BEAM DYNAMICS USING GRAPHICAL PROCESSING UNITS*

M.D. Salt†, R.B. Appleby, University of Manchester and the Cockcroft Institute, Manchester, UK
D.S. Bailey, University of Manchester, Manchester, UK

Abstract

Simulation of particle beam dynamics in accelerators is computationally expensive, and requires ray-tracing of high numbers of particles to ensure the accuracy of the result, to model collective effects and to reduce the statistical error. Conventional beam tracking tools operate sequentially on particle phase space to compute the trajectories of particles through many turns around circular and along linear machines. Graphical Processing Units (GPUs) utilise stream processing techniques to dramatically speed up parallel computational tasks, and offer considerable performance benefits to particle beam dynamics processing. In this paper, the application of stream processing to beam dynamics is presented, along with the GPU-based beam dynamics code GPMAD, which exploits the NVidia [1] GPU processor and demonstrates the considerable performance benefits to particle tracking calculations. The accuracy and speed of GPMAD is benchmarked using the DIAMOND [2] Light Source BTS lattice, and the ATF extraction line.

STREAM PROCESSING

Stream Processors

The application of stream processing to the parallel processing of beam dynamics simulations was first pointed out in [3], where the potential gain in processing power was discussed. Conventional beam dynamics simulations are carried out on Central Processing Units (CPUs), which are multifunctional devices with a large proportion of the silicon die devoted to control and cache. Arithmetic floating-point units (FPU) occupy only a small proportion of the die, resulting in comparatively poor floating point performance of the processor. However, stream processors offer improved floating point performance due to a highly optimised single-instruction-multiple-data (SIMD) architecture. Graphics Processing Units (GPU) are a type of stream processor, whose development was driven by the need for ultra-fast image rendering in the computer games industry. Due to the demands of the gaming and image processing industry, GPUs are very powerful: the peak Floating-Point Operations per Second (FLOPS) of the latest GPU is 768 GFLOPS, compared with only 50 GFLOPS for a typical dual-core CPU.

Parallel programming of a stream processor requires a different approach to conventional programming techniques. The subsection of the simulation (kernel function) targeted at the stream processor is written in a dedicated stream processing language.

Early GPUs had closed architectures designed specifically for rendering, but later architectures were opened up, allowing exploitation for alternative purposes. General Purpose computation using GPUs (GPGPU) developed, leading to an array of programming languages. Examples of GPU-based programming languages are Brook [4] and CUDA [5].

Accelerator Physics and Stream Processing

The beam dynamics of the motion of particles through an accelerator is well suited to stream processing techniques. The motion is modelled by representing a particle as a point in a 6-dimensioanl phase space, with positions (x,y,τ), associated canonical momenta (p_x,p_y,p_t) and phase space vector,

$$X = (x, p_x, y, p_y, \tau, p_t)^T . \quad (1)$$

The evolution of the phase space point through a magnetic accelerator elements may be modelled using the second order transport map [6],

$$X_j^{final} = \triangle X_j + \sum_{k=1}^{6} R_{jk} X_k + \sum_{k=1}^{6}\sum_{l=1}^{6} T_{jkl} X_k X_l . \quad (2)$$

Therefore, the complete process of evolving a particle through a single magnetic element requires many multiply-reduce operations. The particles are assumed to be sufficiently relativistic that inter-particle interactions may be ignored, allowing the same operation to be applied to all particles in parallel. With large particle numbers, this is a highly SIMD operation, and thus well suited to the GPU stream processor.

GPMAD - a Stream Processor based accelerator code

GPMAD [7] is a stream processing beam dynamics code, written in C/C++ with CUDA [5] extensions that interface with the GPU hardware. The code is compliant with the XSIF input format, based on the parser from the code BDSIM [8].

GPMAD extracts information from the element list and calculates the zeroth, first and second order transfer maps required for computation in a C++ pre-processor. The particle phase space data is gathered from a seperate file, which GPMAD sends to the stream processing CUDA

* University of Manchester and Cockcroft Institute
† michael.salt@hep.manchester.ac.uk

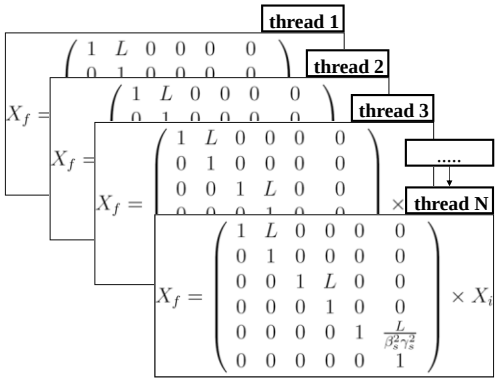

Figure 1: Parallel execution of n threads through a first-order drift matrix element. The operation of the transfer map on n different particle threads occurs simultaneously.

components which perform the linear algebra of the transfer maps. The CUDA implementation handles GPU memory allocation, GPU memory copying operations and the execution of the kernel function. In terms of data structures, the operations on data stored in GPU memory are called threads, which are grouped into blocks, and multiple blocks may be grouped onto grids. GPMAD, as well as performing evolution of particle phase space vectors (tracking), can compute the optical β-functions and perform particle loss calculations with a hard-edged collimation model.

STREAM PROCESSING CASE STUDIES

To verify the effectiveness of stream processors for beam dynamics, the techniques and the GPMAD code described in this paper have been applied to two examples cases - the BTS transfer line in the DIAMOND light source, and the ATF extraction line.

DIAMOND BTS

The 94 element, 68.04 m DIAMOND Booster-to-Storage (BTS) lattice was used to verify precision and performance compared to the code MAD [6].The transfer line contains drifts, sector dipoles, quadrupoles, kickers and collimators, and so is a good test of the linear optics of GPMAD and the performance gains over CPU-based codes. The optical functions computed with GPMAD are presented in figure 2.

The BTS lattice is required to give a small emittance for injection into the main storage ring, which is achieved through collimation of the transverse phase space. GPMAD can perform highly parallel single particle tracking, for computing particle losses and effects of halo phenomena.

To verify the accuracy of GPMAD, the single particle trajectories through the transfer line in both GPMAD and MAD are computed are compared in figure 3. The figure shows the particle trajectory is almost identical in both programs. This is of particular importance because current

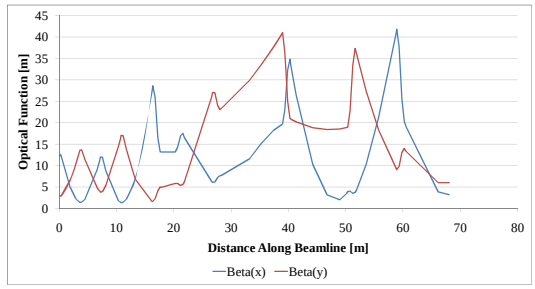

Figure 2: The horizontal and vertical β-functions in the DIAMOND BTS computed using GPMAD.

GPUs are only single-precision devices.

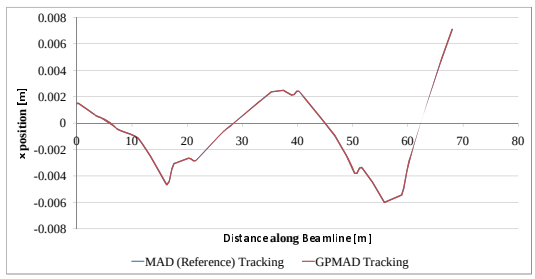

Figure 3: Single particle Tracking in the horizontal phase space co-ordinate x to verify the accuracy of GPMAD against MAD.

Figure 4 demonstrates the deviations of GPMAD relative to a CPU-based version of MAD. The deviation does not exceed 0.025 % at any point throughout the lattice. The behaviour appears to be non-accumulative, which indicates acceptable stability and accuracy for use with longer beamlines.

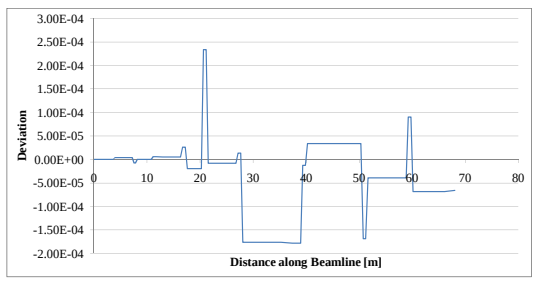

Figure 4: Deviation $\frac{\triangle p_y}{p_y}$ relative to MAD to quantify the accuracy of GPMAD.

Due to the parallel nature of GPMAD, the most benefit is seen with high particle numbers and when performing high statistical accuracy tracking. A bunch of gaussian-distributed particles were evolved through the BTS lattice, and figure 5 compares the distributions at the end of the lattice. For high-statistics tracking, the results from the two codes are identical: the calculated mean and standard deviation differ by less than 0.0001 %, between MAD and GPMAD.

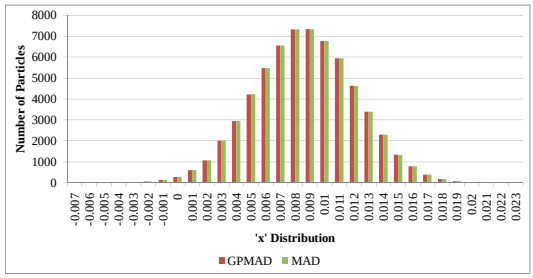

Figure 5: Distribution of x at the exit of the DIAMOND BTS Lattice, computed in both MAD and GPMAD.

The main attraction of GPMAD is the performance benefit, and figure 6 shows a performance comparison of GPMAD running on an NVidia GeForce 8600m GT GPU, with 512 MB of dedicated graphics memory, against MAD running on a notebook computer with a 1.5 GHz Intel Core Duo processor, 2 GB of main memory. The performance benefits of stream processing over conventional CPU-based codes are clear, particularly as the tracked particle number increases. For example, when tracking 4,096,000 particles, GPMAD completes the beam tracking in a quarter of the time. This time saving may be used to further refine a simulation, or to increase the numbers of particles to give better statistical precision.

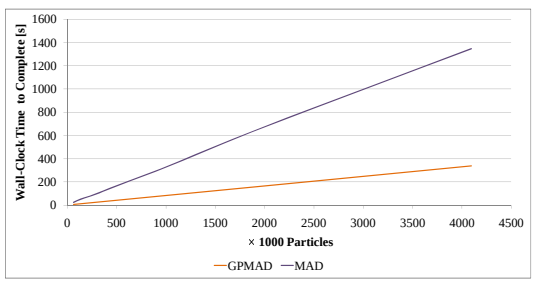

Figure 6: The performance comparison of GPMAD and MAD, when tracking large distributions of particles.

The tracked particles may be collimated due to element apertures, which can be studied with GPMAD using a hard-edged collimation model. Figure 7 shows electron losses generated by GPMAD on a small section of the BTS.

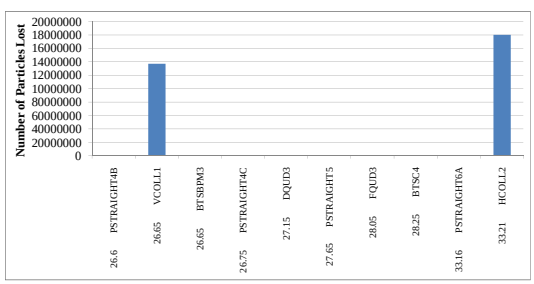

Figure 7: Electron loss rates in the DIAMOND BTS line, as computed using GPMAD.

ATF Extraction Line

To further test the stream processing, GPMAD was used to study beam transport in the ATF extraction line at KEK. This line extracted from the damping ring and provides a series of emittance measurement diagnostics. The beam distribution in the horizontal co-ordinate x plotted at the end of the extraction line in figure 8. The agreement with MAD is excellent.

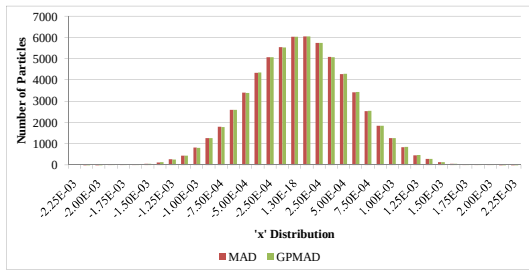

Figure 8: The horizontal beam in x at the end of the ATF extraction line, computed with GPMAD.

CONCLUSIONS

Stream Processing has demonstrated considerable potential to give performance benefits to the study of beam dynamics in particle accelerator. The code GPMAD exploits stream processing to provide accurate particle tracking, with a large performance benefit over CPU-based codes. The greatest performance gains are realised where the number of particles to be tracked is high, and will be of benefit when studying problems in beam dynamics requiring high statistical accuracy or many turns of a circular machine. Stream Processing is an exciting and rapidly developing field, and in accelerator physics there are many problems that may benefit from this parallel method of computation.

REFERENCES

[1] Nvidia ® Corporation `www.nvidia.com`.

[2] Diamond Light Source, Oxfordshire `www.diamond.ac.uk`

[3] R. B. Appleby et al., "High-Performance Stream Computing for Particle Beam Transport Systems", Computing for High Energy Physics Conference 2007 - 66

[4] BrookGPU `graphics.stanford.edu/projects/brookgpu`

[5] NVidia ® "The CUDA Toolkit and NVCC compiler" `www.nvidia.com/object/cuda_home.html`.

[6] F.Christoph Iselin, "The MAD Program, Physical Methods Manual", CERN 1994.

[7] R.B. Appleby, D.S. Bailey, M.D. Salt, "Beam Dynamics using the Stream Processing Code GPMAD", EUROTeV-Report-2008-022.

[8] I. Agapov et al. "The BDSIM Toolkit", EUROTeV-Report-2006-014-1.