Validation of MHD Code for Gravitationally Stratified Media on Graphical Processing Units.

Introduction

The Sheffield Advanced Code (SAC) is a novel fully non-linear MHD code, designed for simulations of linear and non-linear wave propagation in gravitationally strongly stratified magnetised plasma.

Ref. http://adsabs.harvard.edu/abs/2008A%26A...486..655S.

SAC is developed using Fortran and is built using the Versatile Advection code (VAC). With the increasing need to model problems with increasing resolution, higher Magnetic Reynolds number there is a requirement for increasing amounts of storage and CPU cycles with which to solve such problems.

In this document we report work which has been undertaken to enable the Sheffield Advanced Code to run on massively parallel compute architectures such as General Purpose Graphical Processing units (GP/GPU) and multi-core processing systems. We describe the basic architecture of SAC-MP and present timings illustrating the performance benefits of running SAC on multi core systems. Specificaly we present results for the NVIDIA C1060 (tesla) GP/GPU

Backgound

For many years super computing was restricted to dedicated facilities, however, the advent of cluster computing lead to the increased availability of high levels of compute cycles, thus enabling researchers to investigate increasingly challenging problems. The evolution of systems architectures and CPU design continues to evolve. Currently there are two key developments

- Computer processors with multiple cores,
- General Purpose Graphical Processing Units.

With technology shrinking to 18nm it is expected that processors will have around 48 cores it therfore be necessary to enable applications to fully exploit these multi-core systems. It is envisaged that applications developed using OpenCL and CUDA will be able to efficiently exploit the hybrid environment comprising multiple GPU's and multi-core CPU's. For multimedia and gaming systems it was found that a significant work load was used in the rendering of imagery. Given this scenario developers recognised the need for a massivley parallel architecture capable of computing the pixel characterisitcs for high resolution display systems. It was not long before computational scientists exploited such architectures for their particular numerical problems. Recognising this development, NVIDIA introduced the Tesla GPU architecture and the Compute Unified Device architecture (CUDA) for parallel computing. An additional development was OpenCL, this is a framework which enables developers to write applications which run across heterogeneous platforms consisting of CPU's, GPU's and other processors. CUDA-enabled graphics processors operate as co-processors within the host computer. This means that each GPU is considered to have its own memory and processing elements that are separate from the host computer.

Unlike CPUs, GPUs have a parallel throughput architecture enabling the execution of many concurrent threads slowly, rather than executing a single thread very quickly. These threads execute on many processing cores which comprise the GPU. To perform useful work, data must be transferred between the memory space of the host computer and CUDA device(s), for this reason, performance results must include IO time to be informative. The code described here was run on an NVIDIA Tesla C1060 GPU. This consists of 240 proessing cores each running at 1.3GHz these processing cores can support upto 65535 threads of execution. This determines the problem sizes which may be executed on the GPU.

Profiling of the code indicated that one of the most computationally intensive parts of the application is the computation of hyper-viscosity and hyperdiffusion source terms. One of the reasons for this is the necessity to compute global maxima for the hyperviscosity and wave speeds.

The Sheffield Advanced Code

The Sheffield Advanced Code is a numerical MHD tool that allows the simulation of the interaction of any arbitrary perturbation (not necessarily

limited to the linearised regime) with the background plasma in magnetohydrostatic equilibrium. The MHD equations for the perturbations of density, internal energy and

magnetic field strength are redefined by taking into account the magnetohydrostatic equilibrium condition. Hyperdiffusion and hyperresistivity

are implemented to achieve numerical stability of the computed solution of the MHD equations.It was found advantageous to use

this, apparently more complex form of the MHD equations for numerical simulations of processes in a stable gravitationallystratified plasma. The method was implemented on the base of the well-known Versatile Advection Code (VAC; Tóth 1996).

Structure of Source Code

SAC-MP was developed using the CUDA toolkit and consists of a collection of source files which run on the host system or are kernel sources running on the GPU. The SAC initialisation routines is used to generate the initial configurations for input into SAC-MP. The SAC-MP routine consiss of a main routine which loads the initial problem onto the GPU host memory and calls kernel routines evolving the conserved quantities evolved by the MHD equations with control of numerical diffusion using hyperdiffusion source terms. Data is copied from the GPU host memory when it is necessary to save a particular configuration or to communicate neighbouring cells between different GPU's in a multi-GPU configuration, All computations are performed double prescision, however a compiler switch easily enables the application to switch between single precision and double precision.

The kernel routines are divided into a number of individual kernel routines comprising the following groups

- initialisation,
- compute flux terms,
- compute the hyper viscosity,
- compute the hyperdiffusion source terms,
- update routine and
- finalisation (frees memory on the GPU).

The kernel routines are called from the main driver routine.

An important feature of the code is method used to speed up the computation as indicated in the background there are upto 65535 threads running on 240 processing cores. It is necessary to distribute the computational effort across these elements. In the simplest scenario for a hydrodynamics problem solved over a grid each grid point will be updated by

a single thread. But is is clear that it may be neceesary to run much larger problems (larger than 65535 grid points), so , each thread must update many grid points. We refer to the grid-occupancy as the number of grid points computed for each thread.

Performance Tests

Tests were performed by running two codes

1. The SAC code which is optimised for the Sheffield central HPC facility
2. and the SAC-MP code running on a Tesla C1060 GPU.

For the tests running using the original SAC code tasks were run on a single core of a dual core AMD Opteron (2218 HE) 2.6GHz processor. Two test models were run.

1. Orszag-tang test
2. Brio-Wu shock tube test

For the tests performed here it was observed that the time required for transferring data between the host and GPU was much less than the time for a single iteration. It should also be noted that in designing this application we have attempted to minimise the number of data transfers between the host and the GPU.

The tests were performed by measuring the time to complete 100 iterations for models of difrerent sizes. For the Orszag-Tang test the model is run on a square grid and we give the dimensions of just one side of the grid. For some problem sizes it was necessary to increase the grid-occupancy to values greater than 1. For most of the problems it is necessary to stabilise the solutions by enabling hyperdiffusion (case 1) however, for performance testing purposes the problem was allowed to run with hyperdiffusion switched off. Each test was performed for these two scenarios

1. Hyperdiffusion enabled
2. No Hyperdiffusion

Figure 1 and 2 show measurements for the time taken to run 100 iterations, over different model sizes, for the two different hyperdiffusion scenarios and for the two different tests. It can be seen that the computation requires less time using the C1060 GPU and that the times scale in a similar way for both the GPU and CPU. What is notable is the difference in time measured (for 100 iterations) between the scenarios with and without hyperdiffusion. Also notable is the prominent kink in figure 1 around the 400 model size. This occurs as a result of the need to increase the grid-occupancy for the GPU.

Figure 3 shows the speed up factor versus the model size, the speed up factor is defined as the time measured for 100 iterations on the AMD Opteron CPU divided by the time taken for 100 iterations on the Tesla C1060 GPU. What is notable is the difference in speed up between the scenarios with and without hyperdiffusion. As the grid-occupancy increases the speed up tails of

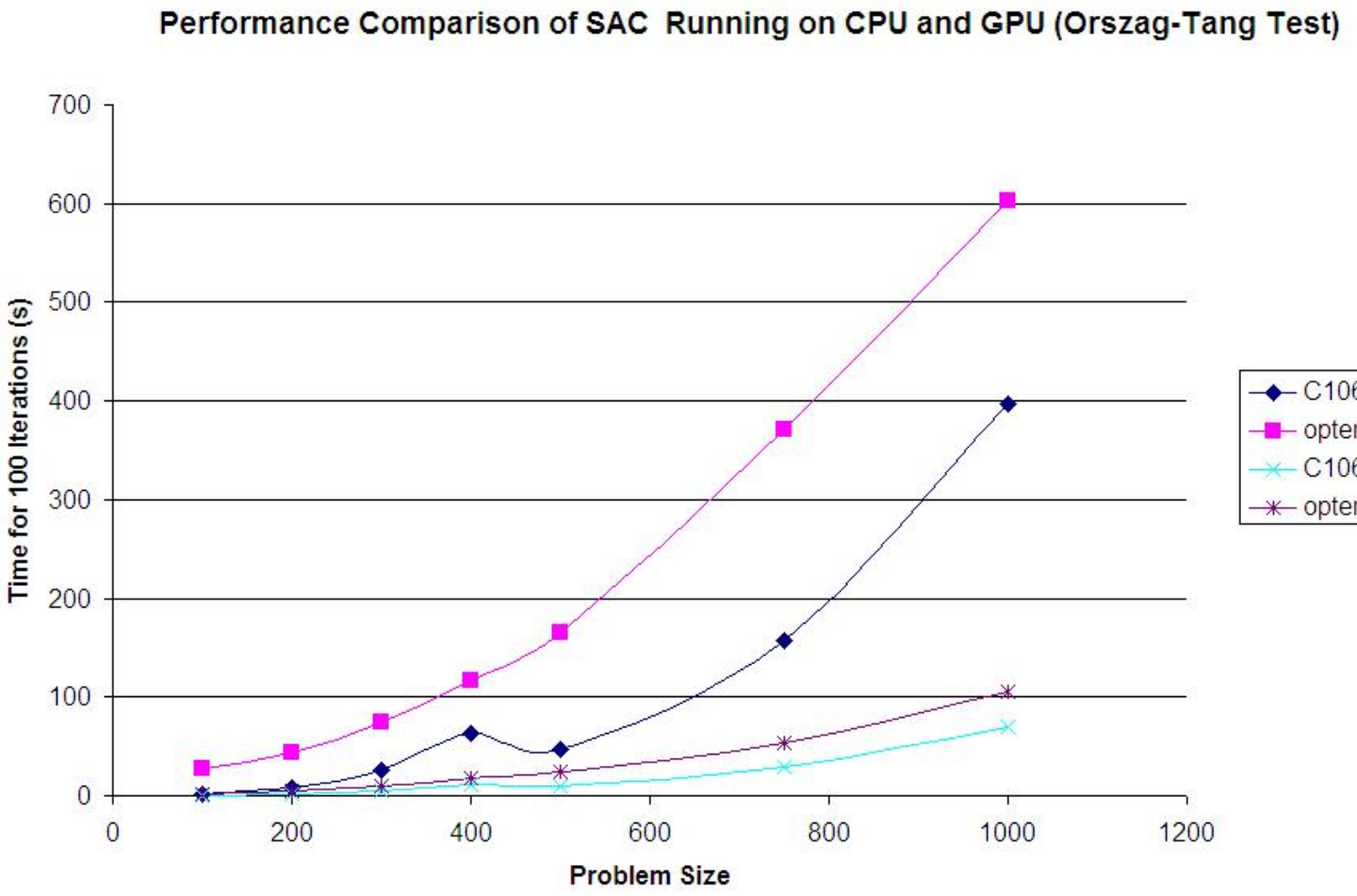

Figure 1. Performance Comparison of SAC runnning on CPU and GPU (Using the Orszag-Tang test)

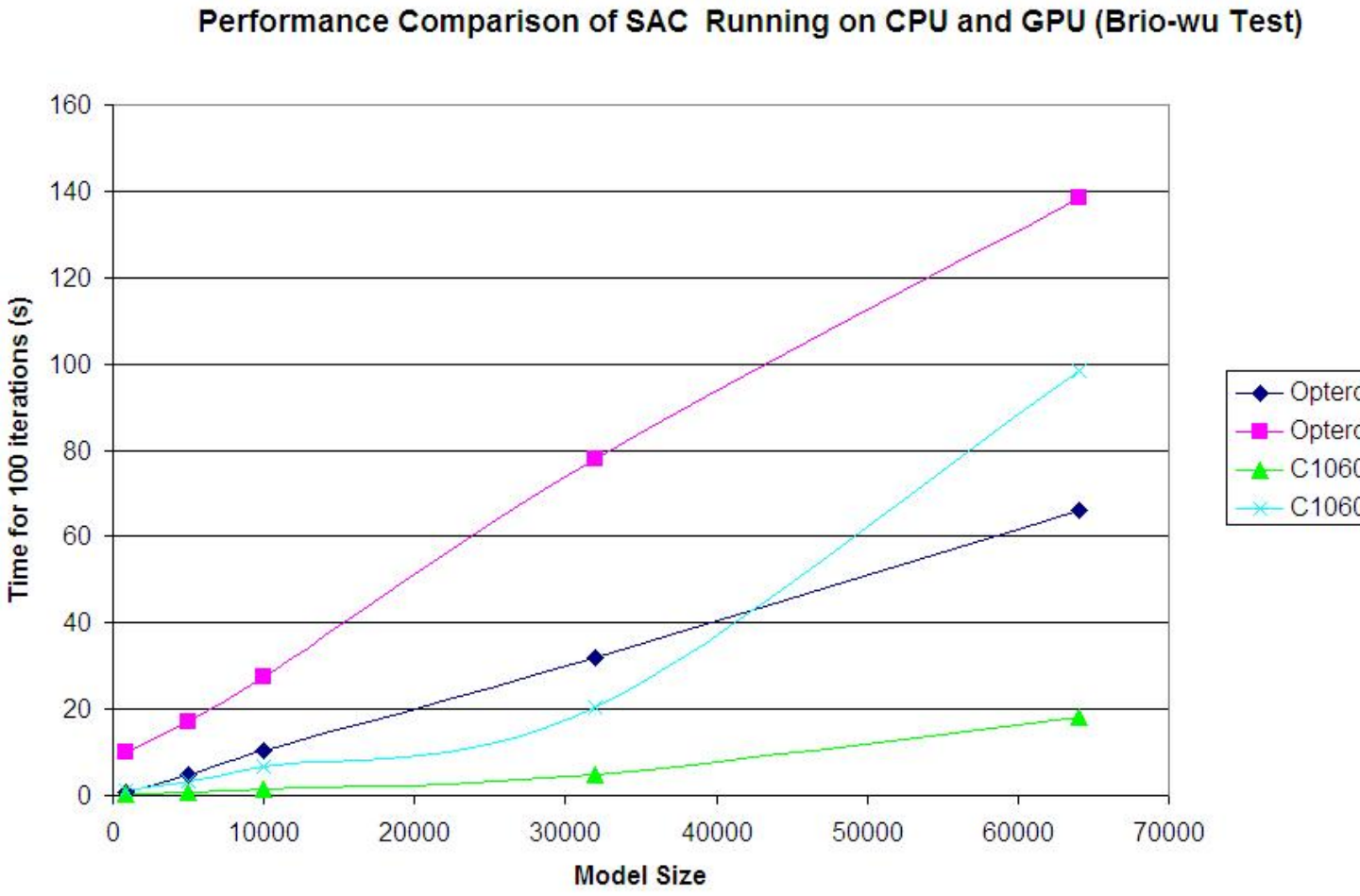

Figure 2. Performance Comparison of SAC runnning on CPU and GPU (Using the Brio-Wu test)

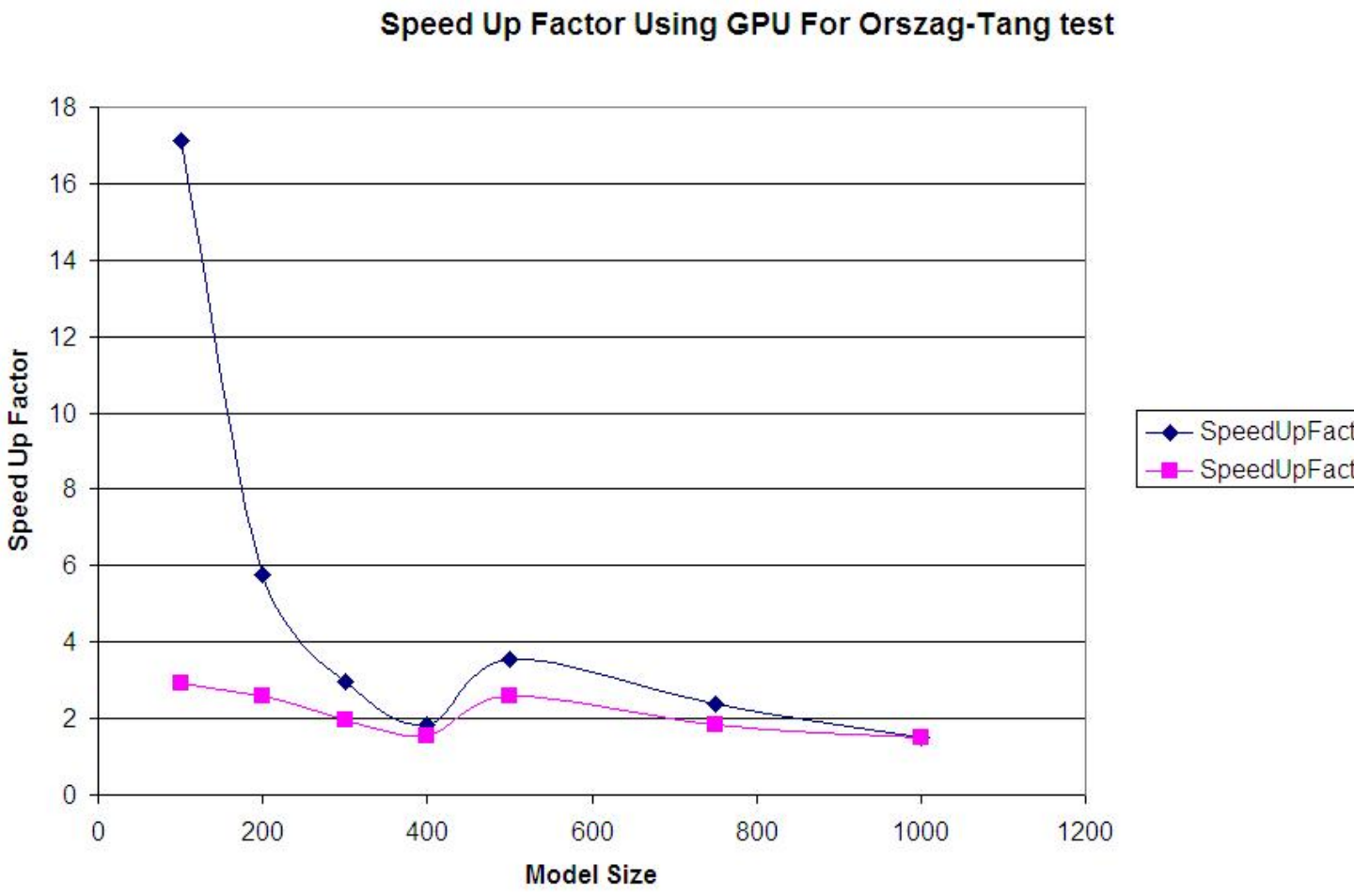

Figure 3. Speed up factor Acheived Using the NVIDIA Tesla C1060 GPU (for the Orszag-Tang test)

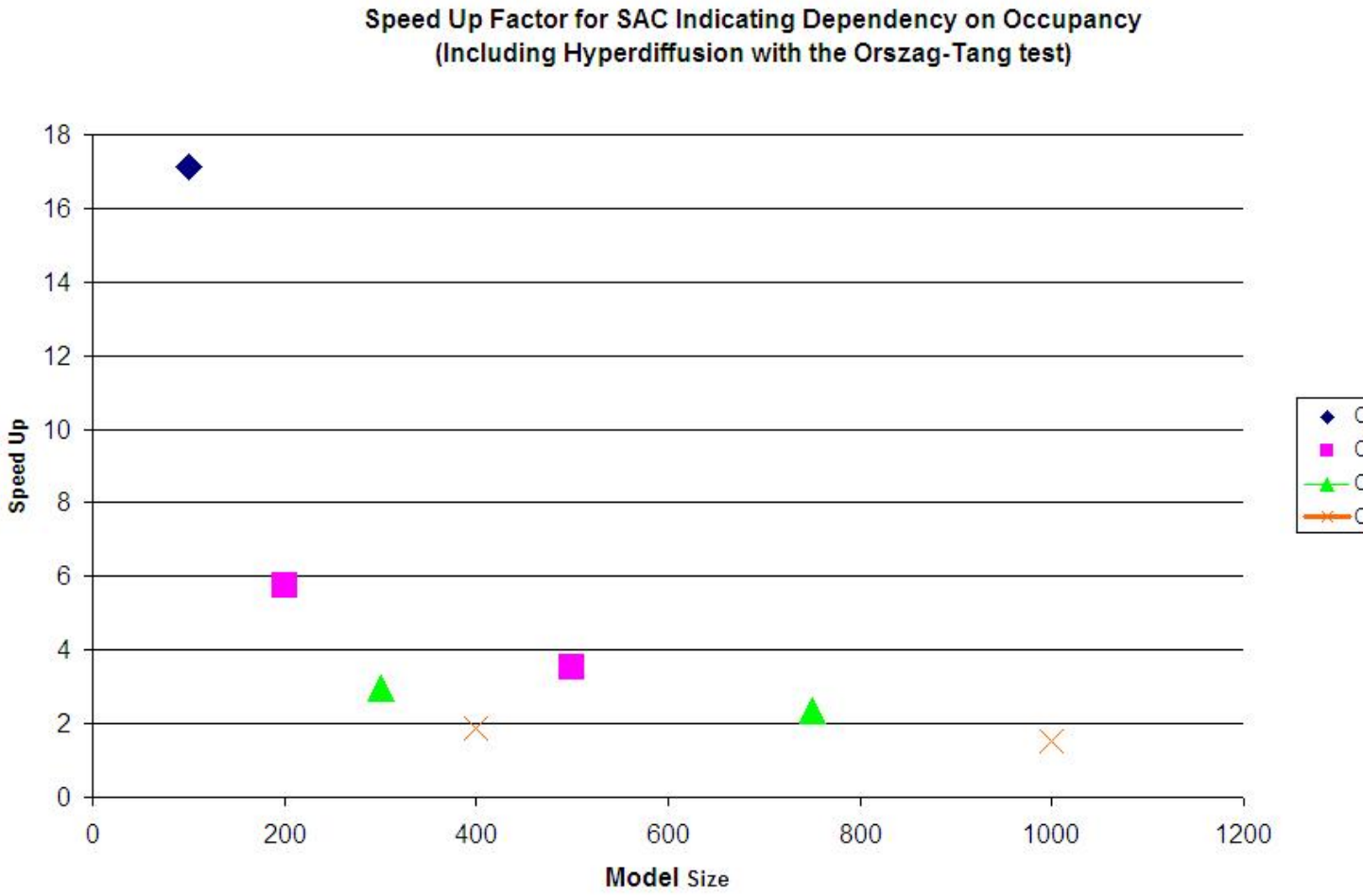

Figure 4. Speed up factor Acheived Using the NVIDIA Tesla C1060 GPU and Comparing Different Grid Occupancies (for the Orszag-Tang test)

Conclusion

Unlike CPUs, GPUs have a parallel throughput architecture enabling the execution of many concurrent threads slowly, rather than executing a single thread very quickly. These threads execute on many processing cores which comprise the GPU. The SAC-MP code

described here was run on an NVIDIA Tesla C1060 GPU. This consists of 240 proessing cores each running at 1.3GHz these processing cores can support upto 65535 threads of execution. This determines the problem sizes which may be executed on the GPU.

The tests demonstrate that by running SAC on the Tesla C1060 GPU models in the size range investigated will require half the execution time as compared to the dual core AMD Opteron (2218 HE) 2.6 GHz CPU. These tests were performed using computations with double precision, it is easy to switch the application to single precision. It is well known that the GPU single precision performance exceeds that of the double precision performance hence we expect an order of magnitude improvement in performance for the single precision version. One possibility is to allow the code to switch between single precision and double precision operation (*1) for those parts of the problem exhibiting instability.

The tests performed here were undertaken using the Tesla C1060, however it has been recognised that there is a requirement improved double precision performance, this was enabled using the Tesla 20 Series (know as Fermi). The Tesla 30 series (codename Kepler) is expected at the end of 2011. The table below illustrates some differences between the Tesla10 and Tesla 20 series, what is notable is that we would expect to imporve the double precision performance by an order of magnitude, we also expect performance gains through the use of reduce grid-occupancy, the possibility of larger model grids and through the use of atomic operations for the computation of global maxima.

	Tesla 10 -series GPU	Tesla 20-series GPU (Fermi)
Number of Cores	240	448
FP Peak Performance	933GFlops (single), 78 GFlops(double)	1.03TFlops (single), 515GFlops(double)

In summary a number of improvements are expected

- Execution on Tesla 20 series
- Computation of global maxima using atomic operations
- Better usage of GPU shared memory

We have successfully developed and tested a mapping of the SAC FORTRAN code onto the NVIDIA compute unified device architecture (CUDA) this represents an excellent investment in code development as the codes may easily be converted to the OpenCL standard and executed on multi-core CPU systems. With a promising start using teh Tesla C1060 it is expected that much greater performance improvements may be acheived using the currently available Tesla 20 series (Fermi) .

References

Code for Modeling MHD flows on Parallel Computers: Versatile Advection Code,

Tóth, G. 1996,

Astrophysical Letters & Communications, 34, 245

Dr Dobbs CUDA, Supercomputing for the Masses

Orszag-Tang Test

Brio-Wu Test

SWAN: A Simple Tool for Porting CUDA top OpenCL

M.J. Harvey & G. De Fabritiis

Computer Physics Communications
Volume 182, Issue 4, April 2011, Pages 1093-1099

OpenCL Wikipedia

*1 Thanks Deniz