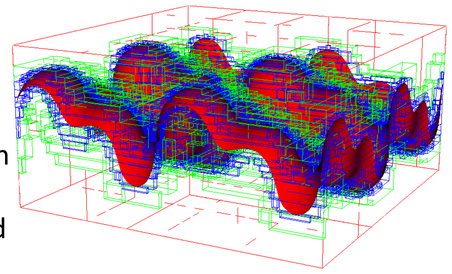
Parallelization Improvements to BoxLib Applications with Tiling and OpenMP

Jessica Kawana ▪ Willamette University ▪ Center for Computational Sciences and Engineering

BoxLib

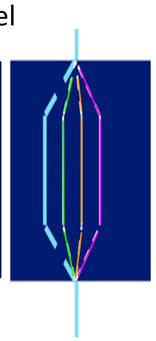
- Software framework for massively parallel structured grid PDE simulations
- Implemented as layered C++ / Fortran
- My work involved implementing hybrid parallelism: MPI + OpenMP



Loop Level OpenMP

- Include OpenMP directives around work loops
- Splits up loop iterations over different threads to be done in parallel

```
!$omp parallel do private(i)
do j = 1, 16
        do i = 1, 16
            // work happens
        end do
end do
!$omp end parallel do
```



```
do j = 1, 4
        do i = 1, 16
            // work happens
        end do
end do
```

```
do j = 5, 8
        do i = 1, 16
            // work happens
        end do
end do
```

```
do j = 9, 12
        do i = 1, 16
            // work happens
        end do
end do
```

```
do j = 13, 16
        do i = 1, 16
            // work happens
        end do
end do
```

Tiling

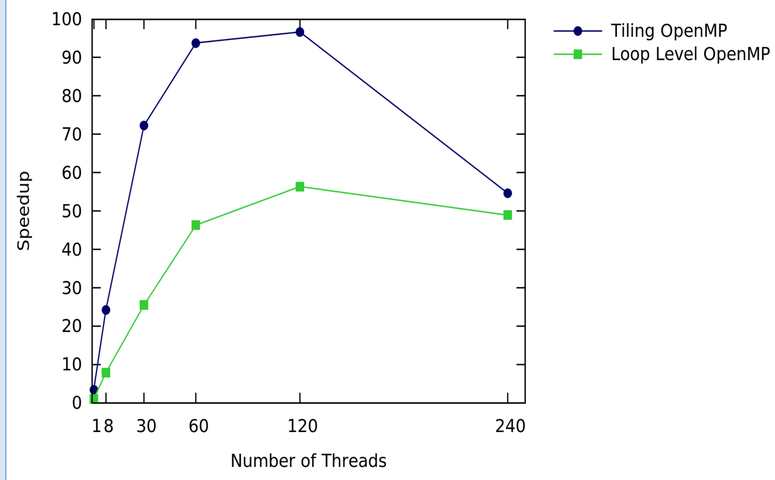
- Parallelization by region, i.e. “tiles”, instead of loop iteration
- Occurs at a “higher level” in the code
  - Parallelism starts before call to subroutine
  - Loops within subroutine adjusted to bounds of tile

```
!$omp parallel
loop over tiles
        get tile box
            call workHappens(tlo,thi)
end loop
!$omp end parallel
```

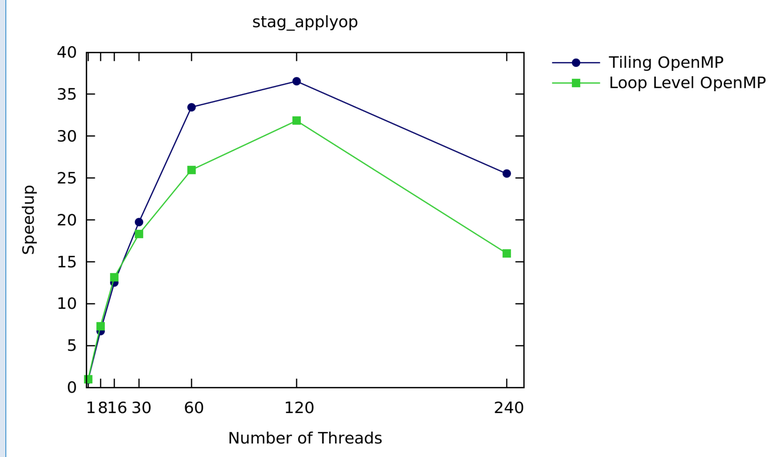
Psuedocode example

Tiling vs Loop Level

Heat Equation Example Speedup on a Babbage Node



FluctHydro (a multi-component flow solver) Subroutine Speed-up on a Babbage Node



*Speedup is in relation to single thread loop level runtimes

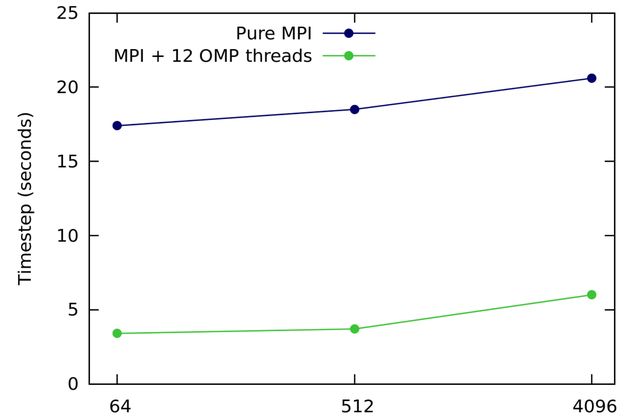
Advantages of Tiling

- Customize tile size to fit in cache
  - Reduces cache misses due to data locality
- Tiling enables better load balancing on architectures with the ability to spawn very large numbers of threads

Weak Scaling Results

- Through profiling of FluctHydro, bottlenecks were identified and removed for better MPI scaling
- Code now scales ~50k processors with MPI + OpenMP

FluctHydro Weak Scaling on Edison



Discussion and Conclusions

- Addition of tiling constructs in combination with OpenMP is more effective than loop level OpenMP
- These hybrid parallelization techniques with tiling are projected to work efficiently on next generation architectures
- Simpler codes such as the heat equation example experience larger gains but production codes such as FluctHydro also show improvement
- Speed-up due to tiling is related to problem size, tile size, and the nature of the subroutine
- Further research may be done to characterize the relation of these factors on various codes

Acknowledgments

I would like to thank my mentor Andy Nonaka, group leader Ann Almgren and the other members of CCSE who helped me throughout the summer