Algoritmos Paralelos / Laboratorio

Joseline Maria Zevallos Aliaga
joseline.zevallos@ucsp.edu.pe
Universidad Católica San Pablo

1 Multiplicación de Matriz-vector

Implementando la multiplicación de matriz-vector en MPI primero necesitaremos distribuir la información de x a todos nuestros procesos y luego cada uno debe compartir esta información con todos, una forma de hacer esto es con Gather y luego BCast, sin embargo, existe la opción de hacer todo esto con una sola instrucción,Allgather, la cual usaremos en nuestro programa, está también forma parte de las instrucciones de MPI colectivas. El las siguiente tabla 1 que presenta el libro es una comparación de tiempos de la multiplicación Matriz-Vector.

	Order of Matrix				
comm_sz	*1024*	*2048*	*4096*	*8192*	*16,384*
1	4.1	16.0	64.0	270	1100
2	2.3	8.5	33.0	140	560
4	2.0	5.1	18.0	70	280
8	1.7	3.3	9.8	36	140
16	1.7	2.6	5.9	19	71

Figure 1:

En la Tabla1 se mide el tiempo de ejecución de los programas, el tiempo se disminuye como se ve en la tabla. Como se puede ver las ejecución son con $1, 2, 4, 5, 16$ procesos. Comparando

Comm_sz	1024	2048	4096	8192	16384
1	2.96 e-3	1.24 e-2	5.01 e-2	1.98 e-1	7.92 e-1
2	1.53 e-3	8.86 e-3	2.72 e-2	1.85 e-1	4.33 e-1
4	1.81 e-3	5.71 e-3	2.69 e-2	1.05 e-1	3.98 e-1
8	1.78 e-3	5.33 e-3	2.42 e-2	1.01 e-1	3.80 e-1
16	1.23 e-3	4.79 e-3	2.40 e-2	1.03 e-1	3.41 e-1

Table 1: Comparación de tiempos con diferentes tamaños de la multiplicación Matriz-Vector

los resultados adquiridos se nota que a medida se va aumentando los procesos el tiempo en el que realiza la operación va ir disminuyendo, y cuando se va aumentado el tamaño se de la matriz-vector este también incrementara el tiempo.

2 Odd-Even-Sort

En la búsqueda paralela de Odd-even se comparan por fases, en la fase odd, solamente se hacen swaps en las posiciones pares. El espacio de memoria $a[i]$ debe ser capaz de comunicarse con $a[i-1]$ en la fase j y con $a[i+1]$ en la fase $j+1$, para recolectar y entregar datos respectivamente.

	Number of Keys (in thousands)				
Processes	*200*	*400*	*800*	*1600*	*3200*
1	88	190	390	830	1800
2	43	91	190	410	860
4	22	46	96	200	430
8	12	24	51	110	220
16	7.5	14	29	60	130

Figure 2:

Comm_sz	200	400	800	1600	3200
1	3.69 e-05	4.72 e-05	7.20 e-05	7.27 e-05	9.10 e-04
2	5.91 e-04	4.12 e-04	5.33 e-04	6.31 e-04	7.24 e-04
4	3.64 e-04	3.36 e-04	4.31 e-04	4.96 e-04	5.26 e-04
8	5.45 e-03	5.76 e-03	5.99 e-03	6.62 e-03	6.77 e-03
16	3.23 e-03	4.39 e-03	4.44 e-03	5.03 e-03	5.41 e-03

Table 2: Comparación de tiempos con diferentes tamaños del Odd Even Sort

Los tiempos que da el Odd_Even_Sort a medida se va incrementando los procesos va ir disminuyendo el tiempo ya que cuando se hace las particiones cada proceso va ir ordenando para luego comparar entre procesos e intercambiar si es el caso. El tiempo se va ir incrementando cada ves que el tamaño de la lista incremente.

3 Conclusiones

En conclusión, se puede ver que la manera en que se distribuyen los datos a través de los procesos va a tener un gran impacto en el tiempo de ejecución de nuestros sistemas, por eso se debe, en la medida de los posibles usar las funciones que ofrece MPI y sabemos que son optimizadas(Reduce, Allredude, Scatter, Gather).