Feedback Guided Scheduling for 2D Loops

Oleh Olkhovskyy

EPCC, University of Edinburgh
University of Edinburgh
email oleg@epcc.ed.ac.uk

Introduction

The aim of this work is investigation of 2-dimensional (2D) imbalanced loops scheduling in a shared-memory machine. Loop scheduling is assignment of loop iterations to threads/processors to minimise overheads. These overheads are synchronisation, process management, communication and load imbalance. Synchronisation occurs when processor must wait for some action by another processor, such as relinquishing a critical region. Process management refers to time needed to calculate iterations boundaries of each processor. Communication is interaction between processors. Load imbalance occurs when some processors finish their calculations earlier than other processors.

Loop scheduling algorithms

- Guided algorithm[2]
- Affinity[3]
- Trapezoid[4]
- FGLS 1D: This 1D version of FGLS 2D, which will be described later[1].
- Feedback Guided Loop Scheduling(FGLS)[1]:
  The loops iterations are divided into P patches, and are distributed between processors. Each processor executes its iterations, keeping a track on time of executing of the whole patch. Then dividing this time by the number of iterations in patches we get the mean load per iteration. Next, we find new boundaries, based on equipartitioning of the area under the mean load per iteration. New boundaries are formed as follows: first we divide the area in the x-direction, so that agregate time in the left and right parts is in the ratio $\left[\frac{P}{2}\right]$. Then with each of new patches we repeat the procedure, cutting each patch in the direction perpendicular to the previous one, until we have P patches. A second way is similar, except that we choose cut direction in such a way that new patches are as square as possible.
- Guided 2D:
  This algorithm is based on the guided algorithm. The size of a patch is $\frac{R}{2P}$. Dividing the area into rectangles of exponentially decreasing size is quite a difficult task. Here we use an algorithm, that allows us to get almost square patches, with almost exponentially decreasing size. The first patch $\left(sized\frac{R}{2P}\right)$ is square. The next few patches are of the same width, but decreasing heights, and are under the first one. The remained of the height, is either distributed between these patches, or forms a new patch, if it is close to the patch size. This operation is repeated for the rest of the area but in alternating direction.
  Figure 1 shows an example of dividing an area of the size 200x200 for four processors. Darker color show patches build in x-direction, lighter color shows patches build in y-directon.

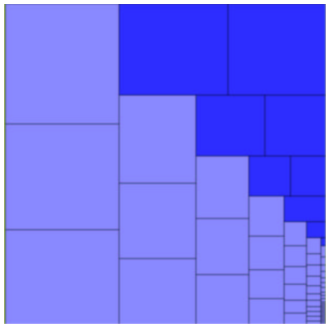

Figure 1 : Guided 2D

Workload

The workload consists of communication and imbalance. The communication part calculates mean value of array element, based on its neighbours. There are several imbalance loads including Gaussian,Pond,Ridge and Sinusoidal standing wave.

Benchmark Results

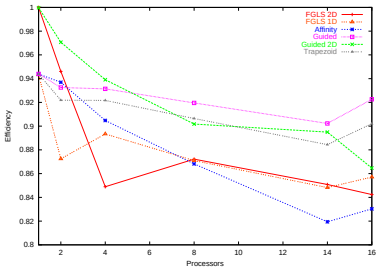

Figure 2 : Gaussian Load.Communication>>Imbalance

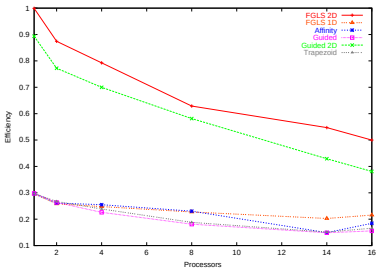

Figure 3 : Gaussian Load.Communication≈Imbalance

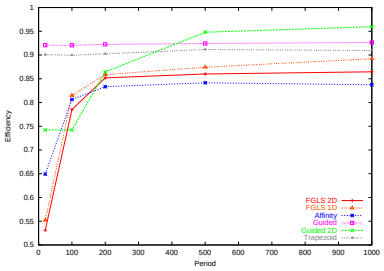

Figure 4 : Varying period.Communication=0

Conclusions

Both algorithms FGLS 1D and FGLS 2D require slowly changing imbalance;cumulating this can be clearly seen from Figure 4. In clear imbalance case 1D algorithms are more efficient, since they distribute iterations by lines, not by rectangles,as 2D algorithms does. Hence each patch has points with high load and low load, thus, the correspondent imbalance is spread more than in the case of 2D algorithm.
In situation when time of imbalance is approximately the same as communication time, 2D algorithms are ahead of 1D algorithms, due to fact that they use patches close to squares. Since FGLS 2D produces only P patches, it is more efficient then Guided 2D in communication load.

Acknowledgements

I would like to thank my supervisor Mark Bull for helping in this project, and all EPCC staff for giving value information and sharing their experience.

References

1. Bull M.,Feedback Guided Dynamic Loop Scheduling: Algorithms and Experiments,in Proceedings of EuroPar '98, Lecture Notes in Computer Science vol.1470,Berlin,1998.
2. Polychronopoulos C.D., Kuck D.J.,Guided self-scheduling: A practical scheduling scheme for parallel supercomputers,IEEE Transactions on Computers, Vol. 36,No 12,1987.
3. Subramaniam S., Eager D.L.,Affinity Scheduling of Unbalanced Workloads,University of Saskatchewan Saskatoon.
4. Tzen T.H., Ni L.M.,Dynamic loop scheduling dor shared-memory multiprocessors,In Proceedings 1991 International Conference on Parallel Processing.