

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 valutare le prestazioni del software, è stato creato uno script in linguaggio Python che esegue il programma con un numero crescente di thread, partendo da 1 e arrivando a 12, e con un numero di particelle che varia da 1400 a 5700. Lo script registra i dati rilevanti in un foglio di lavoro Excel, dai quali verrà calcolata la media. Queste informazioni saranno poi utilizzate per valutare le prestazioni.

I software sono stati testati sul server `ISI-Raptor`

Versione OpenMP

Implementazione

In questa implementazione del software, effettuata con *OpenMP*, sono state parallelizzate le sezioni di codice che richiedono maggiore elaborazione utilizzando la direttiva `#pragma omp parallel for`.

Dove necessario sono state effettuate riduzioni al fine di ottimizzare le prestazioni del software parallelizzato.

Prestazioni

Dalla figura 1 si nota come all'aumentare del numero di thread, lo *speedup* aumenti in modo significativo, come si può notare dall'aumento della curva. Tuttavia, dopo aver raggiunto 8 thread, si può notare una fase di saturazione, in cui l'aumento del numero di thread non porta più a un miglioramento significativo delle prestazioni. In questo punto, l'utilizzo di thread aggiuntivi potrebbe peggiorare le prestazioni.

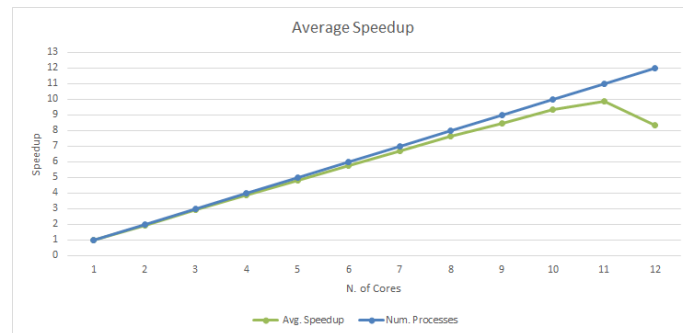


Figura 1: OpenMP implementation's average speedup

Dal grafico in figura 2 si può notare come la *Strong Scaling Efficiency* raggiunga il suo picco massimo per circa 6 thread, corrispondente a un'efficienza di circa il 90% rispetto all'efficienza ideale. Dopo il picco, questa inizia a diminuire, il che significa che l'aggiunta di thread aggiuntivi non porta a un miglioramento significativo delle prestazioni, e può addirittura causare un degrado delle prestazioni complessive.

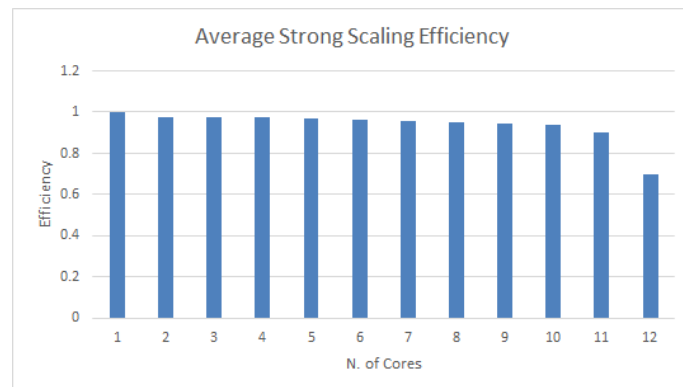


Figura 2: OpenMP implementation's average strong scaling efficiency

Nella figura 3 si può notare come la *Weak Scaling Efficiency* rimanga costante all'aumentare della dimensione del problema. Questo indica che il software è in

grado di gestire in modo efficiente problemi di dimensioni diverse, distribuendo il carico di lavoro tra le CPU disponibili senza una diminuzione significativa delle prestazioni. Questa costante è un'indicatore di una buona scalabilità del software. A 12 thread, l'efficienza debole cala, indicando che l'aggiunta di ulteriori CPU non comporta migliori prestazioni.

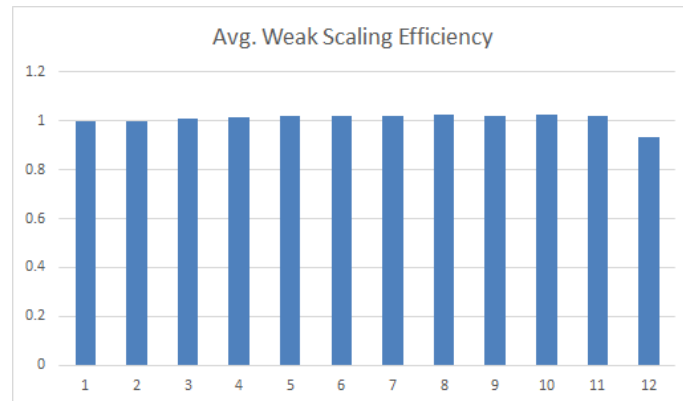


Figura 3: OpenMP implementation's average weak scaling efficiency

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 messaggi. 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