Adaptive OpenMP for Large NUMA Nodes

Aurèle Mahéo[1], Souad Koliaï[1], Patrick Carribault[2,1], Marc Pérache[2,1], and William Jalby[1]

[1] Exascale Computing Research Center, Versailles, France
[2] CEA, DAM, DIF, F-91297, Arpajon, France

1 Introduction

The advent of multicore processors advocates for a hybrid programming model like MPI+OpenMP. Therefore, OpenMP runtimes require solid performance from a small number of threads (one MPI task per socket, OpenMP inside each socket) to a large number of threads (one MPI task per node, OpenMP inside each node). To tackle this issue, we propose a mechanism to improve performance of thread synchronization with a large spectrum of threads. It relies on a hierarchical tree traversed in a different manner according to the number of threads inside the parallel region. Our approach exposes high performance for thread activation (parallel construct) and thread synchronization (barrier construct). Several papers study hierarchical structures to launch and synchronize OpenMP threads [1, 2]. They tested tree-based approaches to distribute and synchronize threads, but they do not explore mixed hierarchical solutions.

2 Adaptive OpenMP Runtime

Multiple tree shapes exist to perform thread synchronization. The most straightforward solution is a *flat tree* with one root and one thread per leaf. It allows fast synchronization for a few number of threads: the master thread iterates through leaves to flip one memory cell. But with an increasing number of threads, performance drops. A tree mapping the topology of the underlying architecture is more suitable. Such tree exposes more parallelism for synchronizing a large number of threads, but invoking few threads requires a high overhead for tree traversal.

Our approach is to bypass some parts of the tree when the number of threads is small enough to impact only a sub-tree of the topology tree. Figure 1 depicts this mechanism on a 32-core node (4 processors with 8 cores). Thus, when the number of threads is lower than 8, the master thread starts at the second level (leftmost child of the root). For a larger number of threads, the topology tree is still fully used.

3 Experimental Results

We implemented our mechanism in MPC [3] inside the existing OpenMP runtime. We conducted experiments on a Bull bullx 6010 server embedding a

B.M. Chapman et al. (Eds.): IWOMP 2012, LNCS 7312, pp. 254–257, 2012.

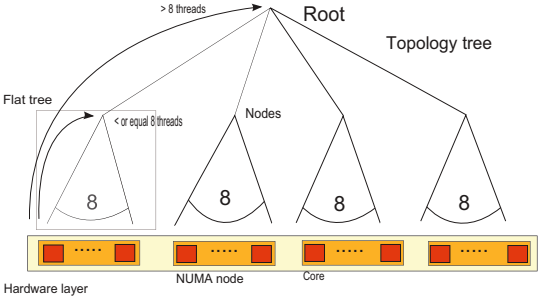
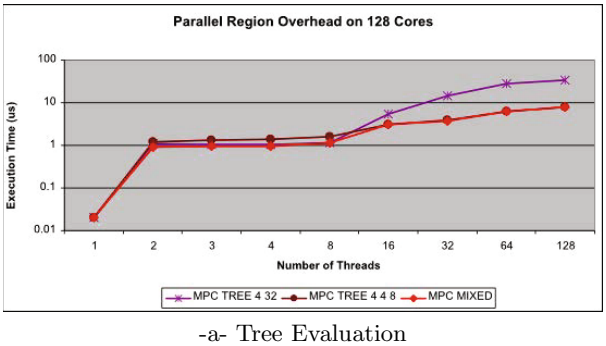
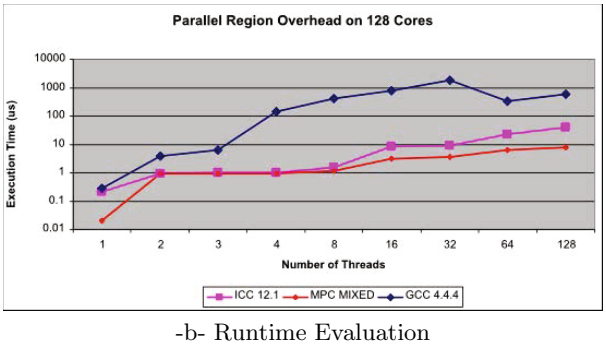


Fig. 1. Tree Structures for 32-Core Node



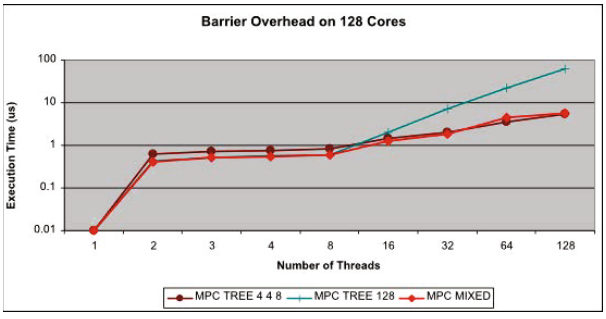
-a- Tree Evaluation



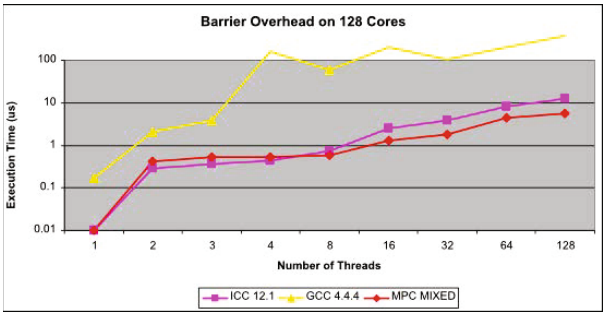
-b- Runtime Evaluation

Fig. 2. Parallel Overhead Evaluation on 128 Cores

memory controller for cache coherency called Bull Coherency Switch (BCS). This ccNUMA system allows configuration with up to 16 processors sockets (4 modules containing 4 processor sockets and a BCS) sharing a single coherent memory space. Thus these 16 processor sockets provide 128 CPU cores.



-a- Tree Evaluation



-b- Runtime Evaluation

Fig. 3. Barrier Overhead Evaluation on 128 Cores

Figures 2 and 3 present the results of EPCC benchmarks [4] up to 128 threads. Figure 2-a depicts the overhead of entering and exiting a parallel region for different trees: 2-level tree (MPC 4-32), topology tree (MPC 4-4-8) and our approach (MPC MIXED). We benefit from both trees by using this bypass mechanism: the overhead is equal to the minimum of trees. Furthermore, Figure 2-b illustrates that our approach achieves better performance than state-of-the art implementations. Figure 3-a and 3-b illustrate the same experiments for the OpenMP barrier construct.

4 Conclusion and Future Work

We introduced a new mechanism to increase the performance of OpenMP thread activation and synchronization for a wide spectrum of threads. It shows significant performance improvement on a 128-core node on EPCC microbenchmarks. For future work, we have to investigate more this strategy by extracting a generic algorithm to bypass trees in a flexible way. Finally, it would be interesting to integrate OpenMP tasks and check the influence of task scheduling.

References

1. Nanjegowda, R., Hernandez, O., Chapman, B., Jin, H.H.: Scalability Evaluation of Barrier Algorithms for OpenMP. In: Müller, M.S., de Supinski, B.R., Chapman, B.M. (eds.) IWOMP 2009. LNCS, vol. 5568, pp. 42–52. Springer, Heidelberg (2009)
2. Broquedis, F., Furmento, N., Goglin, B., Wacrenier, P.A., Namyst, R.: ForestGOMP: an efficient OpenMP environment for NUMA architectures. International Journal on Parallel Programming 38(5), 418–439 (2010)
3. Carribault, P., Pérache, M., Jourdren, H.: Enabling Low-Overhead Hybrid MPI/OpenMP Parallelism with MPC. In: Sato, M., Hanawa, T., Müller, M.S., Chapman, B.M., de Supinski, B.R. (eds.) IWOMP 2010. LNCS, vol. 6132, pp. 1–14. Springer, Heidelberg (2010)
4. Bull, J.M., O'Neill, D.: A Microbenchmark Suite for OpenMP 2.0. SIGARCH Comput. Archit. News 29(5), 41–48 (2001)