Benchmarks
- Actualizado (01.12.2005)

Benchmarks Subsistema Beowulf

- Linpack
- Pallas MPI Benchmark PMB
- CPMD Si64

Linpack
No test Linpack feito en Xullo do 2002, o resultado obtido foi de 10,3 GFlops. O benchmark foi compilado cos compiladores de Portland versión 3.3-2, GM 1.5.1_Linux e MPICH-GM 1.2.1..7b.

Pallas MPI Benchmark PMB
A continuación recóllense os resultados obtidos dos benchmarks de Pallas GmbH. Estos benchmarks empreganse para comparar o rendemendo das implementacións MPI. Os resultados obtivéronse en Xullo do 2002 coa versión GM 1.5.1_Linux e a versión MPICH-GM 1.2.1..7b, baseada en MPICH 1.2.1. O benchmark PMB foi compilado cos compiladores de Portland pgcc, versión 3.3-2.
Rendemento Punto a Punto
O rendemento punto a punto midese entre dous procesos que execútanse en dous nodos distintos, e se expresan en MBytes por segundo de ancho de banda por nodo (enviar + recibir), así como a latencia nas comunicacións expresada en microsegundos.
PMB PingPong

PingPong é o patrón clásico empregado para medir o arranque e o throughput dunha única mensaxe enviada entre dous nodos. A secuencia de comunicación é un bucle MPI_Recv() seguida por un MPI_Send() (figura 1).

PMB PingPing

PingPing é un test de comunicación en dous direccións concurrentes. Ao igual que PingPong, PingPing mide o arranque e o throughput dunha única mensaxe enviada entre dous procesos, coa diferencia de que as mensaxes están obstruidas por mensaxes na dirección contraria. Para elo, dous procesos comunícanse (MPI_Isend/MPI_Recv/MPI_Wait) entre sí, con chamadas MPI_Isend enviadas simultáneamente (figura 1).

PMB SendRecv

Este test está baseado na chamada MPI_Sendrecv(). Nél os procesos forma unha cadea de comunicación periódica, na que cada proceso envía ao veciño que se atopa a súa dereita e receve do veciño que se atopa a súa esquerda na cadea (figura 2).

PMB Exchange

Neste test, o grupo de procesos tamén actúa coma unha cadea periódica, e cada proceso intercambia (exchanges) datos cos seus veciños dereito e esquerdo na cadea (figura 3).

Benchmarks colectivos

O rendemendo colectivo ou da interconexión do sistema no seu conxunto, mídese entre todos ou un subconxunto dos nodos do sistema. Os datos dos benchmarks colectivos amosan a latencia media nas comunicacións en microsegundos.
PMB Allreduce

Este é o benchmark da función MPI_Allreduce. Reduce vectores de números en punto flotante de lonxitude L = X/sizeof(float) desde cada proceso a un único vector e o distribue a todos os procesos. O tipo de datos MPI é

MPI_FLOAT e a operación MPI é MPI_SUM (figura 4).

PMB Reduce

Este é o benchmark da función MPI_Reduce. Reduce vectores de números en punto flotante de lonxitude L = X/sizeof(float) desde cada proceso a un único vector no proceso pai. O tipo de dato MPI é MPI_FLOAT e a operación MPI é MPI_SUM. O proceso pai da operación cambia cíclicamente (figura 5).

PMB Reduce_scatter

Este é o benchmark da función MPI_Reduce_scatter. Reduce vectores de números en punto flotante de lonxitude L = X/sizeof(float) nun único vector. O tipo de dato MPI é MPI_FLOAT e a operación MPI é MPI_SUM. Na fase dispersa (scatter), os valores de L divídense uniformemente entre todos os procesos (figura 6).

PMB Allgather

Este é o benchmark da función MPI_Allgather. Cada proceso envía X bytes e recibe o grupo dos X*(nº_procesos) bytes (figura 7).

PMB Allgatherv

Este é o benchmark da función MPI_Allgatherv. Cada proceso envía X bytes e recibe o grupo dos X*(nº_procesos) bytes (figura 8).

PMB Alltoall

Este é o benchmark da función MPI_Alltoall. Cada proceso envía e recibe X*(nº_procesos) bytes (X para cada proceso) (figura 9).
PMB Broadcast

Este é o benchmark da función MPI_Bcast. Un proceso pai envía (broadcasts) X bytes a todos os outros procesos (figura 10).

PMB Barrier

Este é o benchmark da función MPI_Barrier(). Non se intercambia ningún dato (figura 11).

CPMD Si64

Para proba-lo BeoWulf cunha aplicación real, compilouse o programa de dinámica molecular CPMD neste sistema utilizando MPI. O compilador utilizado foi o de Portland e utilizaronse tres librerías diferentes de BLAS:

- As propias do compilador de Portlan (marcadas nas figuras como Pallas)

- As do proxecto Atlas

- As propias do fabricante (Intel)

Como entrada utilizouse un coxunto de 64 moléculas de Si sobre o cal fíxose unha optimización da función de onda (marcado como Si64 nas fuguras) e, nun segundo paso, un cálculo ad initio MD (marcado como Si64md). Nas figuras seguintes amósasen os resultados obtidos tanto en tiempo consumido como en SpeedUp en funció do número de procesadores utilizados.

Figura 12.- CPDM si64
Figura 13.- CPDM si64md
Figura 14.- CPDM si64 speedup
Figura 15.- CPDM si64md speedup