Assignment 2, Task 1
Parallelization So2011

Pattreeya Tanisaro

May 16, 2011

Tasks

- Write MPI for numerical integral based on the Trapezoidal Rule using MPI_Send() and MPI_Recv()

Tasks

- Write MPI for numerical integral based on the Trapezoidal Rule using MPI_Send() and MPI_Recv()
- Number of intervals and integration limits must be specified by user

Tasks

- Write MPI for numerical integral based on the Trapezoidal Rule using MPI_Send() and MPI_Recv()
- Number of intervals and integration limits must be specified by user
- Run the program with a multiple number of intervals

Tasks

- Write MPI for numerical integral based on the Trapezoidal Rule using MPI_Send() and MPI_Recv()
- Number of intervals and integration limits must be specified by user
- Run the program with a multiple number of intervals
- Use the program to compute the following integrals $\int_{0.1}^{10000} \frac{1}{x} dx = 11.512925$ and $\int_{10}^{2000} \sin^2 x dx = 995.399112$

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Tasks

- Write MPI for numerical integral based on the Trapezoidal Rule using MPI_Send() and MPI_Recv()
- Number of intervals and integration limits must be specified by user
- Run the program with a multiple number of intervals
- Use the program to compute the following integrals $\int_{0.1}^{10000} \frac{1}{x} dx = 11.512925$ and $\int_{10}^{2000} \sin^2 x dx = 995.399112$
- Measure computation and communication time for sequential and parallel processing and compare the results

Executable

```
$ mpiexec -np 16 -f hosts ./comtrap a 1000000 0.1 10000
```

- `comtrap` takes 4 parameters,

Executable

```
$ mpiexec -np 16 -f hosts ./comtrap a 1000000 0.1 10000
```

- `comtrap` takes 4 parameters,
- The first parameter is to select the function we want to integrate where a for $f(x) = \frac{1}{x}$ and b for $f(x) = \sin^2 x$ etc.,

Executable

```
$ mpiexec -np 16 -f hosts ./comtrap a 1000000 0.1 10000
```

- `comtrap` takes 4 parameters,
- The first parameter is to select the function we want to integrate where a for $f(x) = \frac{1}{x}$ and b for $f(x) = \sin^2 x$ etc.,
- Second parameter is the number of intervals

Executable

```
$ mpiexec -np 16 -f hosts ./comtrap a 1000000 0.1 10000
```

- `comtrap` takes 4 parameters,
- The first parameter is to select the function we want to integrate where a for $f(x) = \frac{1}{x}$ and b for $f(x) = \sin^2 x$ etc.,
- Second parameter is the number of intervals
- Third parameter is the lower limit

Executable

```
$ mpiexec -np 16 -f hosts ./comtrap a 1000000 0.1 10000
```

- `comtrap` takes 4 parameters,
- The first parameter is to select the function we want to integrate where a for $f(x) = \frac{1}{x}$ and b for $f(x) = \sin^2 x$ etc.,
- Second parameter is the number of intervals
- Third parameter is the lower limit
- Fourth parameter is the upper limit

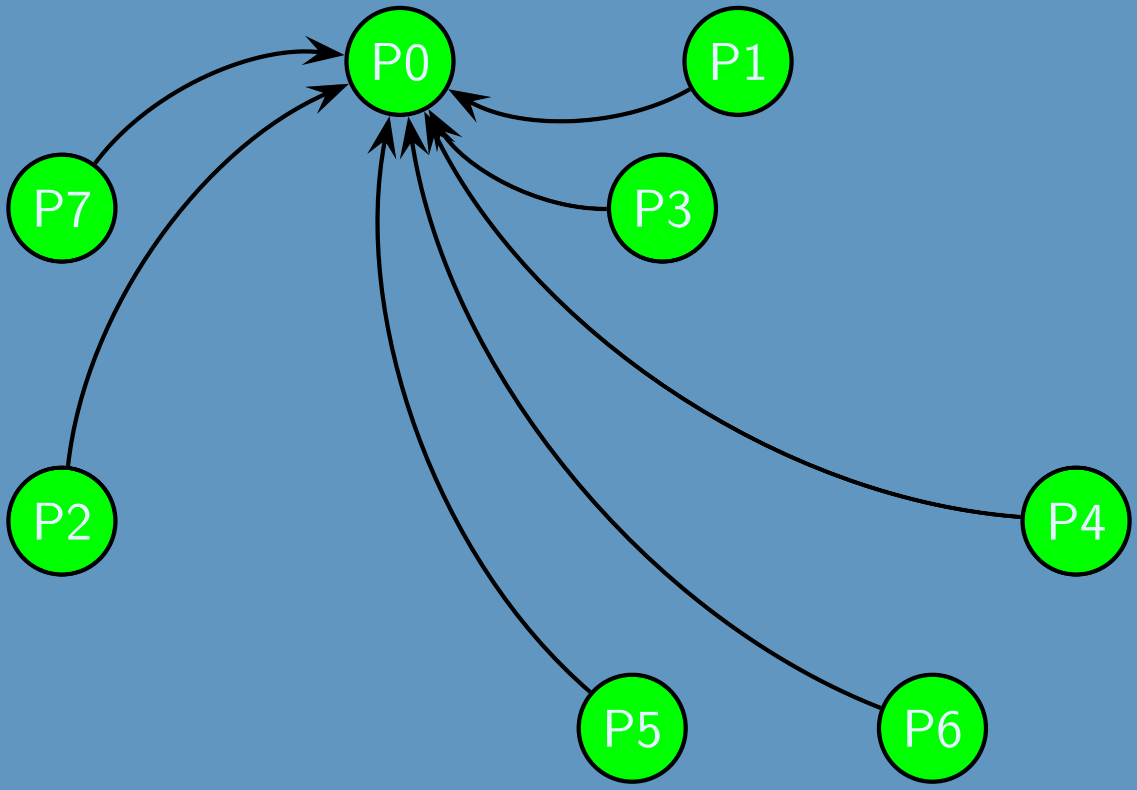
Executable

```
$ mpiexec -np 16 -f hosts ./comtrap a 1000000 0.1 10000
```

- `comtrap` takes 4 parameters,
- The first parameter is to select the function we want to integrate where a for $f(x) = \frac{1}{x}$ and b for $f(x) = \sin^2 x$ etc.,
- Second parameter is the number of intervals
- Third parameter is the lower limit
- Fourth parameter is the upper limit

Communication Topology

- Avoid deadlock, do not send and receive to its own node



Performance Analysis

Table 1:

$f(x) = \frac{1}{x}$ with $N = 1000000, 100000000$

Num of Procs	N=1000000					N=100000000				
	comm t	calc t	total t	speedup	efficiency	comm t	calc t	total t	speedup	efficiency
1	0	39414	39414	1.00	1.00	0	2069204	2069204	1.00	1.00
16	2730	3662	6392	6.17	0.39	839	153504	154340	13.41	0.84
32	3058	4153	7218	5.46	0.17	1674	106746	108425	19.08	0.60
128	6002	2381	8383	4.70	0.04	5978	62048	68026	30.42	0.24
256	11343	2934	14277	2.76	0.01	12870	62242	75112	27.55	0.11

Table 2:

$f(x) = \sin^2 x$ with $N = 1000000, 100000000$

Num of Procs	N=1000000					N=100000000				
	comm t	calc t	total t	speedup	efficiency	comm t	calc t	total t	speedup	efficiency
1	0	96592	96592	1.00	1.00	0	8602115	8602115	1.00	1.00
16	1298	11801	13099	7.37	0.46	932	569693	570625	15.07	0.94
32	1818	7764	9582	10.08	0.32	1763	319845	321609	26.75	0.84
128	5655	7793	13448	7.18	0.06	6042	152782	152782	56.30	0.44
256	12081	7821	19902	4.85	0.02	12837	203686	216523	39.73	0.16

Conclusion

- Parallel processing yields a better overall performance than sequential processing

Conclusion

- Parallel processing yields a better overall performance than sequential processing
- Speedup is obviously increased as the number of processors increase but at one point although increasing number of processors, the speedup starts to drop. Since parallel overheads (communication operations and redundant computation) start to dominate the execution time. Therefore, optimum number of processors which minimizes overall execution time should be made for an optimistic assumption

Conclusion

- Parallel processing yields a better overall performance than sequential processing
- Speedup is obviously increased as the number of processors increase but at one point although increasing number of processors, the speedup starts to drop. Since parallel overheads (communication operations and redundant computation) start to dominate the execution time. Therefore, optimum number of processors which minimizes overall execution time should be made for an optimistic assumption
- The more complex the algorithm, the higher the speedup

Conclusion

- Parallel processing yields a better overall performance than sequential processing
- Speedup is obviously increased as the number of processors increase but at one point although increasing number of processors, the speedup starts to drop. Since parallel overheads (communication operations and redundant computation) start to dominate the execution time. Therefore, optimum number of processors which minimizes overall execution time should be made for an optimistic assumption
- The more complex the algorithm, the higher the speedup
- The more complex the algorithm, the less the efficiency. As our result shows, when number of processors are 128 and 256, the efficiency can be less than 5%