Análisis Broadcast

En esta ocasión compararemos el rendimiento de la función broadcast con la versión de este código:

```
void M_Broad(float* mensaje,int tam)
{
	int my_rank,size;
	MPI_Comm_rank(MPI_COMM_WORLD,&my_rank);
	MPI_Comm_size(MPI_COMM_WORLD,&size);

	int mask=size-1;
	int d = (int)(log(size)/log(2));
	for(int k=d-1;k>=0;k--)
	{
		mask=mask ^ (int)(pow(2,k));
		if((my_rank & mask) == 0)
		{
			int punto=my_rank ^ (int)(pow(2,k));
			if((my_rank & (int)(pow(2,k))) == 0)
				MPI_Send(mensaje,tam,MPI_FLOAT,punto,0,MPI_COMM_W
				ORLD);
		else
				MPI_Recv(mensaje,tam,MPI_FLOAT,punto,0,MPI_COMM_W
			ORLD,MPI_STATUS_IGNORE);
		}
	}
}
```

Como se puede ver este codigo solo funciona para datos del tipo float y solo funcionara para un cantidad de procesos igual a 2^k asimismo este código siempre envía la información del proceso 0 a los demás.

Ahora el método de trabajo de una función MPI_Bcast es bastante similar a este por lo cual nos esperamos una variación de tiempo no tan grande.
Para la medición del tiempo tomaremos como base al proceso que duró más tiempo de entre todos, para hacer la comparación más justa el tipo que se pondrá en la función original será exactamente igual al de la función creada así como la cantidad de procesos serán los mismos.

Tiempo MPI_Bcast = 0.002256 segundos con 16 procesos | 0.005594 segundos con 32 procesos.

Tiempo Función = 0.002005 segundos con 16 procesos | 0.005935 segundos con 32 procesos.

Análisis: Como podemos ver el tiempo con 16 procesos era bastante parecido pero nuestro programa fue más rápido, esto se debe a que la función principal debe encargar de otros procesos más que solo enviar los datos, por ejemplo castear los datos al tipo deseado, y también la función te permite seleccionar el root de la misma es decir de donde se obtendrá el dato a copiar, estos procesos extra que no tomamos en cuenta retrasan el proceso, en cambio si observamos el tiempo con 32 procesos nuestro proceso fue más lento en casi la misma medida, esto se debe a que el proceso MPI_Bcast no solo dispone de este algoritmo sino se basa en un algoritmo de Árbol binomial, también pudo influir el tratamiento de la CPU por parte del scheduler del Sistema Operativo, es bastante difícil de calcular este tipo de programas ya que hay muchos factores presentes.