Relazione progetto MPI (APSD)

Giorgio Andronico (mat.189614)

Premessa:
Tutti i test sono stati svolti su un cluster di calcolatori composto da due nodi e collegati in LAN (IP statico) mediante un cavo ethernet crossover (no switch).
Nodo 1: Intel Core i5-560M, 4-core @2.6GHz - 4GB RAM (1333MHz DDR3)
Nodo 2: AMD Vision A6, 2-core @2.7GHz - 4GB RAM (1333MHz DDR3)
Entrambi i nodi usano Linux come sistema operativo. Lo scambio di dati viene effettuato mediante l'uso di una partizione NFS condivisa in LAN (creata collegando due pc con un cavo ethernet crossover e assegnando IP statici ad ogni nodo), e la comunicazione avviene mediante OpenSSH.
MPICH 2 è la versione su entrambi i nodi di MPI.

Breve descrizione delle metodologie impiegate per implementare il "Game of Life"

Ho deciso di implementare il "Game of Life" mediante il meccanismo "scatter-gather" fornito da MPI. Solo il master inizializza la matrice completa, mentre ogni processo (master incluso) ha una sua matrice "locale" orlata da ghost cells. (Ogni processo ha due righe fantasma.) Ad ogni iterazione generazionale, ogni processo lavora sulla sua matrice locale, e finita l'iterazione, il master fa il gather della matrice, per disegnarla usando semplici routine della libreria *Allegro.h*. Vengono anche aggiornate le ghost rows di ogni processo.
L'allocazione della memoria è completamente dinamica (uso dell'operatore *new),* e la griglia è completamente toroidale. Non ci sono vincoli sul numero di processi con cui lanciare il programma.

Tempistiche, scalabilità, efficienza

Tutti i test sono stati eseguiti usando entrambi i nodi del cluster. Prima provando solo il master, e poi congiunti master+slave.

Matrice 180x280, 1000 generazioni – tempi di esecuzione

	Nodi del cluster	
Processi	Uno	Due
1	27.66	N/A
2	15.74	69.00
4	14.90	62.00
6	127.00	79.00
8	127.00	75.00
12	185.00	135.00

Matrice 180x280, 1000 generazioni – speedup

	Nodi del cluster	
Processi	Uno	Due
1	1.00	N/A
2	1.76	0.40
4	1.86	0.45
6	0.22	0.35
8	0.22	0.37
12	0.15	0.20

Matrice 180x280, 1000 generazioni – efficienza

	Nodi del cluster	
Processi	Uno	Due
1	1.00	N/A
2	0.88	0.2
4	0.47	0.11
6	0.03	0.05
8	0.03	0.046
12	0.02	0.016

Osserviamo che, facendo girare il programma solo sul master, la scalabilità è buona. Le prestazioni naturalmente decrementano dopo i 4 processi, ma è normale perché il computer è solo un quad core. Dai 6 processi in su, le prestazioni su macchina singola peggiorano notevolmente: troppo overhead e comunicazione inutile. Facendo girare il programma invece in maniera congiunta sullo slave e sul master, osserviamo che fino a 4 processi le prestazioni sono peggiori rispetto a quelle master-only, sicuramente dovuto al fatto che la comunicazione tra processi è più lenta: in questo caso avviene su un cavo ethernet gigabit, mentre se sono solo sul master, la comunicazione avviene sui bus di memoria, che sono molto molto più veloci ovviamente. Dai 6 processi in su, però, il cluster ha prestazioni migliori della macchina singola, perché si, c'è overhead e comunicazione aggiunta, però almeno è divisa su due macchine. Il lavoro di compuazione extra di MPI viene diviso su più core.