Motivation

- ORBIT was developed 30 years ago to study the drift orbit effects of charged particles in the strong equilibrium fields of tokamaks.
- Recently there is more need for lengthy simulations.
- Goal is to maximize performance while minimizing changes made to the FORTRAN code.

Techniques

OpenMP – shared memory multiprocessing on the CPU

- Compiler directives are added as comments:
  ```fortran
  !$OMP PARALLEL DO PRIVATE(npt)
  DO 10 J = 1, 100
     NPT = J * J
     CALL NSQRS(J)
  CONTINUE
  !$OMP END PARALLEL DO
  ```

- Same code can be compiled without OpenMP.
- Threads are created when entering a "parallel region".
- Clauses specify whether variables are shared or private.
- APIs provide additional capabilities.

CUDA Fortran – Parallel processing on the GPU using separate code

- Programmer writes separate code for the host (CPU) and device (GPU).
- GPUs perform sets of instructions known as kernels.
- The host calls device code by specifying the dimensions (number of threads per block) of the kernel.

Techniques (cont.)

OpenACC 1.0 – Parallel processing on the GPU using Compiler Directives

- Similar to OpenMP directives and clauses:
  ```fortran
  !$ACC PARALLEL COPY(npt)
  DO 10 J = 1, 100
     NPT = J * J
     CALL NSQRS(J)
  CONTINUE
  !$ACC END PARALLEL LOOP
  ```

- Directives specifying "Data Regions" copy data between host (CPU) and device (GPU).
- Parallelism divided into three levels – gang, worker, and vector.
- Synchronization is only possible within each gang.
- Vector-level parallelism uses stream processors to run the same instructions on multiple data (SIMD).

Results

Testing

We tested the code using the following procedure:
1. Run the original ORBIT code 3 times on the Dawson cluster for total particle count (nprt) set to 5000 and 3 times for nprt = 25000.
2. For each implementation, determine a set of configurations to test.
3. Test each configuration 3 times for nprt = 5000 and 3 times for nprt = 25000.
4. Compare the best possible configuration of each implementation against the original runtime.

Results

- Speedup of the ORBIT code for different configurations.
- Comparison of the best possible configurations for each implementation.
- Performance of the GPU compared to the CPU.

Conclusion

- The OpenMP One-Region and the CUDA Fortran Fully-Optimized versions were combined into the final code, since the former is the fastest non-GPU version and the latter is the fastest GPU version. The final code would select the implementation based on whether the machine has a GPU.
- Future work could attempt using the newly released OpenACC 2.0 to more success.

References


Acknowledgements

This work was supported by the National Undergraduate Fellowship, funded by the Department of Energy, and made possible by the Princeton Plasma Science Education Department. We also thank Jianying Lang for her expertise.