Final Report for Phase 1 Enhanced Intellectual Services – Direct PRAC Support -- Petascale plasma physics simulations using PIC codes

Team Lead: Warren B. Mori

Other Personnel: Dr. V. K. Decyk, Dr. F. S. Tsung

Affiliation: University of California, Los Angeles

The goal of this proposal is to explore new particle-in-cell (PIC) algorithms that will take advantage of the unique hardware configuration on Blue Waters. The three major goals are:

(1) Explore the use of SSE/AVX extensions to speed up single node performance of the OSIRIS code.
(2) Extend the development of our MPI/GPU PIC framework (UPIC).

The results from these activities are summarized below.

(1) Optimize OSIRIS for the Intel Processor using SSE/AVX vectorization, and demonstrate good parallel scaling on the full Blue Waters supercomputer

The CPU part of this project involves adding SSE/AVX vectorization to our production code OSIRIS. There are two main stages to this task. In the first stage, we ported OSIRIS to the Blue Waters and demonstrates good strong and weak scaling, and in the second stage, we implemented SSE instructions in the known hotspots of the code and demonstrates good performance on nearly the entire Blue Waters machine.

We performed strong scaling studies on the Blue Waters machine in both 2D and 3D (OSIRIS has run-time polymorphism which allows it to run in 2D more efficiently). In our 2D scaling study, there are 16384 x 16384 grids and 128 particles per grid (256 million grids and 32 billion particles total) (which is the typical setup for our 2D simulations for long pulse laser plasma interaction simulations). In our 3D simulations, there are 1024x1024x1024 grids and 32 particles per grid (totaling 32 billion particles). The 3D benchmark mimics a typical large 3D LWFA simulation. On > 256,000 processors, where there are only ~120,000 particles per core (much less than that in a typical simulation), OSIRIS is 93% efficient without SSE acceleration (where the code execution time is longer and therefore the parallel overhead is less) and 87.5% efficient with SSE optimizations turned on in 2D, and 89% efficient without SSE and 78.8% efficient with SSE in 3D. These results are summarized in Figure 1.

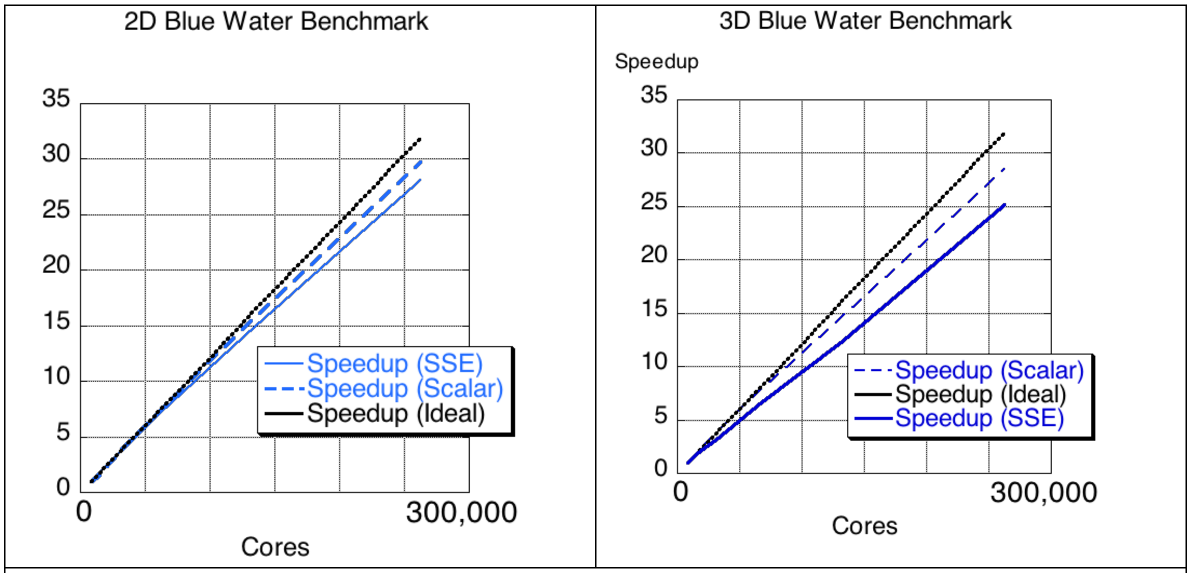


Figure 1: OSIRIS strong scaling results on Blue Waters supercomputer. In each of these simulations there are 32 billion particles total. In the 2D simulations there are 256 particles per cell (which is typical of our large 2D LPI simulations) and in the 3D simulations there are 32 particles per cell (which is typical for our 3D LWFA simulations)

The most time consuming parts of OSIRIS (the charge deposition and the orbits integration) is converted to include SSE and AVX vector instructions. The vector version of OSIRIS achieves a speedup of 1.9 to 3.8 using SSE vectorization depending on the order of the particle shape function used for particle interpolation. As the order increases the current and force interpolation calculations become more complicated and the computational intensity increases, thereby increasing the effectiveness of the SSE instructions. These results are summarized in Figure 2.

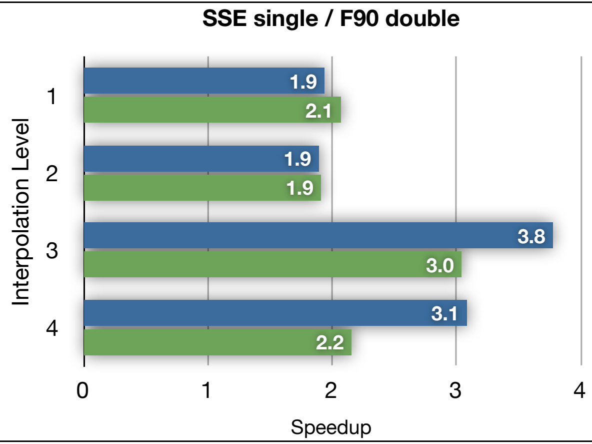
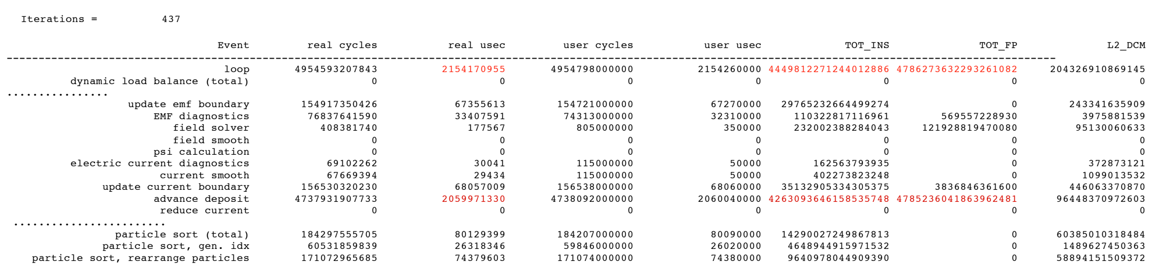


Figure 2: Speedup of the SSE code over scalar OSIRIS for various shape functions. Ideally the SSE should be 4 times faster than the scalar code. For higher order particle shapes where the computational intensity is higher, OSIRIS with SSE instructions achieved speedup which is comparable to that limit.

The vectorized version of OSIRIS was benchmarked on the entire Blue Waters supercomputer, on 772,480 cores. The simulation uses 32 x 32 x 32 grids per core (25.3 billion grids total) and 400 particles per cell (10.12 trillion particles total) and ran for 437 steps. The PAPI output from this simulation is included in table 2. On the Blue Waters supercomputer, the code sustained a speed of 2.2 PetaFLOPS for a period of > 40 minutes. During that time, OSIRIS executed nearly 4.8 10^{18} floating instructions and pushed close to 4.4 10^{15} particle-steps.



```
  Iterations =          437

                                 Event         real cycles           real usec         user cycles           user usec             TOT_INS              TOT_FP              L2_DCM
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
                                  loop       4954593207843          2154170955       4954798000000          2154260000 4449812271244012886 4786273632293261082     204326910869145
            dynamic load balance (total)                   0                   0                   0                   0                   0                   0                   0
................
                   update emf boundary        154917350426            67355613        154721000000            67270000   29765232664499274                   0        243341635909
                       EMF diagnostics         76837641590            33407591         74313000000            32310000     110322817116961        569557228930          3975881539
                          field solver           408381740              177567           805000000              350000     232002388284043     121928819470080         95130060633
                          field smooth                   0                   0                   0                   0                   0                   0                   0
                       psi calculation                   0                   0                   0                   0                   0                   0                   0
          electric current diagnostics            69102262               30041           115000000               50000        162563793935                   0           372873121
                        current smooth            67669394               29434           115000000               50000        402273823248                   0          1099013532
                update current boundary        156530320230            68057009        156538000000            68060000   35132905334305375       3836846361600        446063370870
                       advance deposit       4737931907733          2059971330       4738092000000          2060040000 4263093646158535748 4785236041863962481       9644837097260 3
                        reduce current                   0                   0                   0                   0                   0                   0                   0
 ........................
                  particle sort (total)        184297555705            80129399        184207000000            80090000   14290027249867813                   0      60385010318484
                particle sort, gen. idx         60531859839            26318346         59846000000            26020000    4648944915971532                   0       1489627450363
     particle sort, rearrange particles        171072965685            74379603        171074000000            74380000    9640978044909390                   0      58894151509372
```

Figure 3: PAPI output from our 2.2 PFlops test. In this benchmark, there are more floating point operations than instructions due to the use of SSE vector instructions in the optimized version of OSIRIS. This simulation uses more than 10 trillion particles and uses nearly all of the Blue Water CPU cores.

(2) Extend the development of our MPI/GPU PIC Framework

The GPU part of this project involves porting a compact skeleton 2-1/2D Electromagnetic Particle-in-Cell (PIC) code to a cluster of GPUs. Our design plan has 3 stages. The first stage was to implement a simpler 2D skeleton Electrostatic code on a single GPU. The second stage was to implement a 2D Electrostatic code running on multiple GPUs using MPI to connect the GPUs. The third and final stage was to add the additional features needed for the electromagnetic code. All the codes divide space into tiles that are small enough to fit into the shared memory of the device and to process the particles in each tile independently [1]. Two different approaches were used on the single GPU codes: one algorithm uses one thread per tile and has no data collisions. The other uses a thread block per tile and needs efficient mechanisms for resolving data collisions. The collision-free algorithm works best on the older GT200 architectures, whereas the collision-resolving algorithm works best on the Fermi.

In the third quarter (ending in April 15th), we completed the only remaining task from the first stage: the addition and replication of guard cells (ghost cells) for the field quantities. We implemented 2 different versions of the addition procedure, one each for the collision-free and collision-resolving data structures. Only one version was needed for the replication of guard cells.

We then implemented 2 versions of the 2-1/2D Electromagnetic code on a single GPU, with both the collision-free and collision-resolving algorithms. Altogether 26 new Cuda C procedures were implemented.

In the push procedures, two different approaches were used. One created a list of particles which had to be moved to another tile as part of the push. The other approach created the list as part of the reordering scheme. Creating the list as part of the reordering requires reading a long array of particles just to determine which ones needed special processing, and this dominates the reordering time when the tile size is large. Creating this list in the push avoids reading global memory again, since the push procedure has already read the particle array. However, adding the list creation to the push also requires the use of more registers, a scarce resource, and thus reduces occupancy (the number of independent threads which can run concurrently). As a result, sometimes one scheme is faster and sometimes the other. In addition, there were 2 versions of the push, one with relativity included (which requires the calculation of divides and square roots) and one without which does not require these calculations. The current deposit also updates the particles half a time step, so there are multiple versions of those as well.

The 26 new procedures were distributed as follows:

8 push procedures, 4 each for the collision-free and collision-resolving algorithms.
8 current deposit procedures, 4 each for the collision-free and collision-resolving algorithms.
5 guard cells procedures
5 field solving procedures

In addition, 5 procedures and the FFT from the electrostatic code were reused.

The best results for the 2-1/2D Electromagnetic code on the M2090 show the following:

```
Push Time:            0.43 nsec/particle/time step
Deposit Time:         1.01 nsec/particle/time step
Reordering Time:      0.76 nsec/particle/time step
Total Particle Time: 2.20 nsec/particle/time step
(48x total particle speedup compared with 2.67 GHz Intel i7)
```

The best results on the M2090 were obtained with the collision-resolving scheme, where the list creation was carried out as part of the reordering, with a tile size of 16x16 and cuda blocksize of 128. Relativity was turned on for this benchmark, but there was very little difference (2%) compared to relativity turned off: it appears the cost of the square roots and divides on the GPU was very small. (On the host machine, relativity typically costs about 30%.) The collision-free scheme with a tile size of 2x2 was about 10% slower than the collision-resolving scheme. About 10% additional time was spend on solving the electromagnetic fields, which included 10 FFTs/time step. On the older T10 architecture, the collision-free scheme was best, with a tile size of 1x2, and was about 2.4x faster than the collision-resolving scheme. All the results are for single-precision.

We did not work on multiple GPUs this period. Stage 1 is now complete. The only remaining task for stage 2 is to implement the addition and replication of guard cells (ghost cells) for the field quantities. The single GPU version of the electromagnetic code for stage 3 is now complete, and the multiple-GPU version remains to be started.

In the most recent period (from April 15 onward), we completed the only remaining task from the second stage: the addition and replication of guard cells (ghost cells) for the field quantities, distributed across multiple GPUs. This required the implementation of 4 guard cell procedures on the GPU, two for adding charge density, and 2 for copying the electric fields, and two procedures written in MPI to move edge data from one GPU to another.

The best results for the 2D Electrostatic code on the M2090 show the following:

```
Benchmark size: 2048x2048 grids, 150,994,944 particles

Electrostatic Code, mx=16,my=16, with dt = 0.1
             CPU:Intel i7    1  GPU       2 GPUs       3 GPUs
Push            22.1 ns.    0.326 ns.    0.164 ns.    0.109 ns.
Deposit          8.5 ns.    0.233 ns.    0.118 ns.    0.079 ns.
Reorder          0.4 ns.    0.442 ns.    0.222 ns.    0.148 ns.
Total Particle  31.0 ns.    1.003 ns.    0.505 ns.    0.338 ns.
Total particle speedup on 3 GPUs was about 92 compared to 1 CPU.
The field solver took 5%, 19%, and 27% of the total time, as the number of GPUs
increased from 1 to 3.
```

Stages 1 and 2 are now complete. The single GPU version of the electromagnetic code for stage 3 is now complete, but the multiple-GPU version remains to be started.

In addition, a paper based in part on this project, was submitted for publication, It is entitled, “Particle-in-Cell Algorithms for Emerging Computer Architectures,” by Viktor K. Decyk and Tajendra V. Singh. A preprint is available at the following web site: https://idre.ucla.edu/hpc/research.

References:

[1] Viktor K. Decyk and Tajendra V. Singh ,"Adaptable Particle-in-Cell algorithms for graphical processsing units," Computer Physics Communications 182, 641 (2011)