

BENCHMARKS SUBSISTEMA BEOWULF

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

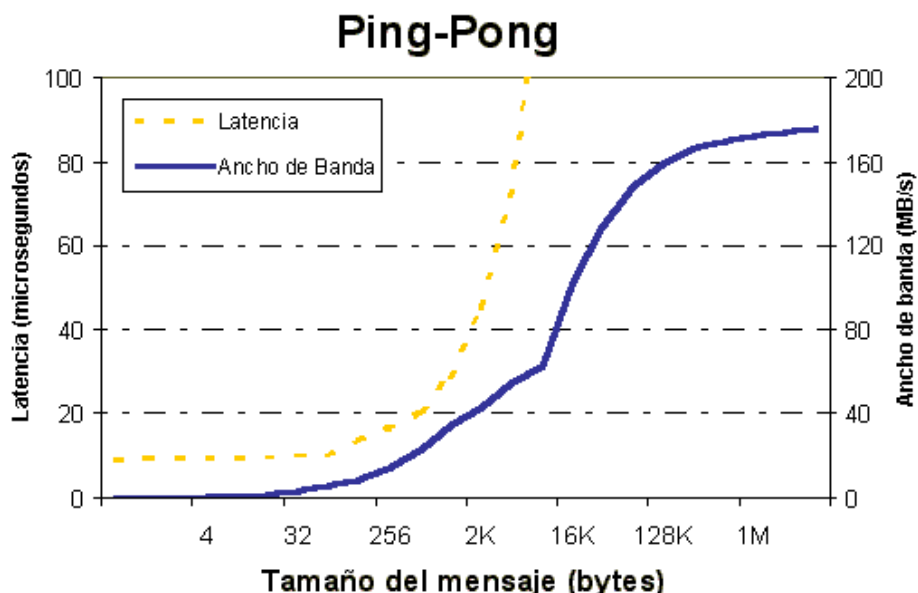
A continuación recóllense os resultados obtidos dos benchmarks de Pallas GmbH. Estes benchmarks empreganse para comparar o rendemento das implementacións MPI. Os resultados obtivéronse en Xullo do 2002 coa versión GM 1.5.1_Linux e a versión MPICH-GM 1.2.1..7b, baseada en MPICH 1.2.1. O benchmark PMB foi compilado cos compiladores de Portland pgcc, versión 3.3-2.

Rendemento Punto a Punto

O rendemento punto a punto mídese 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

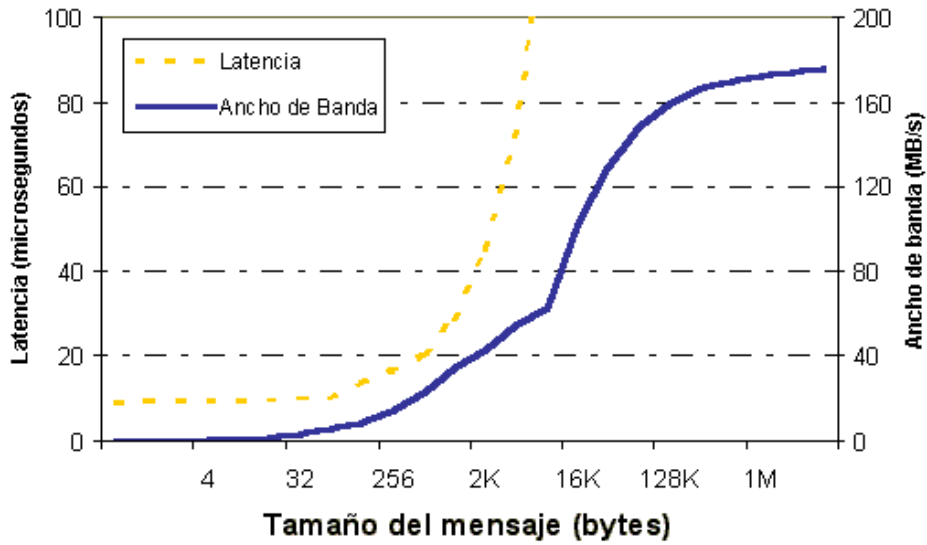
PingPong é o patrón clásico empregado para medir o arranque e o throughput dunha única mensaxe enviada entre dous nodos. A secuencia de comunicación é un bucle MPI_Recv() seguida por un MPI_Send().



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 diferenza de que as mensaxes están obstruídas por mensaxes na dirección contraria. Para elo, dous procesos comunícanse (MPI_Isend/MPI_Recv/MPI_Wait) entre sí, con chamadas MPI_Isend enviadas simultaneamente.

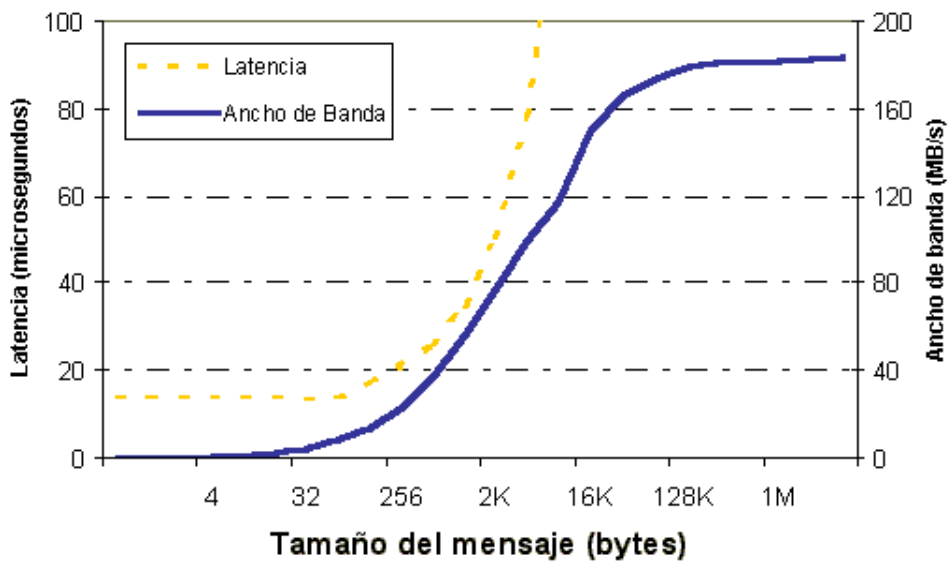
Ping-Pong



PMB SendRecv

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

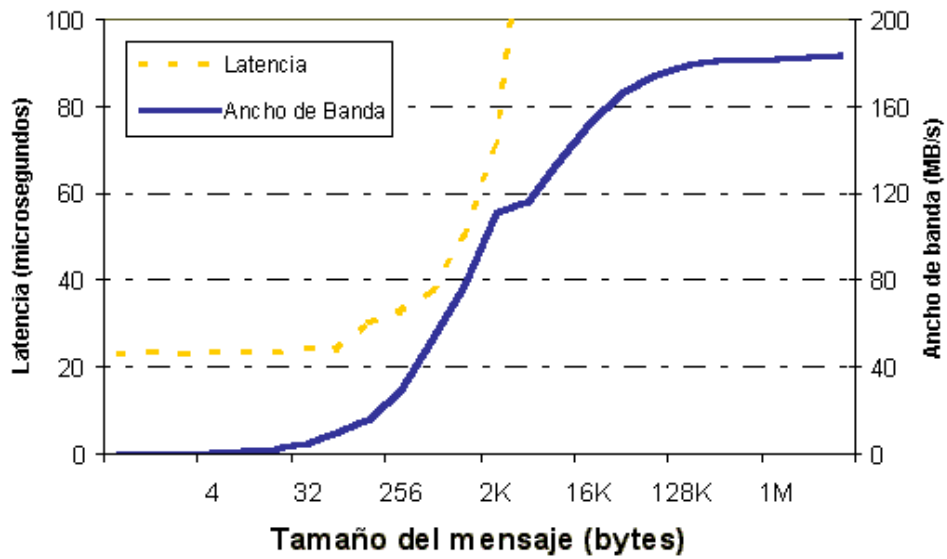
Send-Recv



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.

Exchange



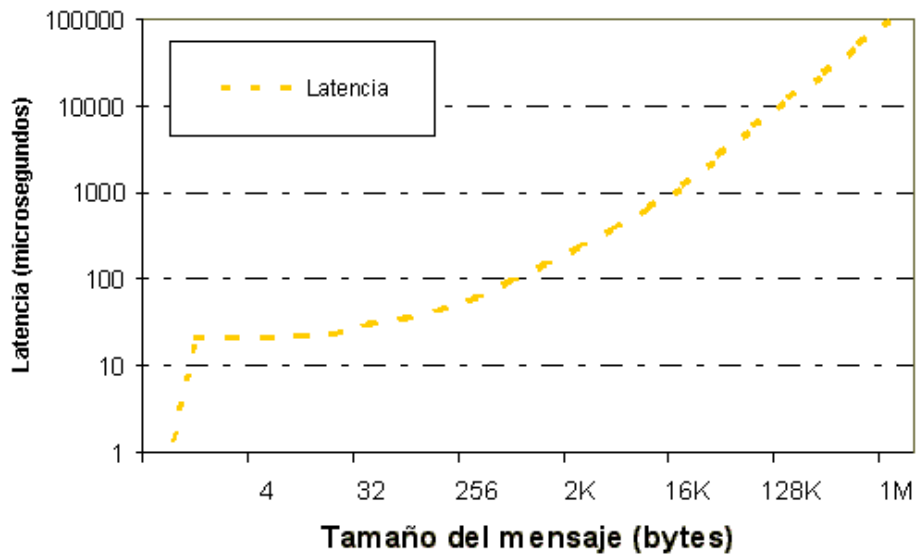
Benchmarks colectivos

O rendemento 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 $L = X/\text{sizeof(float)}$ 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.

AllReduce

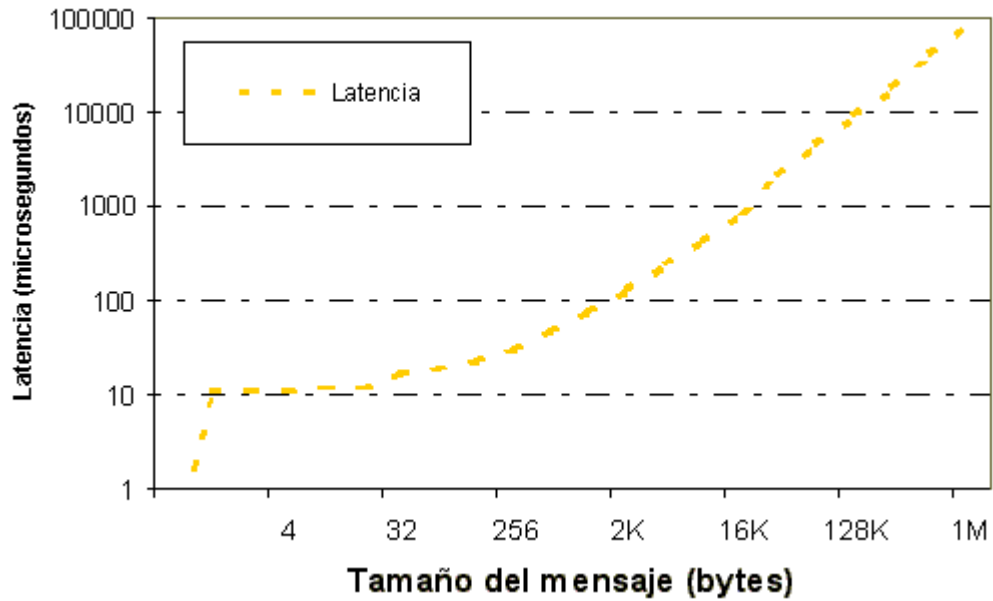


PMB Reduce

Este é o benchmark da función MPI_Reduce. Reduce vectores de números en punto flotante de lonxitude $L = X/\text{sizeof(float)}$ 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.

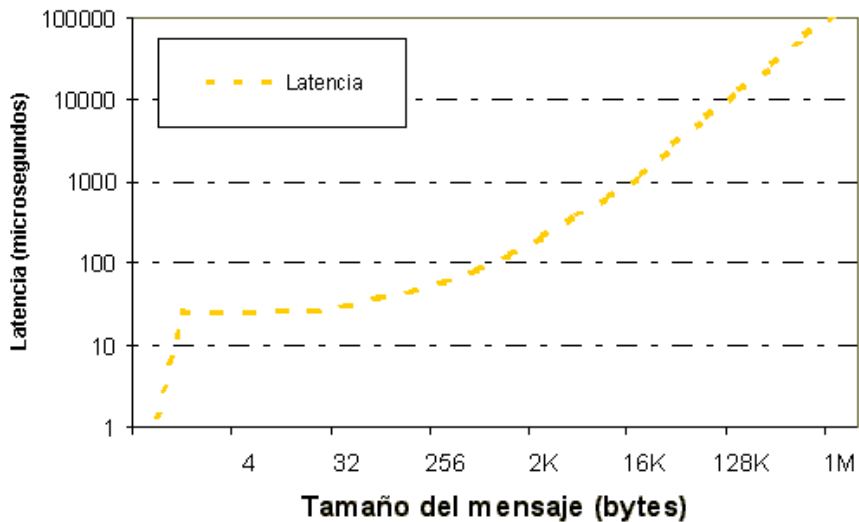
Reduce



PMB Reduce_scatter

Este é o benchmark da función MPI_Reduce_scatter. Reduce vectores de números en punto flotante de lonxitude $L = X/\text{sizeof(float)}$ 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.

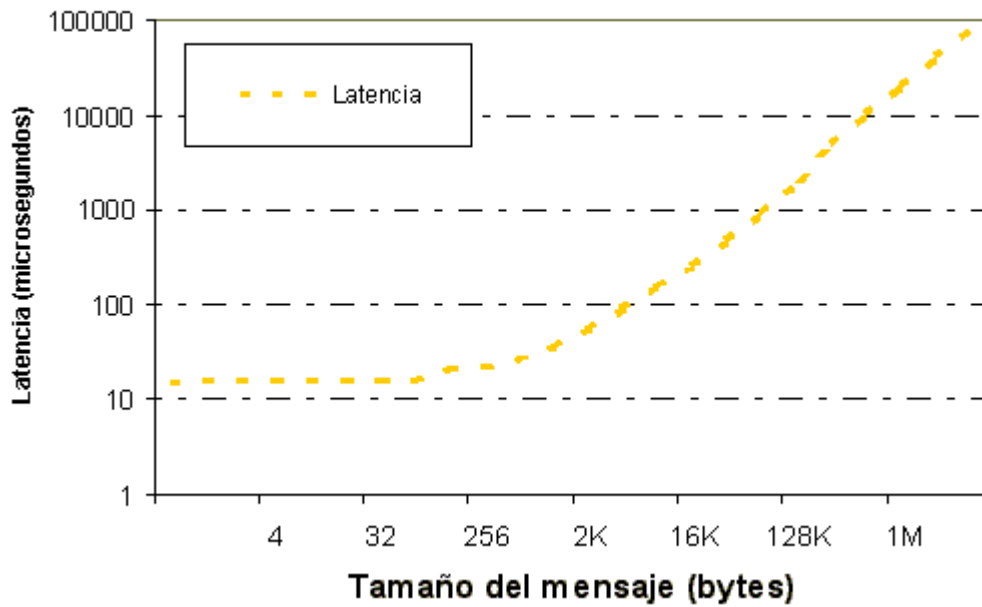
Reduce Scatter



PMB Allgather

Este é o benchmark da función MPI_Allgather. Cada proceso envía X bytes e recibe o grupo dos $X \cdot (n^o_procesos)$ bytes.

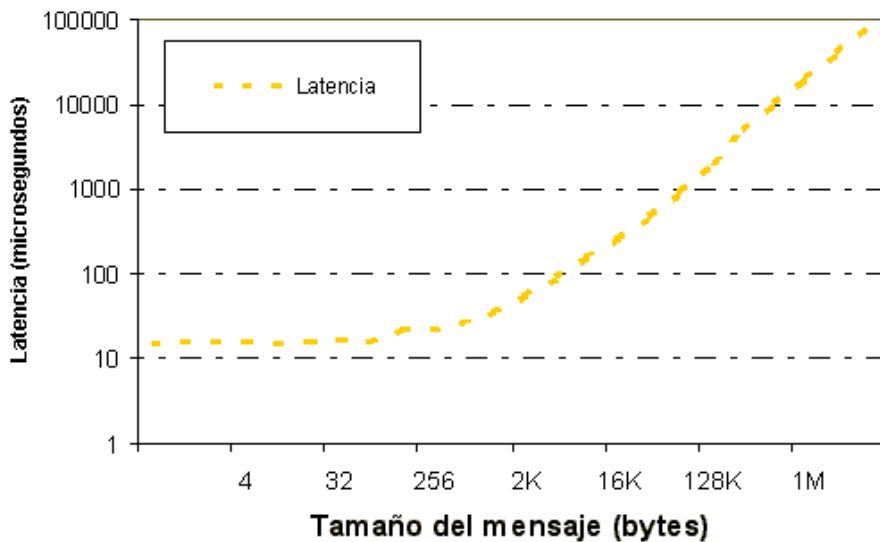
Allgather



PMB Allgatherv

Este é o benchmark da función MPI_Allgatherv. Cada proceso envía X bytes e recibe o grupo dos $X \cdot (n^{\circ}_{\text{procesos}})$ bytes.

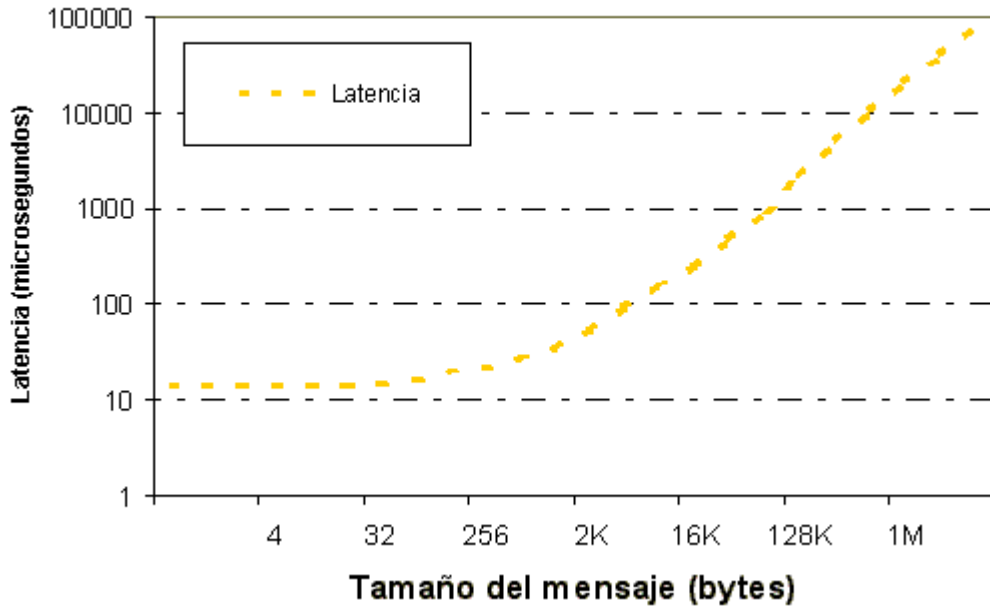
Allgatherv



PMB Alltoall

Este é o benchmark da función MPI_Alltoall. Cada proceso envía e recibe $X \cdot (n^{\circ}_{\text{procesos}})$ bytes (X para cada proceso).

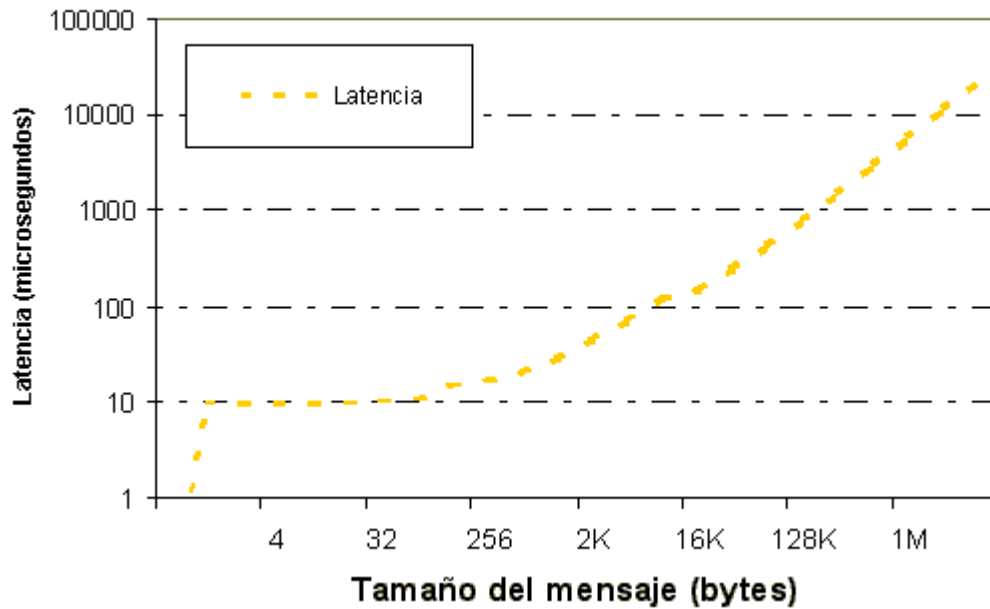
Alltoall



PMB Broadcast

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

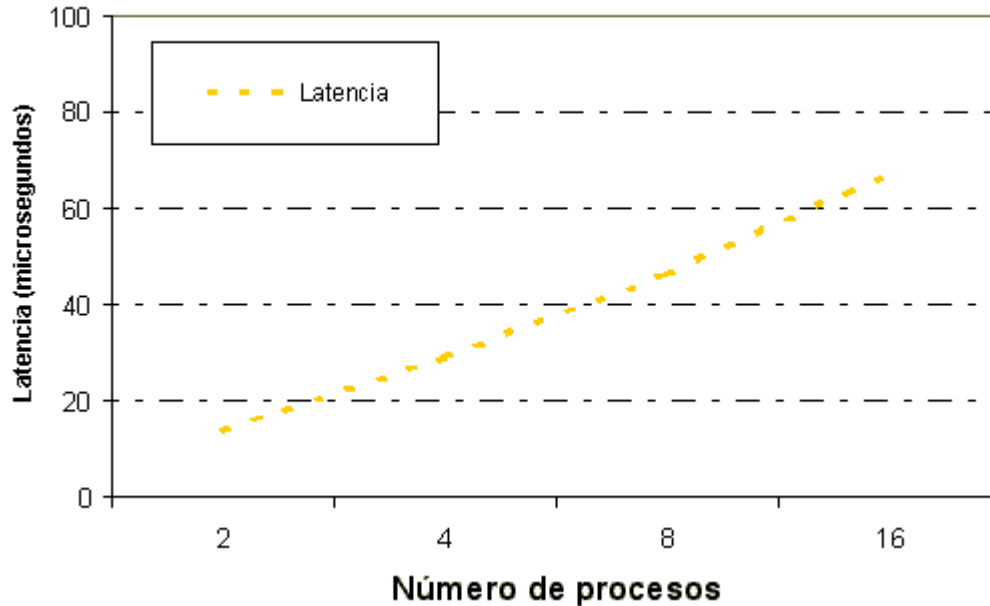
Broadcast



PMB Barrier

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

Barrier



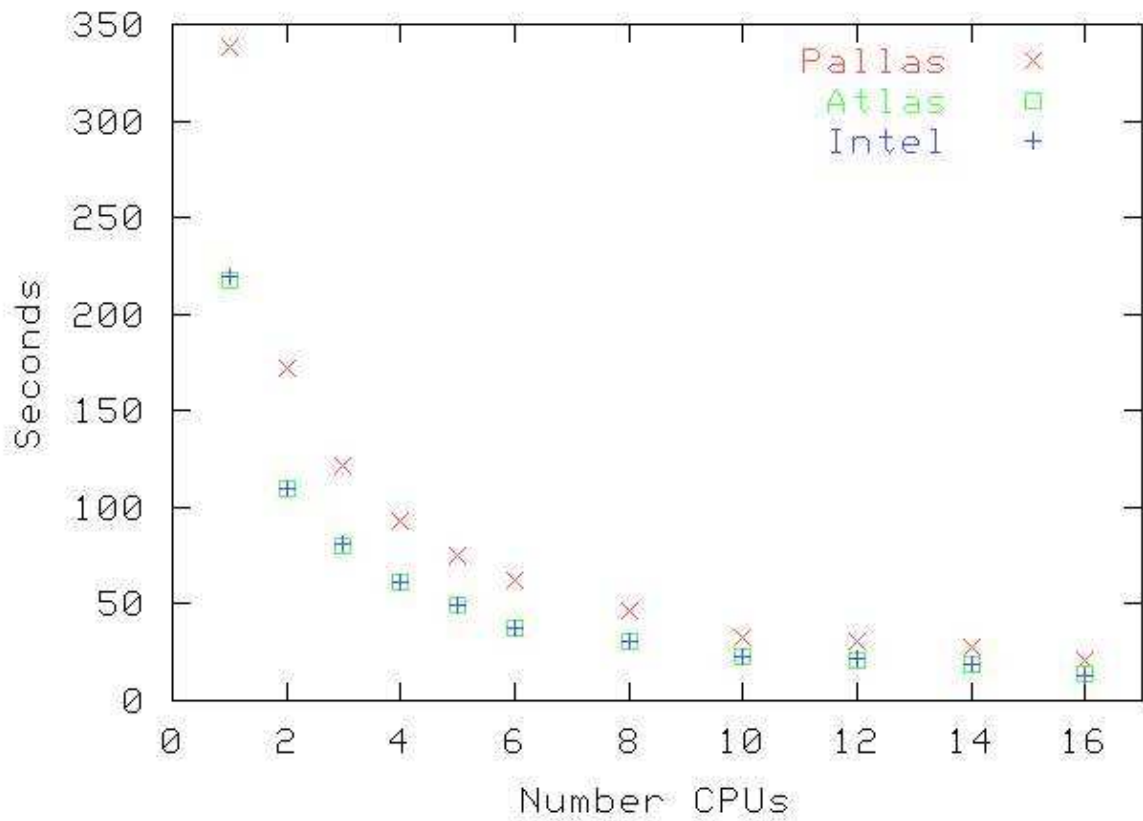
CPMD Si64

Para probalo 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:

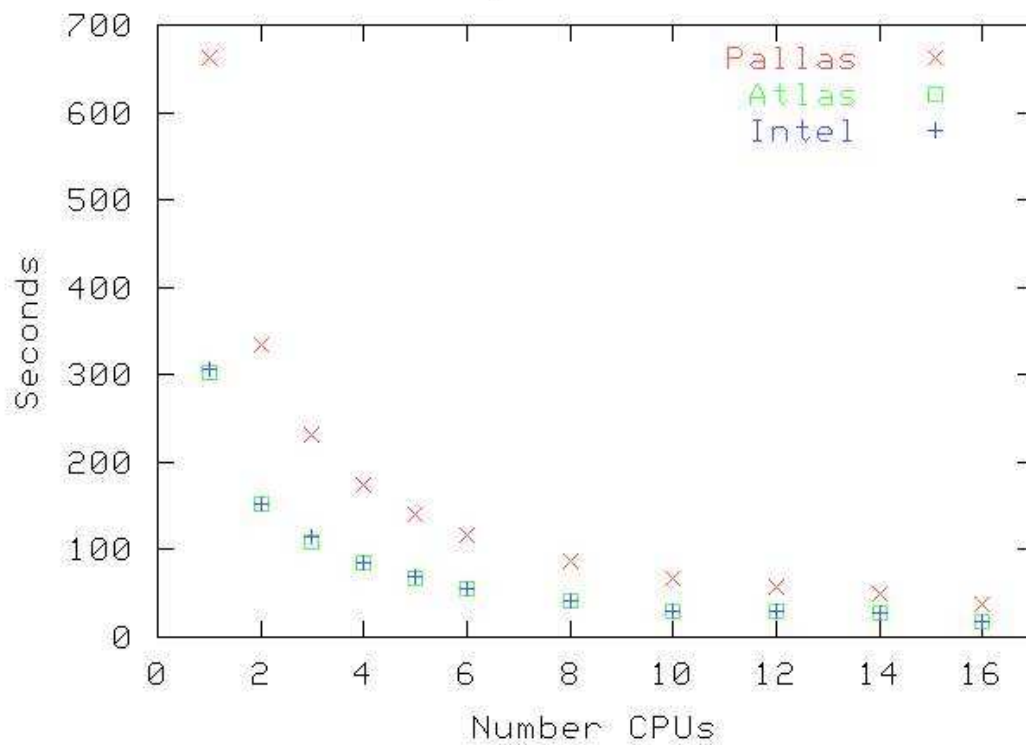
- As propias do compilador de Portlan (marcadas nas figuras como Pallas)
- As do proxecto Atlas
- As propias do fabricante (Intel)

Como entrada utilizouse un conxunto 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 tempo consumido como en SpeedUp en función do número de procesadores utilizados.

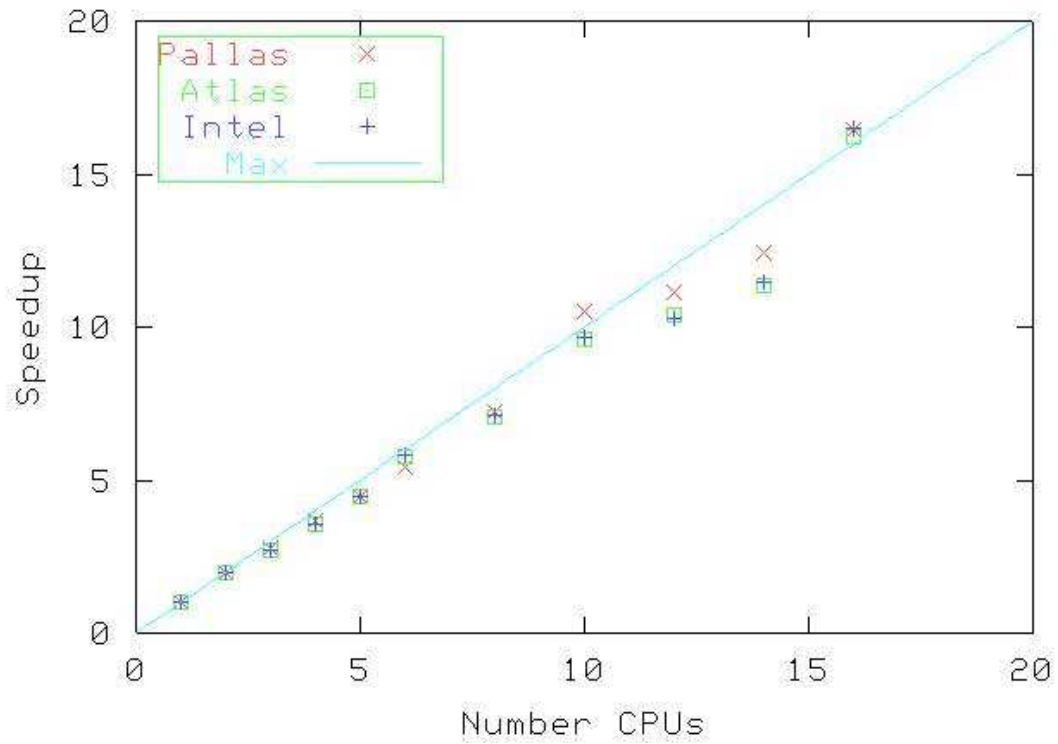
CPMD si64 Elapsed time vs Number CPUs



CPMD si64md Elapsed time vs Number CPUs



CPMD si64 Speedup vs Number CPUs



CPMD si64md Speedup vs Number CPUs

