Exploiting Simultaneous Multi-threading for Parallel Mesh Generation on Intel HT Processors

C. Antonopoulos, N. Chrisochoides, D. Nikolopoulos

Department of Computer Science,

College of William & Mary

1 Motivation

• Quality mesh generation is a computationally intensive application. Parallel mesh generation (PCDT) [1] is a realistic solution for real-world problems, which are modeled with millions or billions of elements.

• Best sequential mesh generator : Triangle [2]

• However:

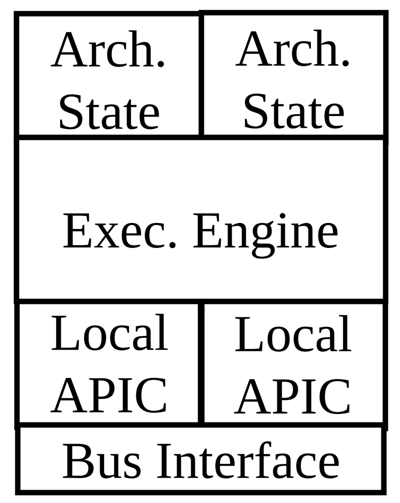
  • We do not have the time and resources to optimize single node parallel mesh generation so that it is comparable to the best sequential algorithm.

  • SMT processors offer a cost-effective alternative:

    • They often come for free (i.e. Intel HT)

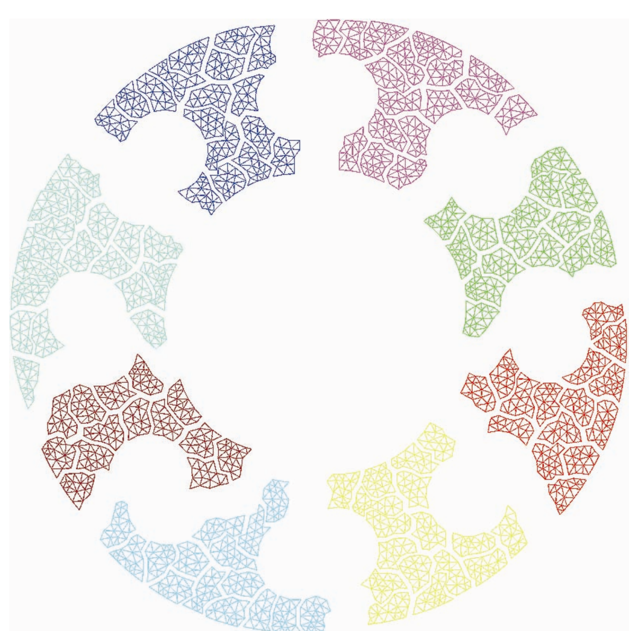
    • Taking advantage of the additional execution contexts may reduce the execution time of the single-node parallel version to make it comparable with the extensively optimized sequential code.

  • Avoid putting too much additional overhead to the programmer.



Intel HT Architecture

2 Multiple Levels of Parallelism



Coarse Grain

• Subdomain level.

• MPI, message-passing implementation across different nodes.

• Superlinear speedup for few processors, almost linear speedup up to 128 processors.

• Use of processes (kernel-level threads provided by the operating system).

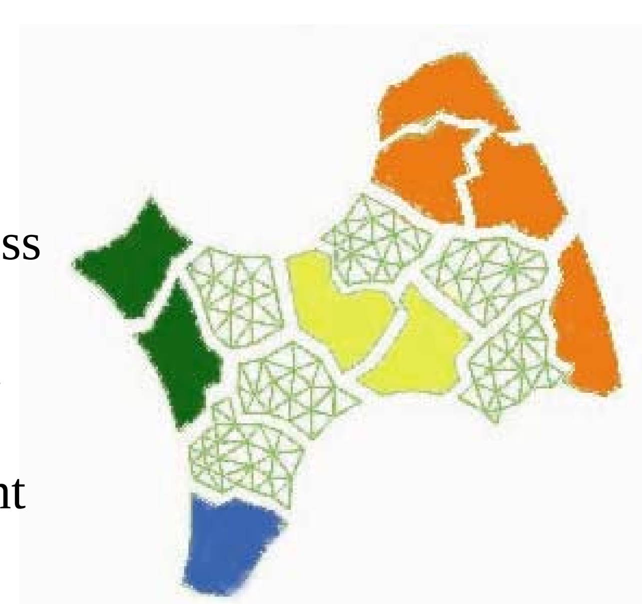
Medium Grain

• Cavity level

• Shared-memory implementation, across the processors of an SMP.

• Requires algorithmic modifications in order to ensure that the computation in one cavity does not affect the concurrent computation of another cavity.

• Future work . . .



Fine Grain

• Element (triangle) level.

• Shared memory implementation, across the multiple execution contexts present in a HyperThreaded processor.

• Too fine-grained parallelism.

• Profiling Data (for a typical problem size) :

  • Element-level code is accountable for **58%** of the total execution time.

  • However:

    • **22 10^6** invocations of the parallelizable function.

    • Only **4-6 usec** computation per invocation.

    • Only **5-6 elements** in average per invocation.

  • Cost of thread triggering is **1 usec** on the specific architecture …

  • Maximum expected speedup is **1.33** to **1.5** without taking into account any additional overheads …

3 Implementation Approaches

Overhead Minimization

• Minimization of the interaction between threads.

  • Use of decentralized application data structures.

  • Use of local, per-thread work queues.

• Use of non-blocking, wait-free synchronization algorithms wherever possible.

• Interesting tradeoff between synchronization and extra work:

  • It is possible to significantly reduce synchronization between threads by bearing the possibility of unnecessary repetition of the computation for the same element by two different threads.

Use of a Threading Substrate (Factory)

• Factory:

  • Facilitates the exploitation of parallelism in C++ programs. Flexible multi-translation of one abstraction of application-level parallelism.

  • User-level threads substrate. Multilevel task parallelism.

  • Strongly typed C++ API.

  • Computation is expressed as "WorkUnits". 1 WorkUnit / element.

  • Custom slab allocator implementation for effective memory management.

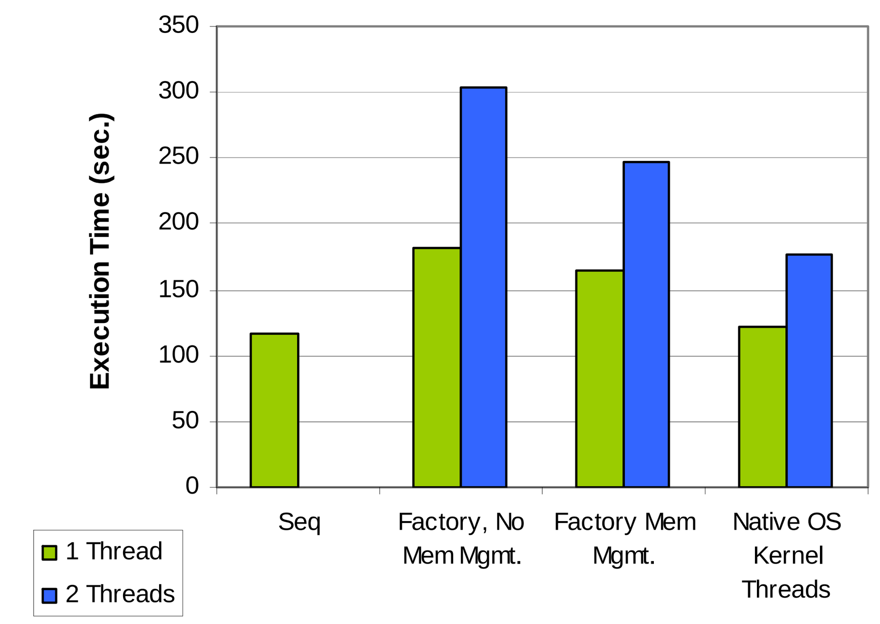
• Overhead for WorkUnit creation, dispatching, destruction : **1 usec.**

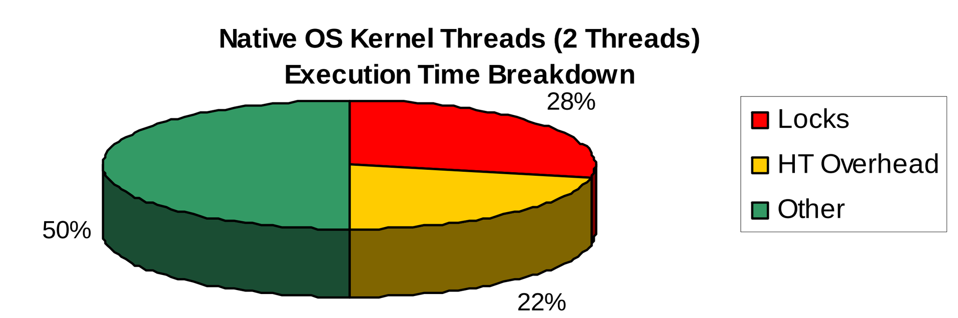
Kernel Threads as Execution Vehicles

• Native, OS kernel threads are used as execution vehicles throughout application life.

• Kernel threads create/enqueue or dequeue/execute work.

4 Prelim. Experimental Evaluation





5 Future Work

• Use simulation to find-out whether the fine-grained parallelism of PCDT is exploitable by architectures that offer H/W support for multithreading

• Target other existing and emerging SMT architectures (IBM Power5, IBM BlueGene, future Intel & AMD multicore processors).

• 3D parallel mesh generation:

  • Coarser-grained parallelism.

  • Higher computational intensiveness.

  • Significant importance real-world application.

6 Acknowledgements

[**NSF Grants**] ITR/ACI-0312980, CARRER/CCF-0346867

We would like to thank Andrey Chernikov and Andriy Fedorov for their valuable comments and help. We would also like to thank Scott Schneider for providing the "Factory" threading substrate.

7 References

[1] **Parallel Delaunay mesh generation kernel**. N. Chrisochoides and D. Nave. Int. J. Numer. Meth. Engng., 58:161–176, 2003.

[2] **Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator.** J.R. Shewchuk. Proc. 1st Workshop on Applied Computational Geometry, 1996.