Scalable Parellel Octree Using HPX With Hilbert Curve

Zahra Khatami [1,2], Hartmut Kaiser [1,2] and J. Ramanujam [1]
[1]Center for Computation and Technology, Louisiana State University
[2]STE||ER Group

STE||AR GROUP

LSU Center for Computation & Technology
Interdisciplinary | Innovative | Inventive

ABSTRACT

Octrees are used widely as a data structure in many applications such as N-Body, 3D computer graphics, molecular dynamics simulations and etc. They are used for recursively subdividing a three dimensional space into eight sub-octrees.

Improving application scalability is one of the challenges in parallelization. Parallelizing an octree using conventional techniques inhibits the desired scalability due to the global synchronization and the communication overheads between processors. HPX provides an efficient scalable parallelism by minimizing the resources starvation, latencies, overheads and the delays due to accessing the shared resources.

In this research, we present the new adaptive scalable parallel octree used as a data structure in N-Body application using HPX runtime system. The performance of the HPX for the octree construction has been improved by sorting data in a space using Hilbert curve, which is used for finding the nearest neighbors in an octree.

The scalability test and the execution time for up to 10,000,000 inputs is done on 16 cores with HPX and OpenMP. The comparison result shows 15.8x speedup for HPX with Hilbert and 11x speedup for OpenMP. Hilbert curve has improved HPX performance scalability and reduces the execution time.

Introduction

HPX is a parallel runtime system which enables fine-grained parallelism to achieve a better load balancing. HPX exposes parallelism using the *future* construct allows users to avoid the global barrier synchronization imposed in many parallization paradigms. [1].

In order to achieve a better load balancing on the parallel processors in HPX, Hilbert Curve is used in this research. Hilbert Curve method is one of the space-filling curves for finding the nearest neighbors in a spatial data structure like an octree.

The octree divides all particles into 8 sub-octrees based on a predetermined threshold. As a result, it will keep all neighbor particles in one sub-octree. Also, it results in having an equal number of particles in all the sub-octrees of each octree, which results in having better load balances on the threads.

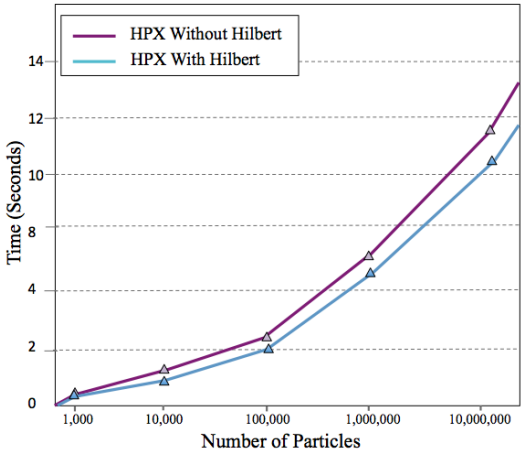

Figure 1: HPX with Hilbert Curve

HPX vs OpenMP

The *future* based parallelization provides rich semantics for exploiting parallelism available within each applications and thereby increase the scaling. Its goal is to let every computation to be proceed as far as possible. *hpx::async* is used for creating the parallel octree recursively, which is based on *future*.

In Hilbert curve method, Hilbert distances is computed for each inputs based on their position in a space. Merge sort is the comparison-based sorting algorithm, which is used here for sorting nodes based on their Hilbert distances. As a result, the three dimensional spatial representation is mapped to a linear array. The octree is constructed after sorting all the nodes based on their neighbors Hilbert distances. HPX performance for an octree using Hilbert curve method is shown in Fig.1.

For evaluating HPX performance, we compared the strong scaling for the Octree construction with HPX and OpenMP.

Dynamic scheduling is used when there is varying amount of tasks on each threads. So for OpenMP parallel code, *"#pragma omp parallel for schedule (dynamic)"* is used. It is difficult to have the desired scalable parallel applications because of having forked-joint model in OpenMP [2].

The strong scaling is shown in Fig.2. It can be seen that HPX octree application has comparable scaling respected to OpenMP [2].

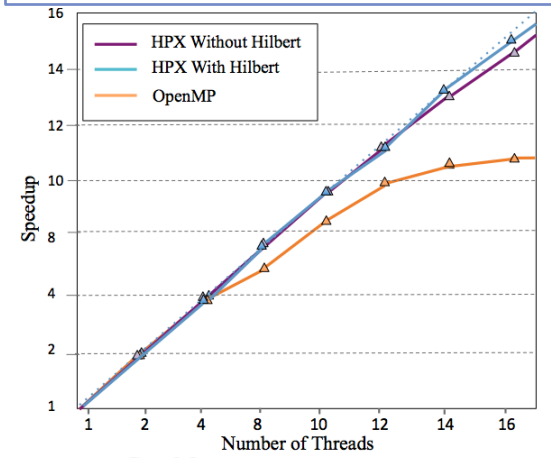

Figure2: Speedup HPX versus OpenMP used in paper [2]

CONCLUSION

By considering the results, it can be concluded that making input data to be as a sorted linear array with Hilbert curve method makes threads having a better load balance and as a result reduces the execution time. Also, it is illustrated that HPX has the ability to remove global barrier synchronization and helps having more scalable parallel application compared with OpenMP.

References

[1] H. Kaiser, T. Heller, B. Adelstein-Lelbach, A. Serio, and D. Fey, "Hpx – a task based programming model in a global address space," *PGAS 2014: The 8th International Conference on Partitioned Global Address Space Programming Models*, 2014.

[2] Zahra Khatami, Hartmut Kaiser, Patricia Grubel, Bryce Adelstein-Lelbach, Adrian Serio and J. Ramanujam, "A massively parallel distributed N-Body simulation code implemented with HPX", 28th ACM Symposium on Parallelism in Algorithms and Architectures, 2016.

This works was supported by NSF awards 1447831.