Relazione Progetto HPC 2023

Michele Montesi
Matricola: 0000974934
E-Mail: michele.montesi3@studio.unibo.it

11 aprile 2023

Introduzione

La presente ricerca presenta lo sviluppo di due versioni parallelizzate del software `sph.c`, implementate utilizzando la libreria `OpenMP` e la libreria `MPI` rispettivamente. L'obiettivo della ricerca è quello di valutare i vantaggi prodotti dall'utilizzo della programmazione multi-processore.

Per misurare le prestazioni del software in questione è stato creato uno script python che lo esegue, incrementando il numero di threads partendo da 1 fino a 12 e il numero di particelle da 1400 a 5700. Lo script trascrive su un foglio excel i seguenti dati interessati. Da questi dati viene poi calcolata la media.

Versione OpenMP

Implementazione

In questa implementazione del software, effettuata con *OpenMP*, sono stati parallelizzati attraverso `#pragma omp parallel for`, i cicli delle funzioni indicate.

Quando necessario sono state effetuate riduzioni così da rendere più efficiente l'esecuzione del software parallelizzato.

Prestazioni

Raccolta dei dati

Per misurare le prestazioni del software in questione è stato creato uno script python che lo esegue, incrementando il numero di threads partendo da 1 fino

a 12 e il numero di particelle da 1400 a 5700. Lo script trascrive su un foglio excel i seguenti dati interessati. Da questi dati viene poi calcolata la media.

Valutazione

Dalla figura 1 si nota come lo `speedup`, all'aumentare del numero di threads, cresca linearmente fino a 11, dove raggiunge il suo massimo, diminuendo, poi, a 12. Questo comportamento indica che 11 è il numero di threads oltre il quale la parallelizzazione OMP diventa inefficiente.

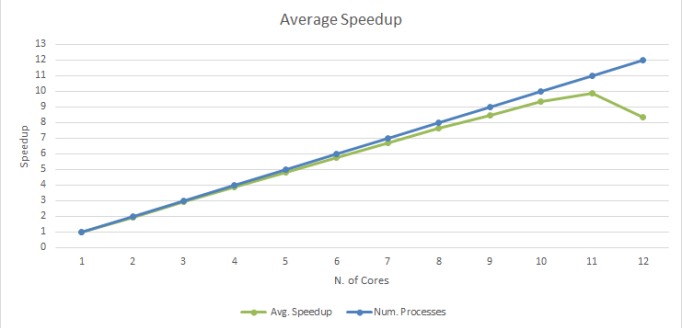

Figura 1: OpenMP implementation's average speedup

Dal grafico in figura 2 si osserva che l'efficienza resta approssimativamente la stessa tra 2 e 10 threads, diminuendo di poco con 11 e di molto con 12. La *Strong Scaling Efficiency* indica come all'aumentare dei threads, per uno stesso carico di lavoro, il tempo diminuisca. In questo caso fino a 11 threads l'efficienza è ragionevole.

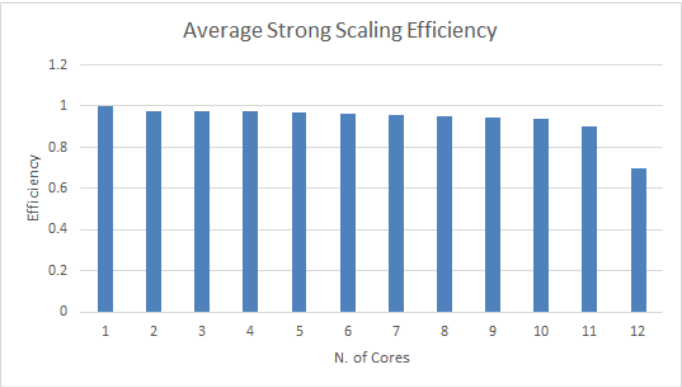

Figura 2: OpenMP implementation's average strong scaling efficiency

Nella figura 3 viene esposto l'andamento della *Weak Scaling Efficiency*, la quale all'aumentare dei processori aumenta la quantità di lavoro proporzionalmente,

in modo da considerare se i problemi con più threads vengono risolti nello stesso tempo del problema lineare. In questo caso fino a 11 threads il tempo è all'incirca lo stesso, mentre con 12 il tempo aumenta.

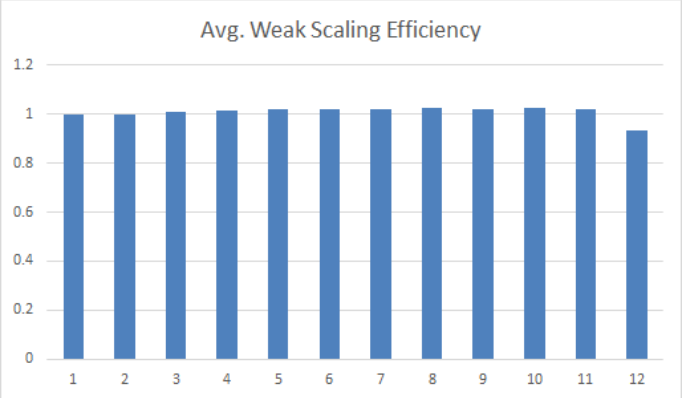

Figura 3: OpenMP implementation's average weak scaling efficiency

Versione MPI

Implementazione

Per l'implementazione del software con `MPI` è stato creato un `MPI_Datatype` contiguo per incapsulare la struttura dati durante lo scambio di messagi. Per suddividere le particelle fra tutti i processori viene fatta una `MPI_Scatterv` dal processo 0. Dopo `compute_density_pressure` e `integrate` viene chiamata una `MPI_Allgatherv` per raccogliere i dati elaborati e ridistribuirli nuovamente a tutti i processori mentre per `compute_forces` non viene fatto in quanto questo agisce solamente in locale. Alla fine, dopo aver calcolato la velocità media delle particelle locali, viene eseguita una `MPI_Reduce` con operatore somma. Questo risultato viene diviso per tutti i processi ottenendo così la velocità media complessiva.

Prestazioni

Conclusioni