Improving Cache Performance in PICSAR's Maxwell's Equations Solver through Field Tiling

[1]Nigel Tan, [2]Mathieu Lobet, [2]Alice Koniges, [2]Henri Vicenti, [1,2]Dong Li, [2]Jean-Luc Vay
[1]UC Merced, [2]Lawrence Berkeley National Lab

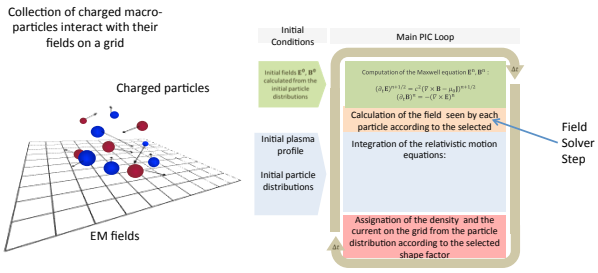
Abstract

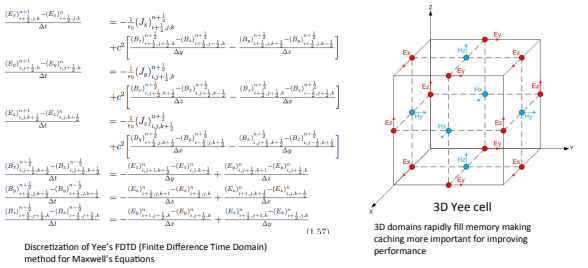
Enforcing cache-coherent shared memory can significantly affect performance in current architectures. This research explores performance benefits in solving Maxwell's equations by refactoring large shared arrays into smaller tiles that fit in L2 caches. Tiles were implemented in two different schemes, one able to operate purely in the caches through guard points and the other partially relying on node-level shared memory. Tests show that the method with guard points is fastest followed by tiling with shared memory. We note that applying the tiling scheme to higher order numerical methods may not have the same performance improvements due mostly to communication costs.

Introduction

PICSAR (Particle-In-Cell Scalable Application Resource) is a proxy app developed to aid PIC codes in their adaptation to evolving large scale architectures such as Cori by providing high performance parallel subroutines. We concentrate here on the Maxwell solver portion of the PIC algorithm. Yee's scheme is a 2nd order FDTD (finite difference time domain) method that advances Electric and Magnetic (EM) fields in time and space. E and B fields are computed on a staggered grid. Each field only requires a set number of points from the other field and current, the number of which is dependent on the methods order (2 for 2nd order scheme) .

The initial implementation applies both MPI and OpenMP for parallel performance but suffers from large amounts of cache misses due to the large size of the field arrays. Shared memory accesses make caching ineffective due to the constant need to pull data from socket-level shared arrays.





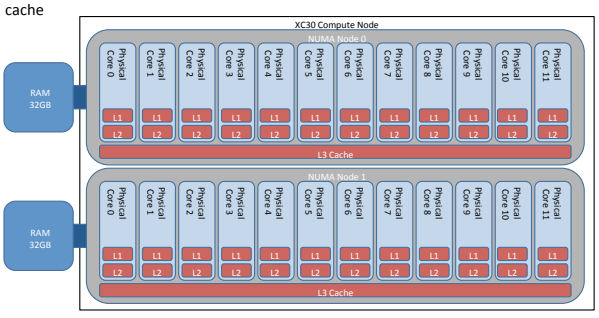
Discretization of Yee's FDTD (Finite Difference Time Domain) method for Maxwell's Equations

3D Yee cell

3D domains rapidly fill memory making caching more important for improving performance

Hardware

Tests were conducted on the NERSC Edison Cray XC30 supercomputer using 2 compute nodes each with 2 Ivy Bridge 2.4GHz processors with 12 cores each for a total of 24 cores per node. Each core contains a 64KB L1 cache, a 256KB L2 cache, and a shared 30Mb L3 cache



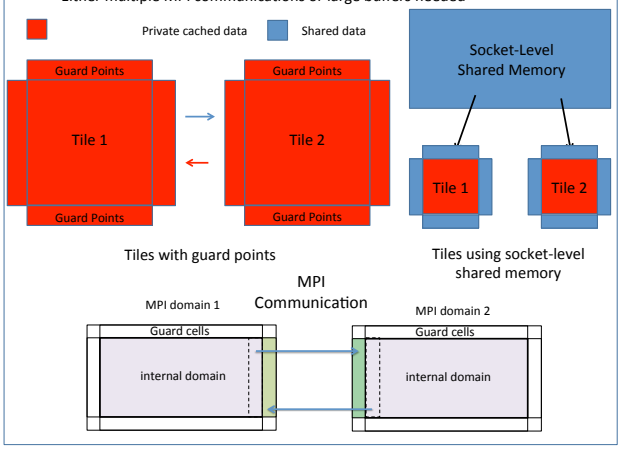
Edison Compute Node Diagram

Implementation Methods

Tiles must contain
- 3D arrays for Ex, Ey, Ez, Bx, By, Bz, Jx, Jy, Jz
- Max/Min coordinates within MPI domain

Yee's method estimates the value with the perpendicular components from the other field, the number of points determined by the methods order.

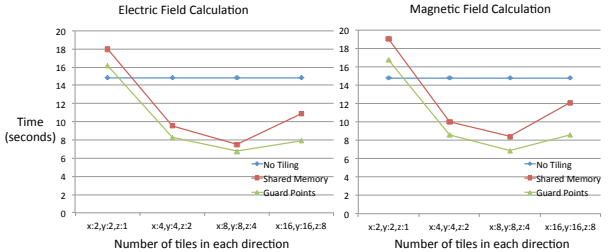
Two possible options
- Only contain necessary points using shared memory for outside data
  - Inner points benefit from cached data
  - Outer points must access shared memory for calculations
  - Simple MPI communication
- Extend arrays for copies of necessary data from neighboring tiles
  - Can calculate all points without shared memory
  - Requires extra communication between tiles
  - Either multiple MPI communications or large buffers needed



Results

The data show clear improvement with tiling for the solver portion of the proxy app. The difference between the shared and guarded versions is smaller because the shared scheme only uses internal data for the majority of the points. There is very little difference between the electric and magnetic field solvers due to the similarities in the calculations with the only difference being the current which doesn't need access to shared memory.

Edison performance for different schemes running with 2 MPI processes (1 per socket) and 10 OpenMP threads per MPI task



Discussion

The general trend was observed as expected with regard to the pure solving step. However, proper communication for the guard version requires either copying the small pieces of tile data to a global buffer for MPI communication or many smaller MPI communications. Both paths have good and bad points and need further investigation. Another interesting avenue is the difference between the tiled version with varying solver method orders. For smaller order methods the guard version has an advantage but large orders can severely inflate array size and limit the benefits of the cache. The shared version also has drawbacks with larger order methods as it increases the amount of shared memory access.

Conclusions

These findings show an interesting start to cache performance improvements for the Maxwell solver in PICSAR. The actual best scheme is difficult to determine due to communication and problem definition issues. Further study into the various issues and solution will yield more interesting future results.