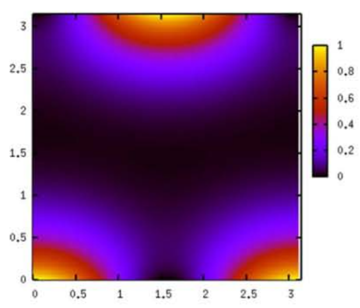
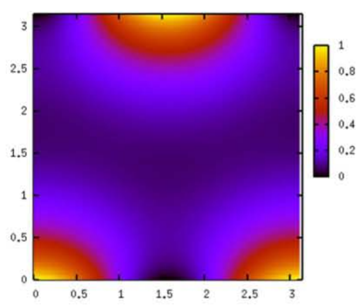
APC524_HW4

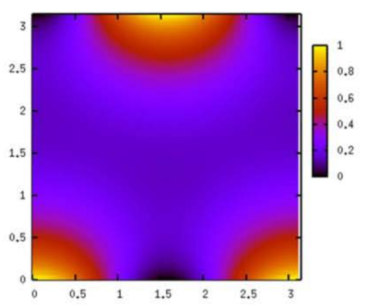
Jin Cheng

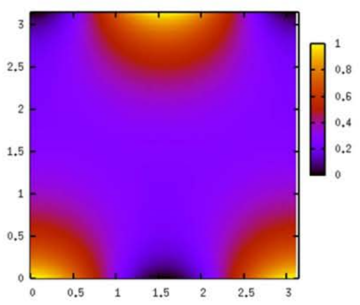
1. Evolution of temperature  (contour plot of temperature at different time points)

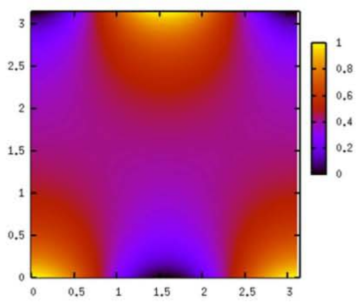
(t0 = 1.0*pi^2)

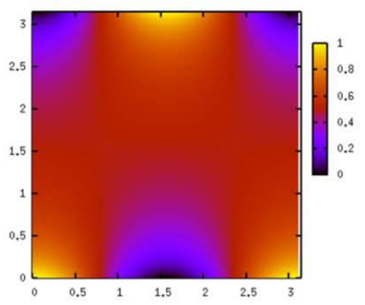

T = 0.02*t0


T = 0.04*t0


T = 0.06*t0


T = 0.10*t0


T = 0.20*t0

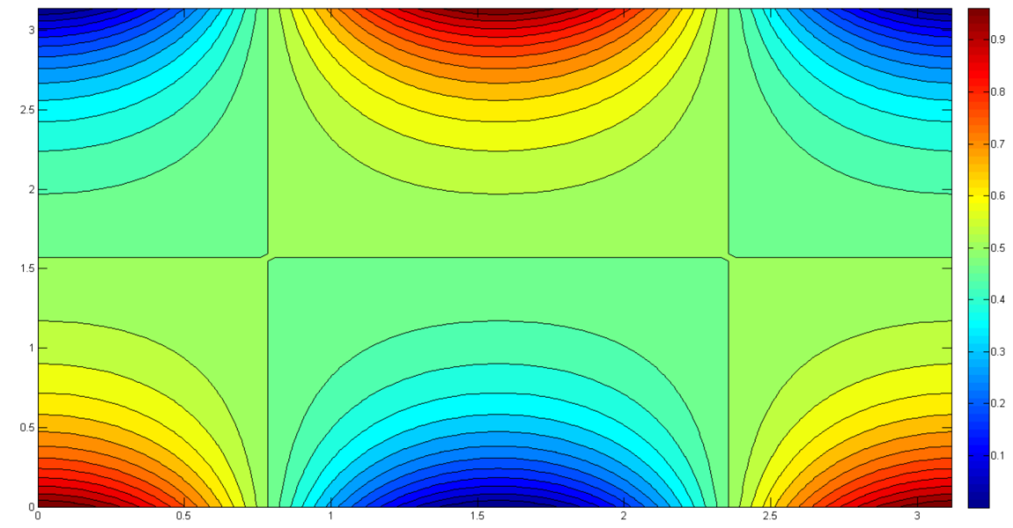

T = 0.50*t0

2. Final temperature result:

Serial and mpi implementations result in the same temperature contour plot and volume averaged temperature, reported as follows:

Final temperature contour plot (at T=10*t0) from serial implementation

volume averaged temperature = 0.500



3. Speed up comparison

**Local calculation tests** for estimating the walltime:

| Grid = 128<br>t0 = 1.0*pi^2 | OMP (s) | estimated wall time for OMP(h) | MPI (s) | estimated wall time for MPI (h) |
|---|---|---|---|---|
| Nproc = 1 | 4.1 | 50 | 4.7 | 57 |
| Nproc = 2 | 2.9 | 35 | 2.6 | 32 |
| Nproc = 4 | 2.5 | 30 | 1.8 | 22 |
| Nproc = 8 | 1.8 | 22 | 1.4 | 17 |
| Nproc = 16 | - | - | 1.6 | 19 |

**CPU times**

Serial

|  | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|
| 1 | 31.73 | 518.08 | 8624.47 | - |

As shown in the table above, computational cost is O($N^4$).

OMP

| | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|
| 1 | 37.40 | 774.09 | 11926.9 | - |
| 2 | 19.62 | 293.69 | 4186.66 | 125216 |
| 4 | 13.07 | 156.0 | 2178.3 | 70932.0 |
| 8 | 9.68 | 93.93 | 1393.3 | 43594.0 |

As shown in the table above, the computational cost with one processor is similar to that in serial case. When increasing the number of processors used, computational cost reduced significantly. Number of processors increases from 1 to 2, the CPU time is reduced by a factor of 0.5 or even less. But the number of processor increases from 2 to 4 or 8, the CPU time is only reduced by a factor generally larger than 0.5.

MPI (I am sorry that I have not finished all the calculation, so some of the data is not available.)

| | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|
| 1 | 32.99 | 539.21 | 8721.43 | - |
| 2 | 17.70 | 291.63 | - | - |
| 4 | 10.99 | 154.66 | - | - |
| 8 | 7.18 | 86.04 | - | - |
| 16 | 7.03 | 61.10 | - | - |

As shown in the table above, the speeding up is generally similar to the OMP case. One thing should be noted that when the number of processors increases from 8 to 16, the CPU time is not reduced significantly. At the end of each iteration two “MPI_SEND” operations and two “MPI_RECV” operations are applied to communicate with two adjacent domains. After integrating to final time, processor0 collects final temperature data from all the rest processors to evaluate the volume averaged temperature.

## 4. Conclusion

OMP:

For programmer, the implementation is straightforward. One does not need to know the details about how the processors arrange the data since shared memory is used.

As far as I understand, OMP implementation is composed of a series of fork-joint combination which is considered to be less efficient than a giant fork-joint structure.

When more CPU is used, the I/O process would be the bottleneck and severely affects the performance.

MPI:

For programmer,  the implementation is more complicated. But one is clear about how each processor deals with the data. Programmers are required to set up how processors communicate with each other and how to divede the whole job into parallel pieces, which may affects the performance in general.

Since distributed memory is used, the implementation is equivalent to several programs running on each processor individually. They only communicate with each other when necessary. Thus the parallel program has a giant fork-joint structure.