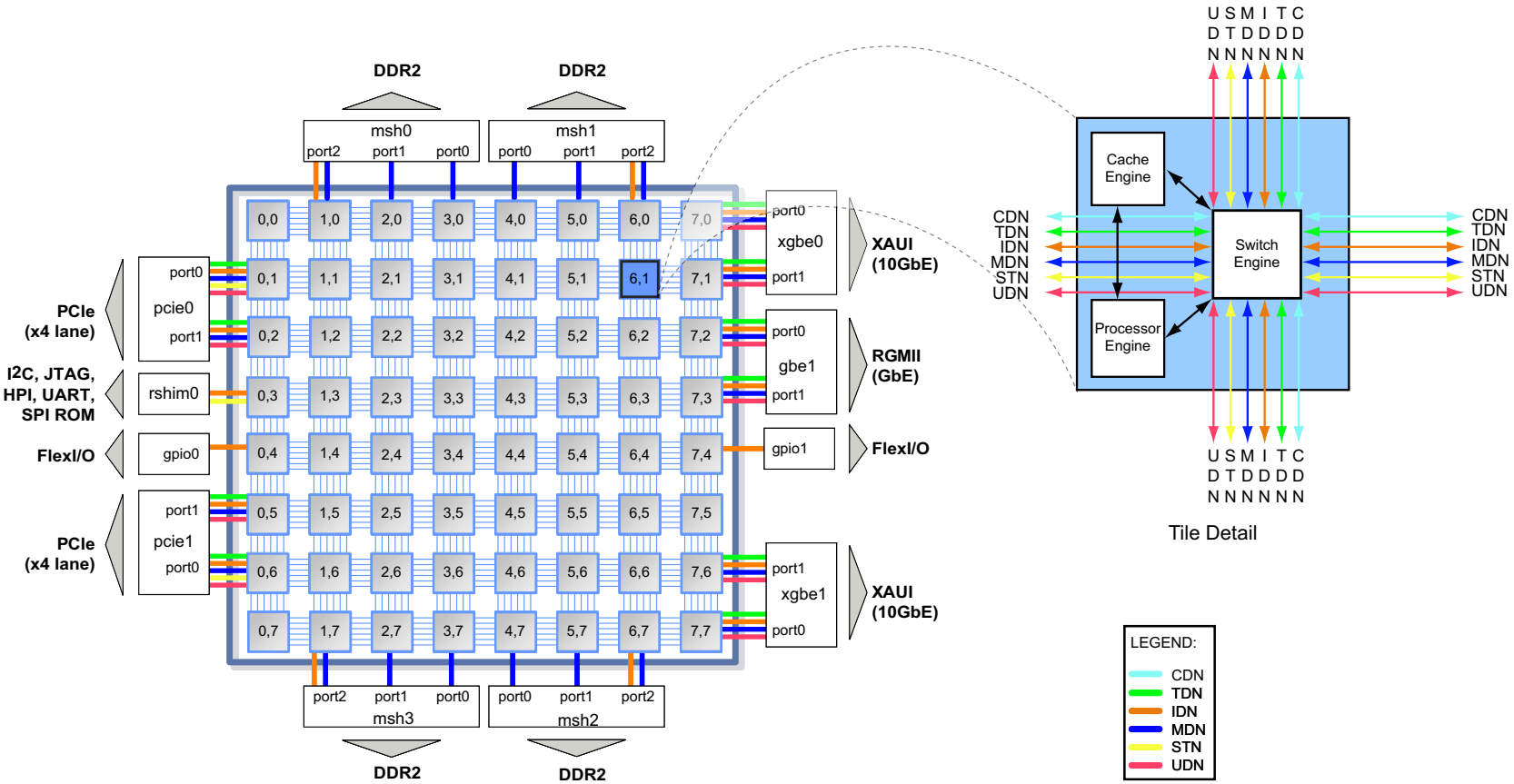
Tilera TILE*Pro*64



Netlogic XLP832



Reti di interconnessione del Tilera TILE*Pro*64



Dominio Applicativo

Computazioni con calcolo di grana fine:

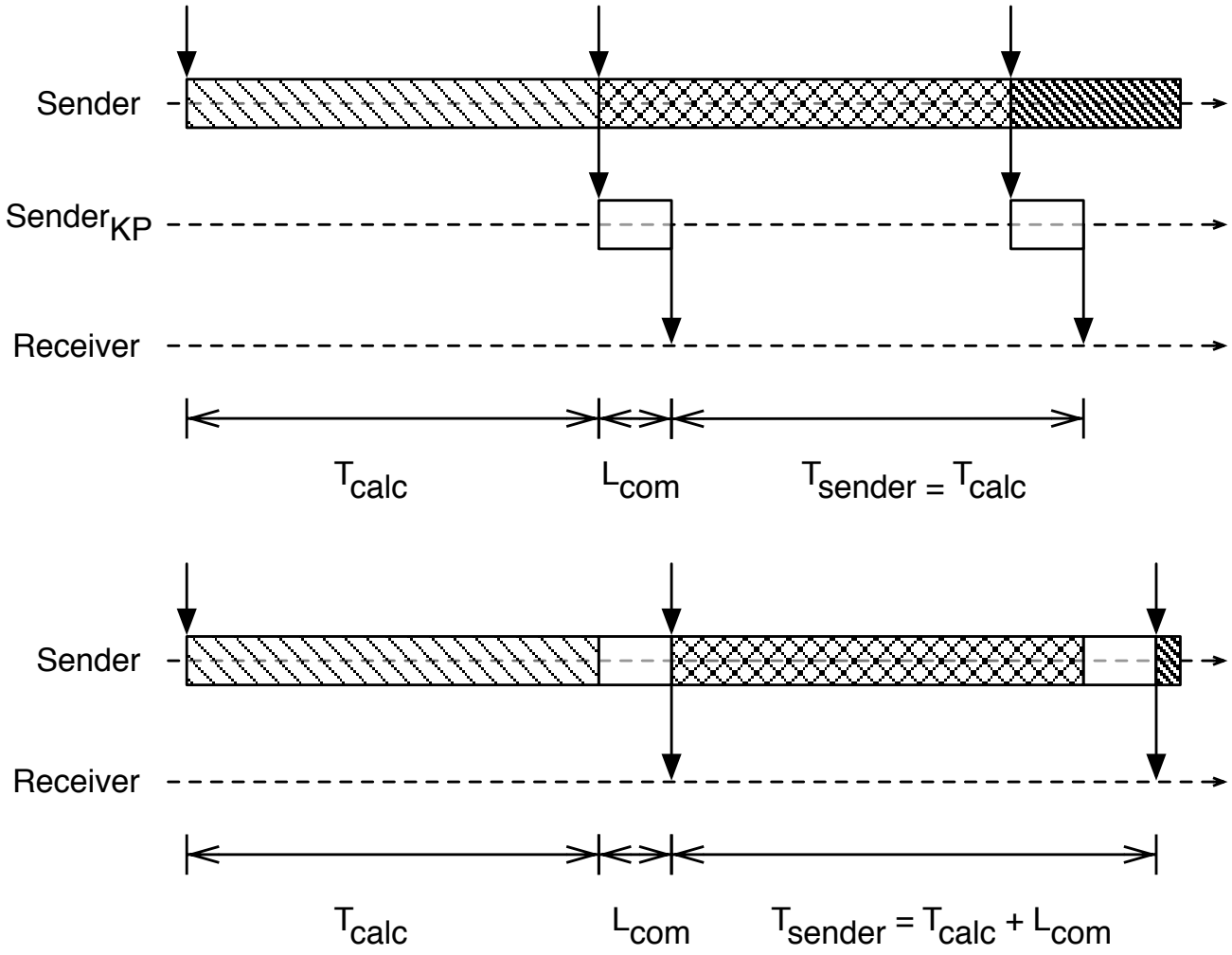
Data Stream Processing uno o più flussi contigui, rapidi e varianti nel tempo di dati; l'elaborazione sui dati in ingresso è fatta on-line:

- Monitoraggio e sicurezza di reti informatiche;
- Applicazioni finanziarie;
- Monitoraggio di sensori e Gestione delle emergenze;
- Altri . . .

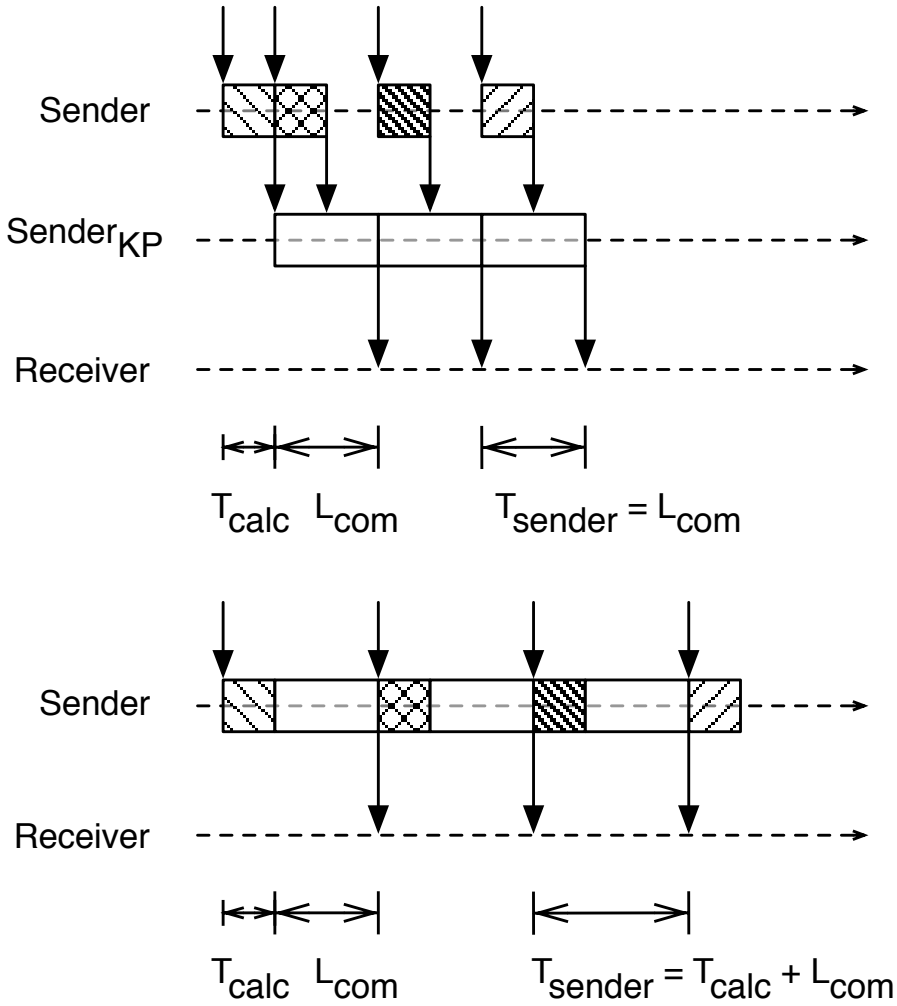
Data Parallel forma di parallelismo applicabile sia a computazioni su stream che su dato singolo; è caratterizzata dal partizionamento dei dati.

- le comunicazioni sono presenti nella realizzazione delle comunicazioni collettive e se esistono per le dipendenze sui dati (forme *Stencil*)

Sulla grana del calcolo (1)

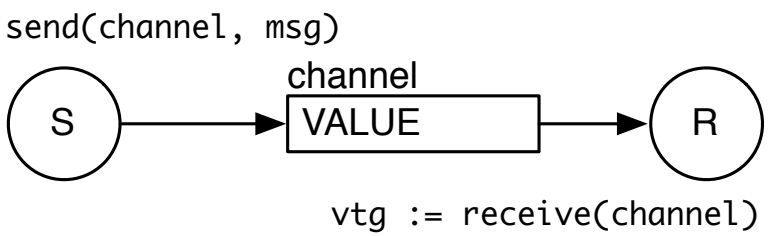


Sulla grana del calcolo (2)

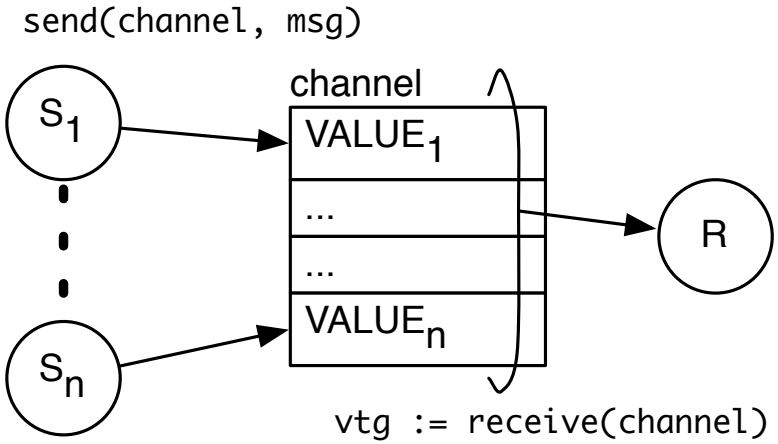


Forme di comunicazione

Canale simmetrico



Canale asimmetrico in ingresso



- Tipo 'riferimento'
- Grado di asincronia unitario
- Semantica Bloccante
- Protocollo Rdy-Ack

Implementazioni del canale simmetrico

uso della Memoria Condivisa

- `ch_sym_sm_rdyack_no`
  gestione predefinita della coerenza della cache;
- `ch_sym_sm_rdyack`
  configurazione della coerenza della cache che massimizza la località delle informazioni del supporto nei core consumatori;
- `ch_sym_sm_nullack`
  utilizzo di un diverso protocollo di comunicazione che garantisce la correttezza senza l'uso di istruzioni di barriera di memoria

uso della UDN

- `ch_sym_udn`
  corrispondenza biunivoca tra i canali firmware e i canali a livello processi

Implementazioni del canale asimmetrico in ingresso

Un canale asimmetrico in ingresso può essere visto come un caso particolare di un certo numero di canali simmetrici in ingresso al processo destinatario, sui quali viene applicato non determinismo.

uso della Memoria Condivisa

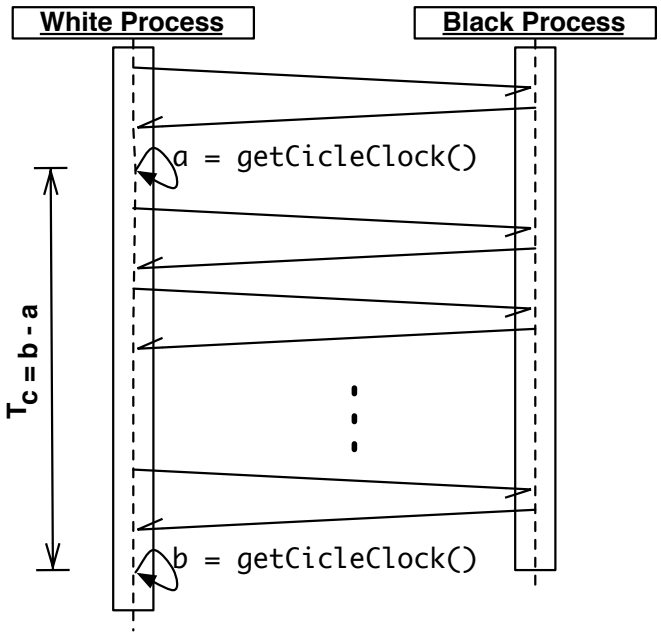
- `ch_asymin_sm`
  utilizzo del protocollo Rdy-Ack, con configurazione esplicita della coerenza della cache. Il destinatario implementa l'attesa con una lettura ciclica dei flag Rdy dei mittenti.

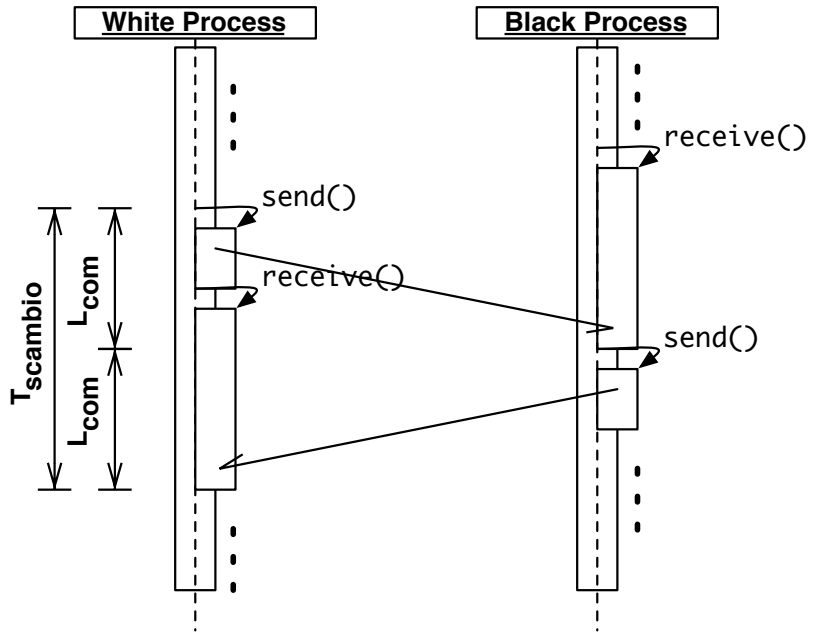
uso della UDN

- `ch_asymin_udn`
  uso di un unico canale firmware nel destinatario per la ricezione dei messaggi, uso di un canale firmware in ogni mittente per la ricezione dei segnali di Ack.

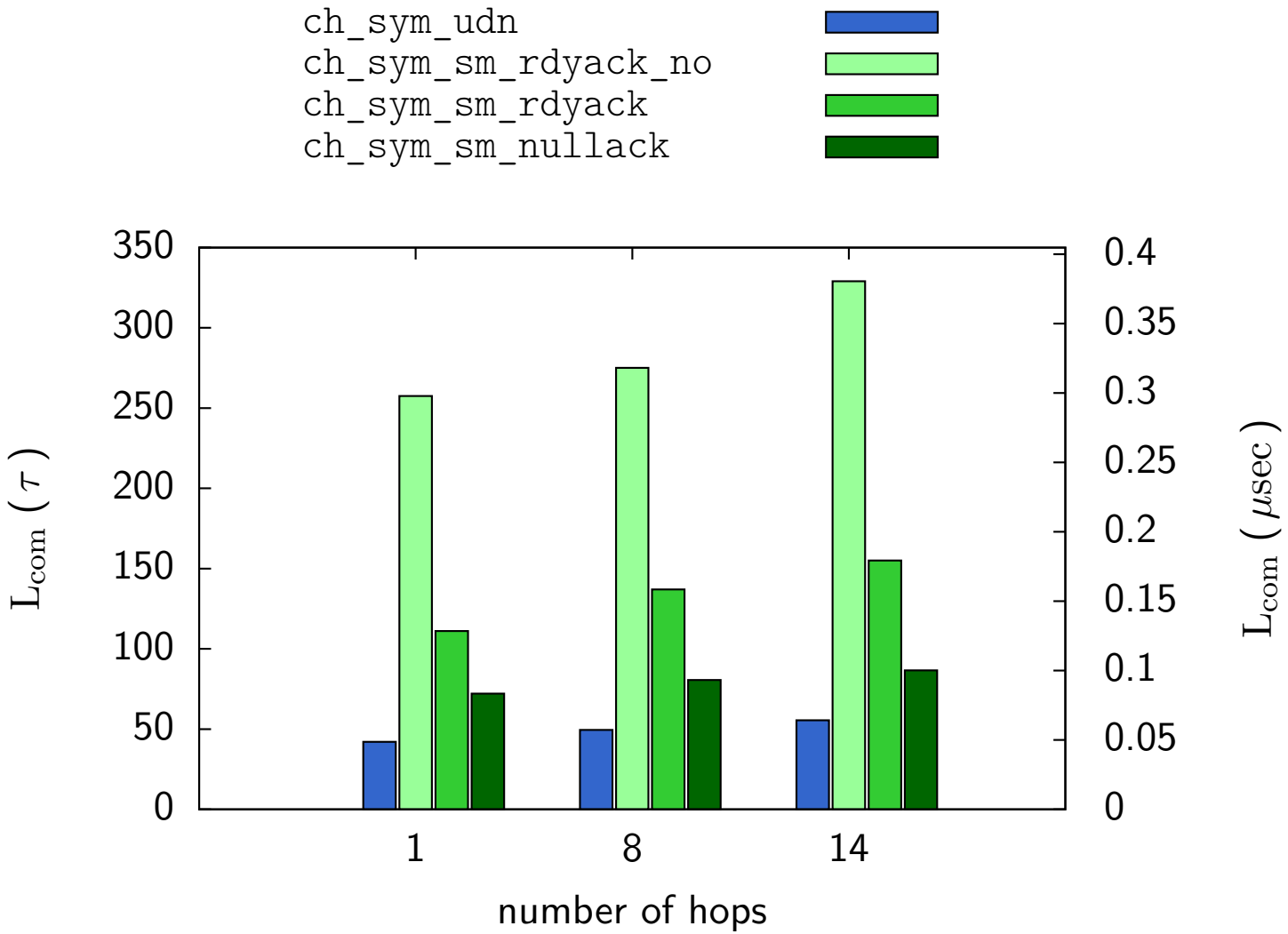
Misura della latenza di comunicazione

- La latenza di comunicazione è misurata per mezzo di una applicazione "ping-pong":
  - composta da due processi collegati da due canali,
  - viene svolto lo scambio di m messaggi tra i due processi;
- La latenza di comunicazione è stimata con $\mathrm{L_{com}} = \mathrm{T_C}/(2 \cdot m)$.

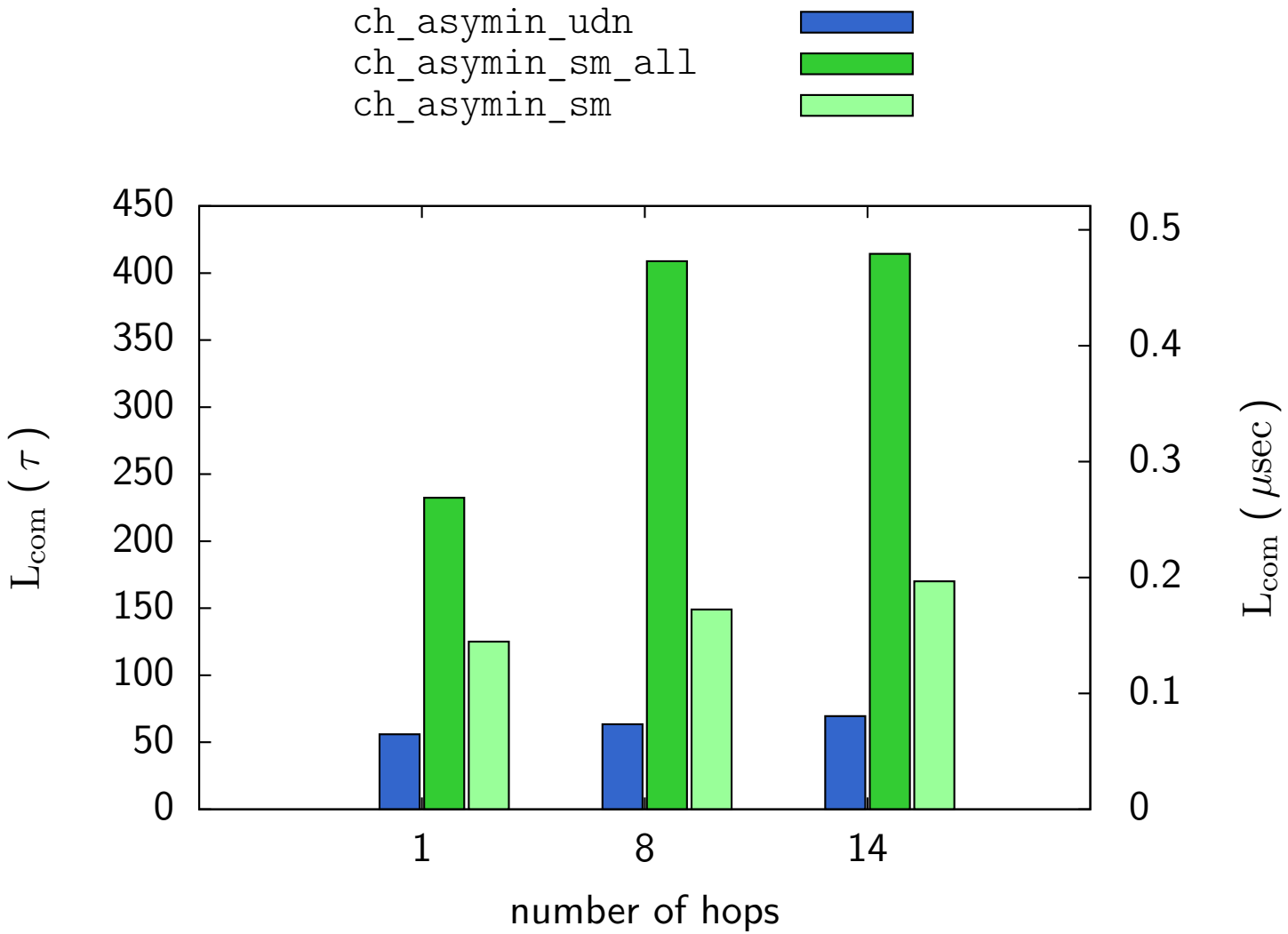




Misura della latenza del canale simmetrico



Misura della latenza del canale asimmetrico



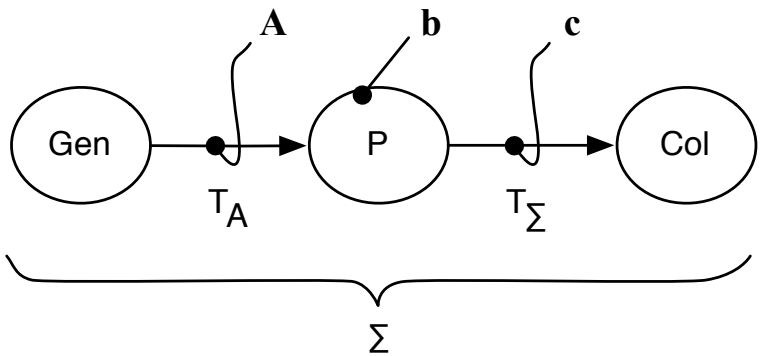
Benchmark: prodotto matrice-vettore su stream

$$\mathbf{A} = (a_{ij})_{i=1,\ldots,\mathrm{M},\, j=1,\ldots,\mathrm{M}} \in \mathbb{Z}^{\mathrm{MxM}}$$

$$\mathbf{b} = (b_1, \ldots, b_{\mathrm{M}}) \in \mathbb{Z}^{\mathrm{M}}$$

$$\mathbf{c} = (c_1, \ldots, c_{\mathrm{M}}) = \mathbf{A} \cdot \mathbf{b} \in \mathbb{Z}^{\mathrm{M}}$$

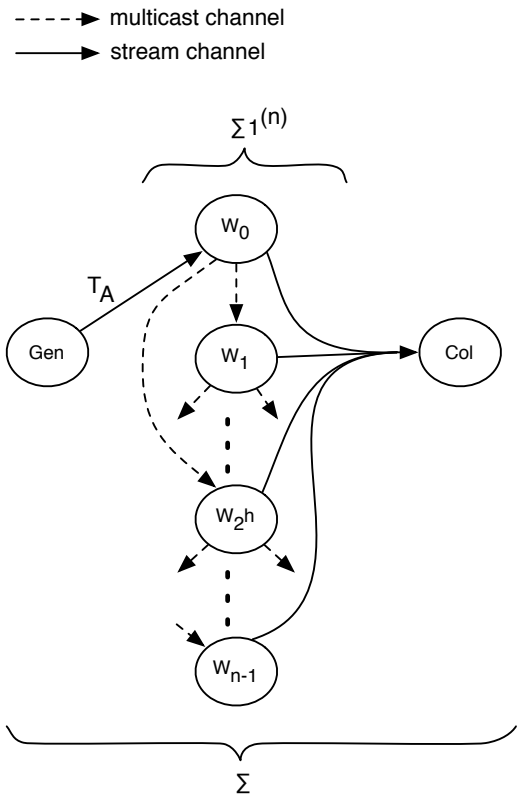
$$\forall\, i \in \{1, \ldots, \mathrm{M}\} \;:\; c_i = \mathbf{a}_i \cdot \mathbf{b} = \sum_{j=1}^{\mathrm{M}} a_{ij} \cdot b_j$$



- computazione su stream di matrici
- il vettore **b** è costante per tutta l'esecuzione

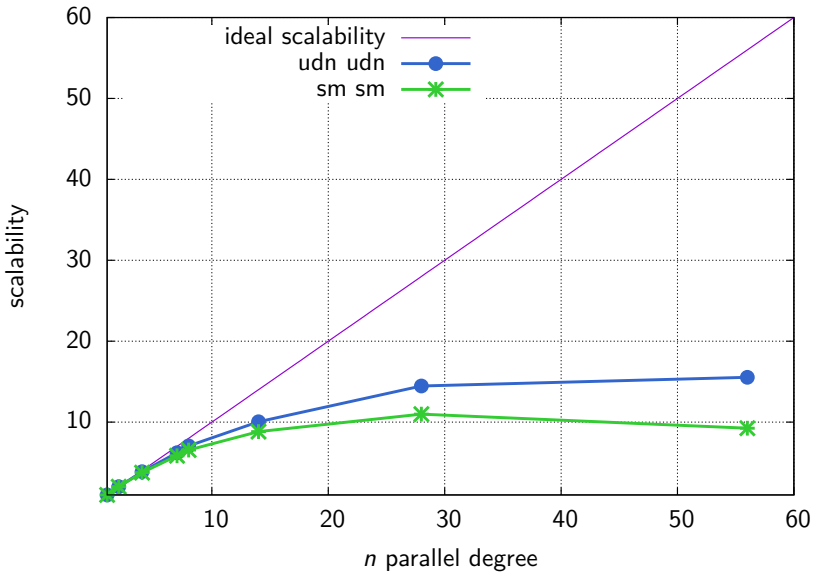
Benchmark: soluzione parallela

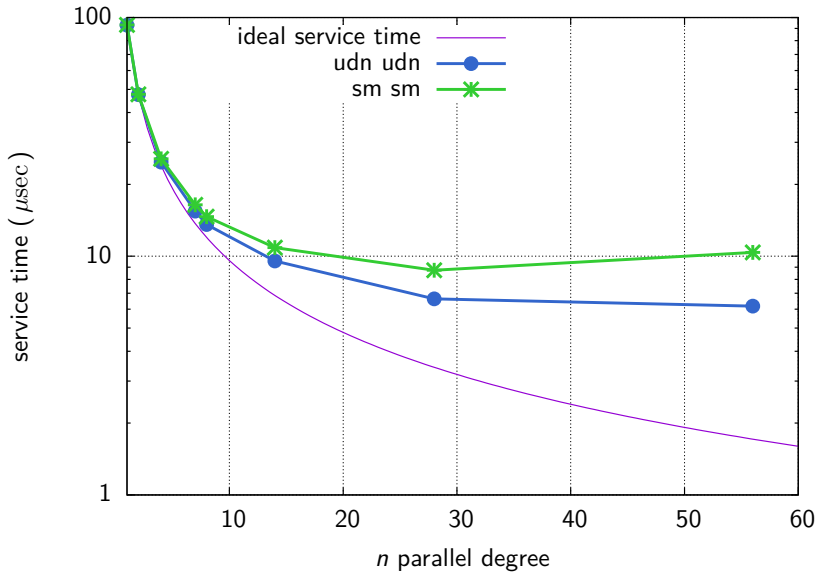
- Data Parallel Map
  - partizionamento di una matrice per righe
  - replicazione del vettore **b** nei processi worker
- Multicast strutturata ad albero distribuito nei worker
- $\mathrm{T}_{\Sigma 1\_\mathrm{id}}{}^{(n)} = 2 \cdot \mathrm{T}_{\mathrm{sym\_send}} + \mathrm{T}_{\mathrm{calc}}/n + \mathrm{T}_{\mathrm{asymin\_send}}$



Confronto: tempo di servizio (1)

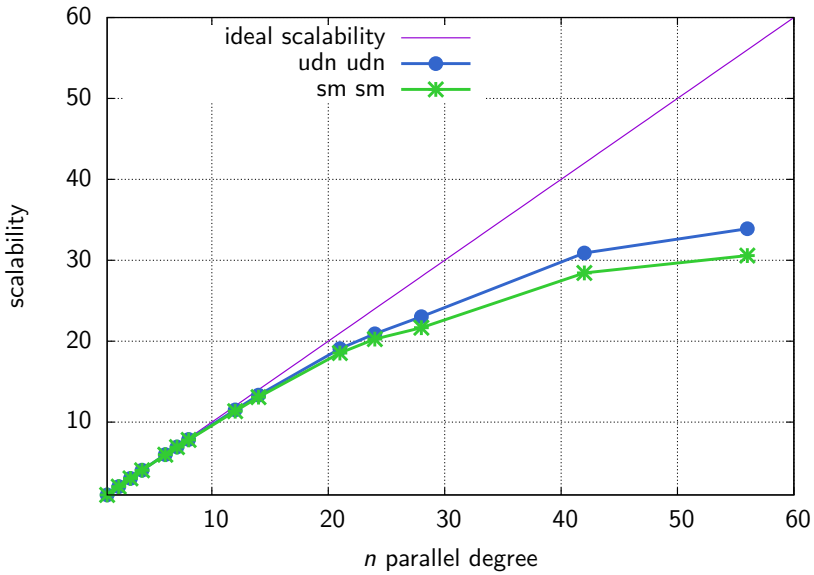
- Tempo di interarrivo 4.627 μsec
- Dimensione delle matrici 56x56
- Tempo di servizio migliore UDN 6.177 μsec
- Tempo di servizio migliore SM 8.735 μsec

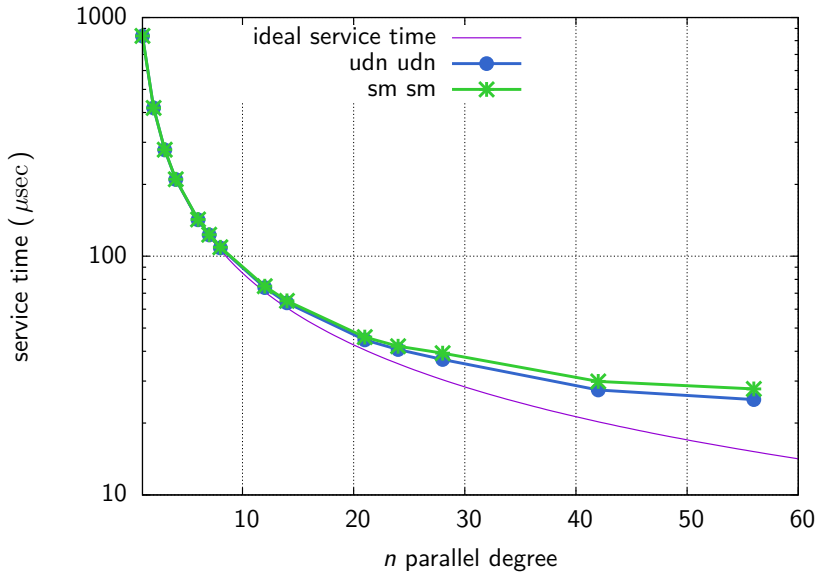




Confronto: tempo di servizio (2)

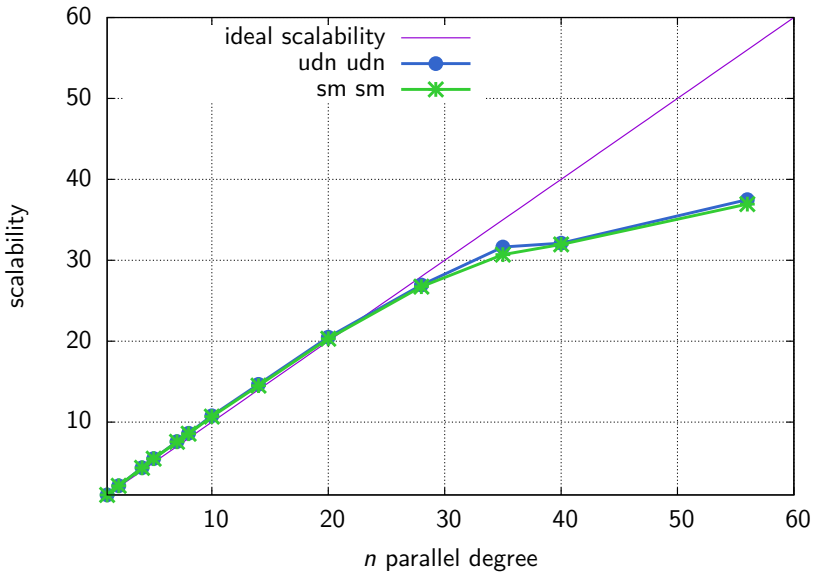
- Tempo di interarrivo 4.627 μsec
- Dimensione delle matrici 168x168
- Tempo di servizio migliore UDN 25.092 μsec
- Tempo di servizio migliore SM 27.810 μsec

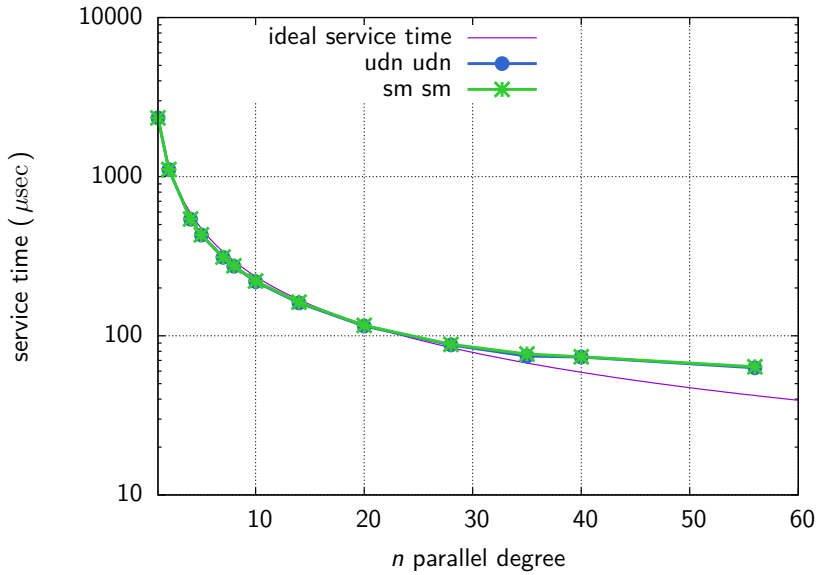




Confronto: tempo di servizio (3)

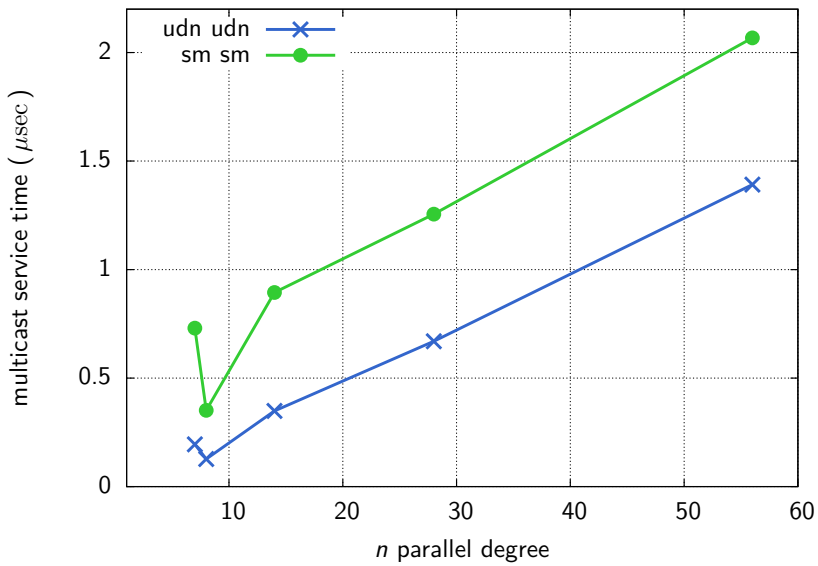
- Tempo di interarrivo 4.627 μsec
- Dimensione delle matrici 280x280
- Tempo di servizio migliore UDN 62.943 μsec
- Tempo di servizio migliore SM 63.893 μsec



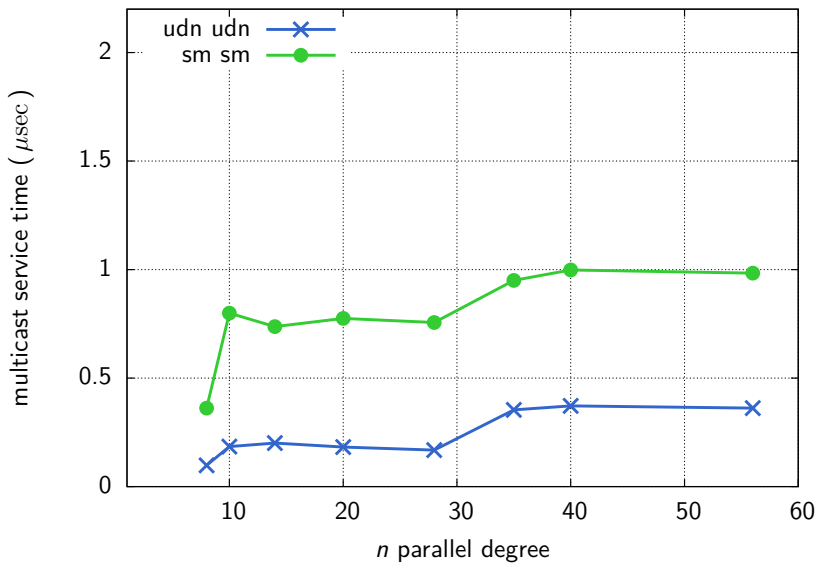


Confronto: tempo di servizio Multicast

Dimensione delle matrici 56x56

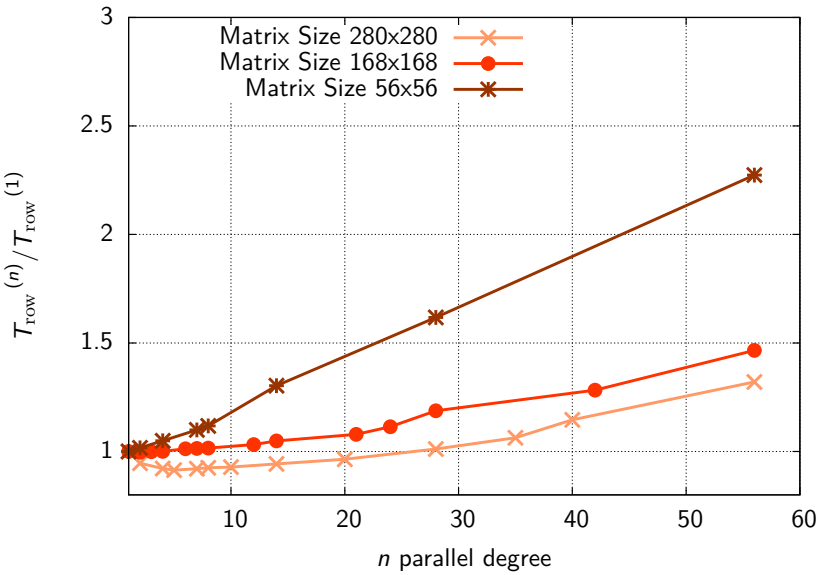


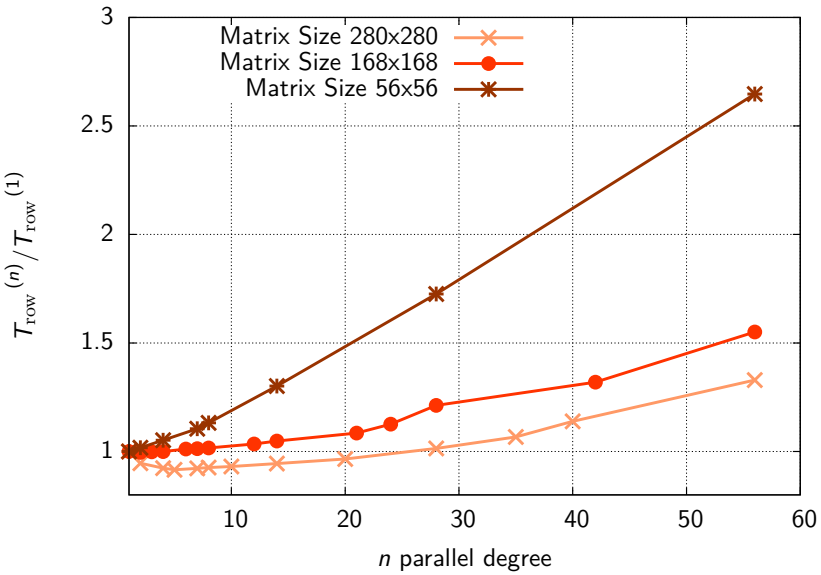
Dimensione delle matrici 280x280



Confronto: tempo di calcolo di un singolo prodotto scalare

Tipo di dato: 'Intero'





Confronto: tempo di calcolo di un singolo prodotto scalare

Tipo di dato: 'Virgola Mobile'