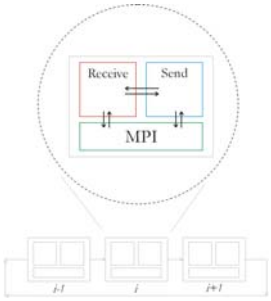
Scambio di dati

- Un'architettura di comunicazione è necessaria dal momento che ogni nodo ha un insieme di termini che non gestisce direttamente (cioè il termID non appartiene al proprio intervallo)
- Molte tipologie di comunicazione erano possibili, nel nostro caso abbiamo scelto una all-to-all personalized communication (con scambio dati a ring)
- Due tipologie sono state testate al fine di analizzare quale delle due effettua il lavoro in un tempo minore:
  - Threaded
  - Not-threaded

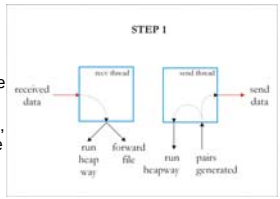
Threaded version

- Tre thread sono stati utilizzati. Tutte le chiamate MPI sono state inserite in un unico thread (MPICH 1.2.5 installato nei laboratori non è threadsafe)
- Un thread che gestisce la logica di spedizione dei dati, un altro per la logica di ricezione
- Questi due thread utilizzano due buffer locali per la ricezione e spedizione dei dati (come forma di comunicazione tra thread MPI e thread Send/Receive)
- La comunicazione tra thread R e S avviene tramite l'utilizzo di un file comune a cui vengono accodate coppie



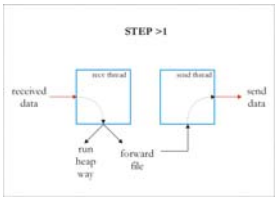
All-to-All personalized communication Threaded version (step 1)

- Sono necessari *p-1* step se *p* è il numero di nodi a disposizione.
- Durante il primo step, il thread receive fa un filtro dei dati che riceve scrivendo nel forward file le coppie che saranno utilizzate nello step successivo e nello runheapway file le coppie che sono gestite localmente.
- Il thread Send, come il thread precedente, fa un filtro delle coppie che sono generate localmente: quelle che sono gestite da altri nodi le spedisce, le altre vengono vanno accodate nei file runheapway
- Ad ogni step, viene creato un file diverso di runheapway



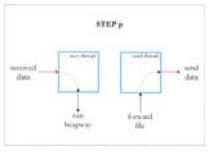
All-to-All personalized communication Threaded version (step >1)

- Gli step successivi, ultimo escluso, funzioneranno tutti allo stesso modo.
- Il thread receive effettuerà il filtro dei dati ricevuti
- Il thread send prenderà in input i dati accodati al forward file dal thread receive
- Se si è raggiuta la fine del file, il thread va in wait fino al momento in cui qualche dato è disponibile.



All-to-All personalized communication Threaded version (step p)

- Durante l'ultimo step, ogni nodo *i* riceve dati che sono partiti dal nodo *i+1* (allo step 1) e che sono già stati filtrati da tutti gli altri nodi.
- Per questo motivo, siamo sicuri che tutte le coppie ricevute apparterranno all'intervallo di gestione del nodo *i*.
- Visto sotto un'altra ottica, questo step corrisponde alla ricezione da parte di ogni nodo *i* delle coppie del nodo *i+1*, il cui termID appartiene al range del nodo destinatario (trasferimento antiorario).



Not-threaded version

- Questa versione riutilizza parte del codice scritto per la versione a thread. Utilizza due funzioni "spedizione" e "ricezione" in cui sono contenute le rispettive chiamate a funzioni MPI.
- In ogni nodo, vengono chiamate fino a quando c'e' lavoro da fare.
- Meno efficiente della versione threaded. Guadagno medio: più del 30%. Avendo avuto a disposizione una documentazione migliore e gli accessi da superuser, si poteva agire a livello di scelta della modalità di schedulazione dei thread e quindi aumentare ulteriormente il guadagno rispetto alla versione not-threaded.

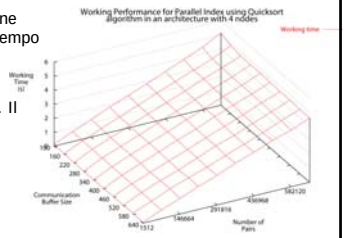
Run Dim buffer	Max CP	Comm. Dim Buffer	Guadagno %
5	Var	Var	-32,26%
Var	Var	400	-45%
Var	654.696	Var	-31,25%

Caratteristica grafici

- Tutti i dati raccolti dai test, sono stati rappresentati in grafici 3D. Le variabili ritenute significative sono:
  - Variazione del numero delle coppie [1512, 654696]
  - Variazione della dimensione del buffer di comunicazione [100, 640]
  - Variazione della dimensione del buffer di run [5, 81851]
- E' stato possibile rilevare degli andamenti caratteristici dei grafici che rappresentano la curva del tempo totale di esecuzione
- I principali sono i seguenti:
  - Lineare
  - Ondulatorio
  - A gradini
  - Cappa
  - Valori spuri

Andamento lineare

- Si presenta nel caso di ordinamento interno con quicksort, mergesort e heapsort.
- Per la dimensione dei dati da noi considerata, solamente la variazione del numero di coppie influenza il tempo totale di esecuzione.
- In alcuni casi, alcuni valori non risultano coerenti rispetto ai vicini. Il comportamento è dovuto all'imprevedibile carico della rete.

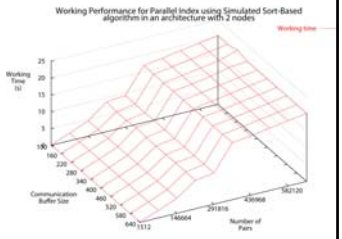


Andamento ondulatorio

- Si osserva a partire dal valore 27287.
- Deriva dalla scelta implementativa nella metodologia di ordinamento del sort-based.
- Il numero totale di run deve essere una potenza di 2. Questo valore dipende essenzialmente da due fattori:
  - Numero delle coppie (in questo test è fissato al valore massimo)
  - Dimensione del buffer di run
- Il mix tra questi due parametri, genera l'andamento ondulatorio.

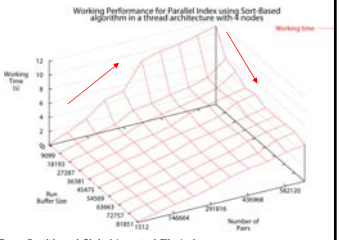
Andamento a gradini

- Nella versione simulata dell'algoritmo sort-based, la seconda fase viene fatta in memoria primaria.
- La caratteristica a gradini deriva, come nel caso precedente, dall'implementazione dell'algoritmo di fusione.
- In questo caso non c'è andamento ondulatorio come nel caso precedente perché la dimensione del buffer di run viene tenuta fissa e pari a 5 unità.



Cappa

- Andamento classico che ci si può aspettare nel momento in cui si mette nell'asse delle x la dimensione del buffer di run e nell'asse delle y il numero di coppe.
- Come ci si può aspettare, aumentando il numero di coppie il tempo totale della computazione aumenta e aumentando la dimensione del buffer, il tempo diminuisce



Speedup ed efficienza

- Valori influenzati da differenti velocità di esecuzioni in due home diverse.
- La versione a thread nettamente più veloce: confermato dal grafico.
- Il grafico deve essere trasformato se vogliamo avere nell'asse delle x il numero reale di coppie ordinate.
- Le versioni che non usano quicksort, hanno tutte lo stesso andamento
- La versione a thread non apporta miglioramenti nel quicksort