Decreasing NAME III Solution Time Using GP-GPU

Kingsley Gale-Sides

August 2011

Contents

Figures

Acknowledgements

Kind thanks are given to Xu Guo (EPCC) and Eike Müller (Met Office) for their support and help with this project.

Abstract

The potential for decreasing the solution time for the UK Met Office NAME III [1] lagrangian particle atmospheric particle dispersion modelling code was examined. The code was ported to the EPCC Ness and Fermi0 machines and compiled with the PGI compiler. Timing benchmarks and profiling completed for a particle only run, and a cloud gamma run to examine potential areas for speed up.

A prototypical simple dispersion model was conceptually compared to the NAME III Particle benchmark. This simple model was accelerated using OpenMP, CUDA-C and CUDA FORTRAN. Timing benchmarks and profiling completed for various problem sizes from 1000, to 10,000,000 particles, 10,000,000 representing a realistic problem size for an emergency particle run [2].

The simple model was found to have a total speed up of up to ~50x for the largest problem size with the particle loop being sped up ~80-100x. The results can be extrapolated to indicate the NAME III code could be sped up by approximately 12x for this specific benchmark, or about 59x scaled to a realistic problem size.

Other areas with potential for speed up in NAME III were also evaluated.

Due to the complexity of the NAME III code although CUDA and GP-GPU acceleration can readily be applied to targeted areas of code representing specific benchmarks, it may not represent the best option for speeding up the whole of the code. The Met Office may like to consider a hybrid OpenMP and MPI approach utilising the existing OpenMP implementation.

1 Introduction

1.1 Name III

The Met Office Numerical Atmospheric Modelling Environment (NAME III) models particle dispersion in the atmosphere and deposition. It is used to create simulations in response to an emergency such as volcanic eruptions and nuclear incidents. The time to produce these simulations is therefore critical.

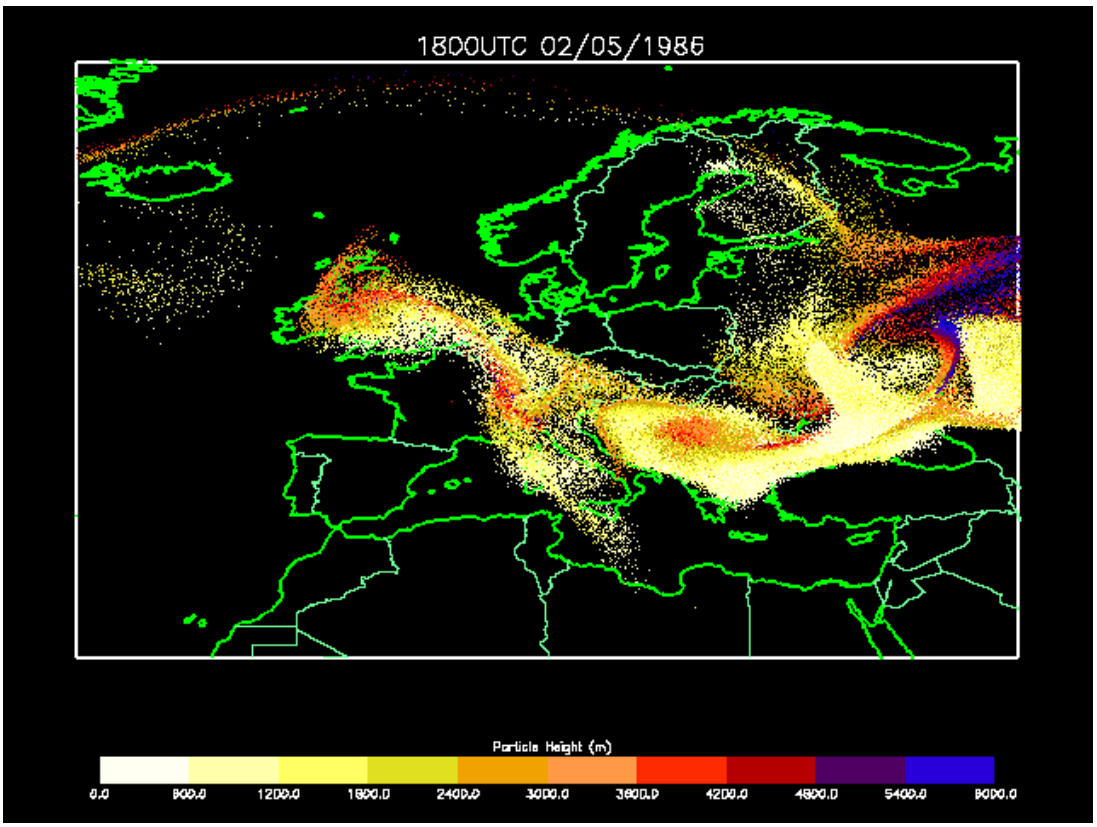


Figure 1 Name output from Chernobyl incident [3]

The Met Office Numerical NAME II code was originally developed in response to the 1986 nuclear incident at Chernobyl, to predict the path and deposition patterns of the nuclear fallout. Since then it has been expanded to deal with a wider range of situations both for disaster response and for research.

For example in disaster response the code can be used to model:

- Nuclear fallout, including radioactive decay of particles
- Chemical leaks from factories, including chemical reactions of released agents
- Volcanic eruptions with dry and wet deposition of erupted particulate

Examples of research include

- Predicting tephra dispersion patterns
- Industrial insurance and risk assessment
- Government agency readiness and training

In all of these situations time to reducing the time to solution is critical either in providing timely data in the incidence of a disaster, for ensemble simulations with different parameters to be completed for research. Decreasing the solution time enables more particles to be simulated in a given time reducing the statistical uncertainty.

1.2 The UK Met Office

The Met Office is the UK's National Weather service established in 1854. The Met Office produces not only the public weather service reports that can be seen on television and the internet, but also climate change reports, guidance for governments, and even the shipping forecast.

The Met Office plays a key role in emergency response planning and preparedness, the remit of which extends from early severe weather warnings to modelling events on a global scale such as the type of modelling completed by NAME III. [2]

As a Volvanic Ash Advisory Centre (VACC) the Met Office uses NAME III to produce forcasts of volcanic ash simulations for many organisations such as the Internationalal Civil Aviation organisation (ICAO), recent reports include the Grímsvötn Volcano in 2011, and the much publicised Eyjafjallajokull eruption in 2010. [3]

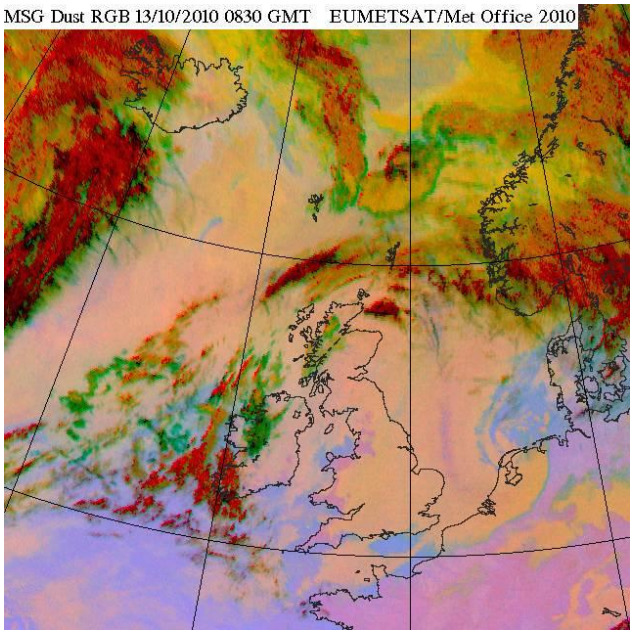


Figure 2 Met office Volcanic Ash Advisory Centre:
Dust cloud image from Eyjafjallajökull [4]

1.3 Dispersion modelling

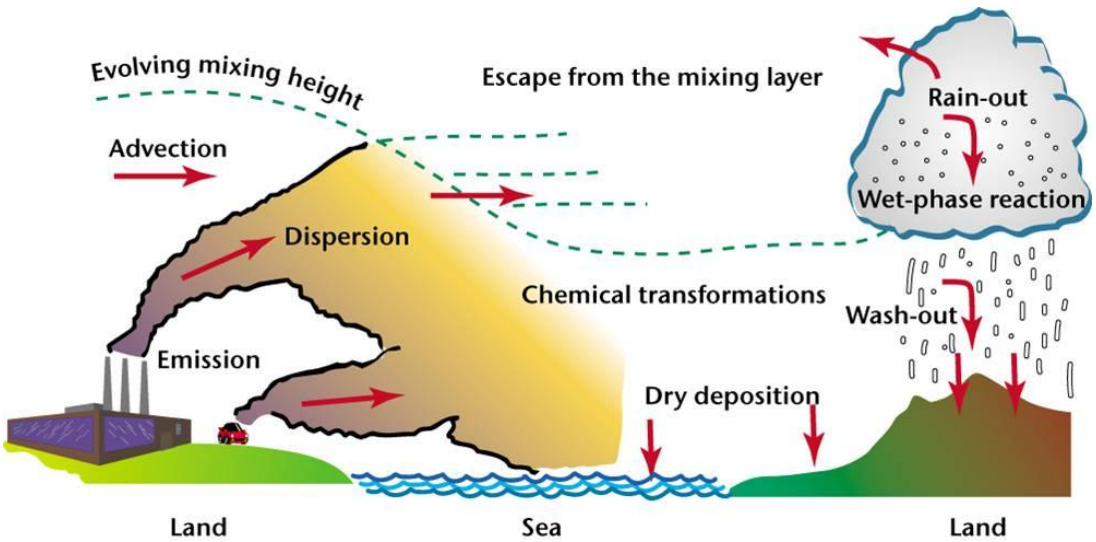


Figure 3 Processes in dispersion modelling [3]

Atmospheric dispersion modelling performed by NAME III is concerned with modelling particulate and chemical transitions in the atmosphere. NAME III utilises a Lagrangian model for the particles, in which the particles can be considered largely independent, rather than an Eulerian model in which the particles and air are treated as interacting continuums [5]. This is particularly useful when considering a

parallel implementation since as the particles do not interact they can be distributed in a task farm fashion and require no communications between updates.

NAME III modelling is very complex as a wide variety of processes is at work. Movement of the particles must be considered, with the effect wind and turbulence (important at atmospheric boundary layers) on both the small and large scale. Particulate removal must also be considered by a variety of processes such as dry deposition and wash out in rain. Chemical reactions may also change one type of particle into another. Radioactive decay is an important mechanism and is used for in simulations of nuclear accidents, as is calculating gamma ray dosages from radioactive emissions.

Much of the motion is turbulent in nature with individual particles modelled as having stochastic (Brownian) motion [6], but the overall effect of large scale phenomena, for example wind have a strong effect. To this end NAME III can utilise data produced by the Met Office Unified Model [9], importing it at various time steps to model the system's evolution over time.

Dispersion modelling is important to the Met Office for many reasons, across many industries and many length scales. These vary from modelling chemical dispersion from a factory accident in a small town, to nuclear fallout across Europe from a reactor melt down. More recently of in the press dispersion modelling has been used to model volcanic ash in the atmosphere and it's deposition. However there are wide-ranging commercial applications for, dispersion modelling; it is of vital importance in agriculture from modelling pollen and seed dispersion in studying plant population and crop potential.

1.4 The Met Office Unified Model

The Unified Model (UM) is the numerical weather prediction system developed by the met office it is used around the world. It can be used to make predictions from the global to national scale.

Based on a structured grid system it takes input from various sources (local observations, weather balloons, satellites etc.) to and produces ensemble weather predictions which are refined by Met Office forecasters.

NAME III can periodically load UM data to provide probably dispersion patterns based on the current predictions of weather. Historical weather data can also be provided in UM (or other) format to validate output given the correct weather patterns.

1.5 Previous research - accelerated lagrangian models

Lagrangian particle models are used in many areas of research and previous uses of GPUs to accelerate the code have been completed. F.Molnar et al's work [8] looked

at a similar model to the NAME III Particle loop, and emphasised the importance or special locality of memory and coalescing memory access.

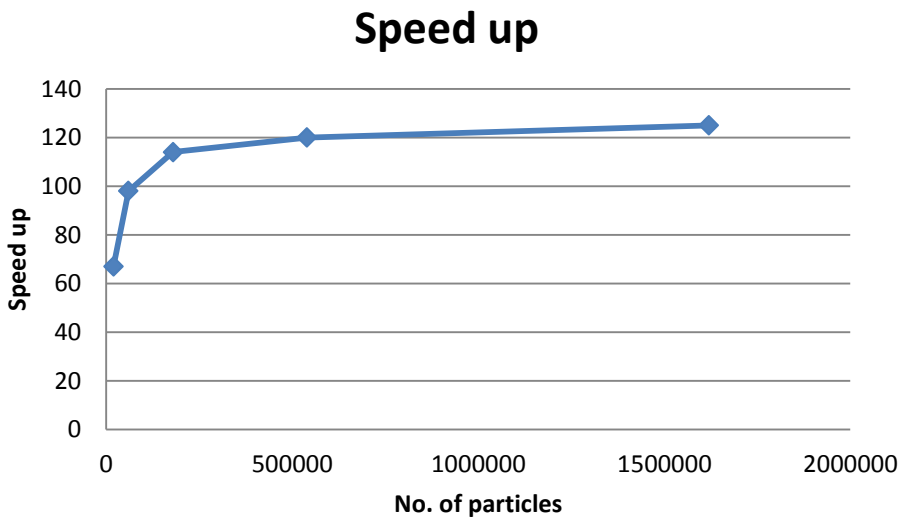


Figure 4 Speed up reproduced from F. Mol**nar et al's paper** [9]

A speed up of 80-120 was found to be possible using an NVIDIA GeForce 8800 GTX compared to a 2.33 GHz Intel Core2 Duo on C code

1.6 Motivation

Accelerating the NAME III code would enable simulations to be produced in a much shorter time for disaster response or reducing statistical error in the current runtime allocation.

By using GP-GPUs to accelerate the code it will be accessible to more organisations and institutions as this is a reasonable low cost system to deploy, especially when compared to dedicated distributed memory machines, or shared memory systems with large processor counts.

2 GP-GPU

2.1 History

Moore's law states that processors will double in complexity every 12 [9] -24 [10] months, this complexity refers to the number of transistors on a chip. Until recently this has broadly tied in with a doubling of clock speed every 18 months.

There are however problems with this approach to performance. Increased processor speed requires exponential increase in power to combat this manufacturer's shrink the die size by using smaller lithographic processes. There are

fundamental physical limitations to how small the die can be shrunk, at very small sizes quantum effects become relevant, and at very high clock speeds the speed of light will fast become a limiting factor!

This has led to very complicated chips, with deep pipelines and many instructions as manufacturers have tried to keep up with the law as failing to do so would be a disaster for sales and marketing!

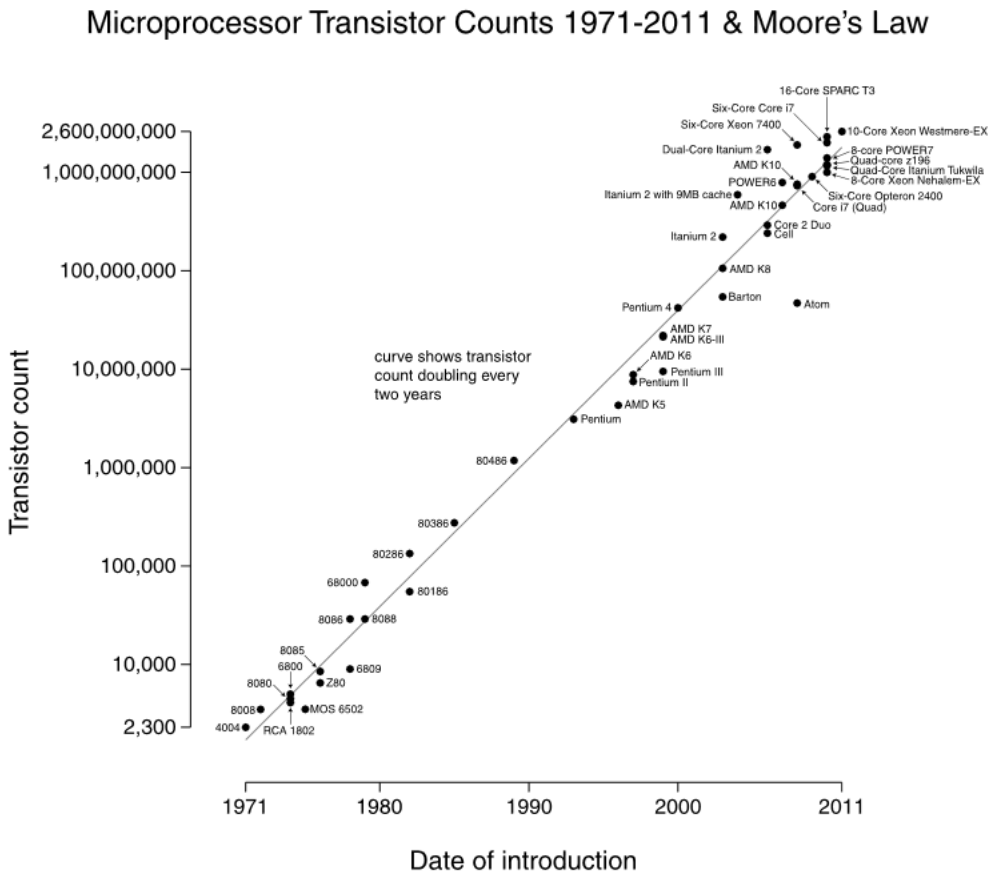


Figure 5 Moore's law for transistor counts [9]

A more recent trend has been for manufacturers to develop multi-core processors. Servers and performance systems have long had multiple sockets for individual processors as a way to cheaply increase performance. Manufacturers initially implemented multiple dies on one package each having a separate cache and route to main memory [12]; this technique was successful for up to 2-4 cores but was expensive and complex to implement and required special motherboards. The next evolution was to combine multiple cores onto one die, having a shared level 3 cache.

This has proved a very successful technique for chip manufacturers scaling to over 16 cores per die [13].

High performance computing is concerned with performing many calculations on floating point numbers for scientific simulation, in computing terms the speed at which these can be completed is referred to as FLOPs (Floating Point Operations Per Second). Hardware designers have spent many years, and billions of pounds developing systems to deliver maximum FLOPS. In the early days of high performance computing this was achieved by designing custom chips and architectures [14], but this was expensive and time consuming, modern HPCs therefore nowadays use commodity parts in novel systems. Many processors are used in parallel systems and connected by a high speed network [15]. Problems are divided up amongst these processors and executed in parallel, escaping the need for single very high performance chips.

These multipurpose chips however are still very complex, and in most cases for HPC overly so. Additional problems arise from having many cores connected to one memory bus: the total available bandwidth remains fixed, so the bandwidth per core decreases; the total memory remains fixed, so the memory per core decreases.

2.2 Accelerators

The concept of using an accelerator or coprocessor to speed up performance is a very old one. The Intel DX Math coprocessor was widely used to add floating point operations to early Intel 386 chips which only performed integer operations, in principle modern accelerators fulfil a similar purpose.

Figure 6 Intel i387 DX maths coprocessor [16]

Accelerators are more specialised processors than the general purpose ones used in host machines, as a result they often have better optimised memory systems, and offer higher performance performing a few tasks for which they're specifically designed.

Modern accelerators are far more complicated and although there are many types currently available, such a FPGAs and Cell Broadband Engine, here the most popular General Purpose – Graphics Processing Unit (GP-GPU) is considered.

2.3 Rise of the GP-GPU

As PC games became more advanced, more performance was required than host processors were able to deliver. Manufacturers such as NVIDIA, ATI and 3Dfx started to offer advanced graphics cards which dealt explicitly with the requirements of producing high speed high resolution graphics. These early cards had specific process paths associated with the exact requirement of rendering many vertices.

Realising the potential of these high performance graphics cards, programmers set about utilising them for accelerating more general programs. There were however major limitations. Programs had to be structured in such a way that the fitted exactly with the graphics model utilising vertices in the same fashion and outputting results as a pixel grid.

At the same time computer games were becoming more complex, demanding real time graphics calculations and some even utilising real time physics calculations. Manufacturers set about designing new architectures to deal with these requirements, and in doing so began to realise the potential of a more general purpose architecture.

Ageia developed the PhysX expansion card specifically for performing physics calculations, developers rapidly saw the benefits of this, as did NVIDIA, which acquired the technology and incorporated the PhysX engine into its GPUs helping create the modern GP-GPU

Figure 7 PhysX by Ageia Physics accelerator card [17]

Modern GPUs now consist of hundreds of simple cores, with access to very high bandwidth memory. Commonly referred to as 'stream processors' these cores perform the same operation in step with one another on different data.

GPUs have provided the means to maintain Moore's law in terms of FLOPS

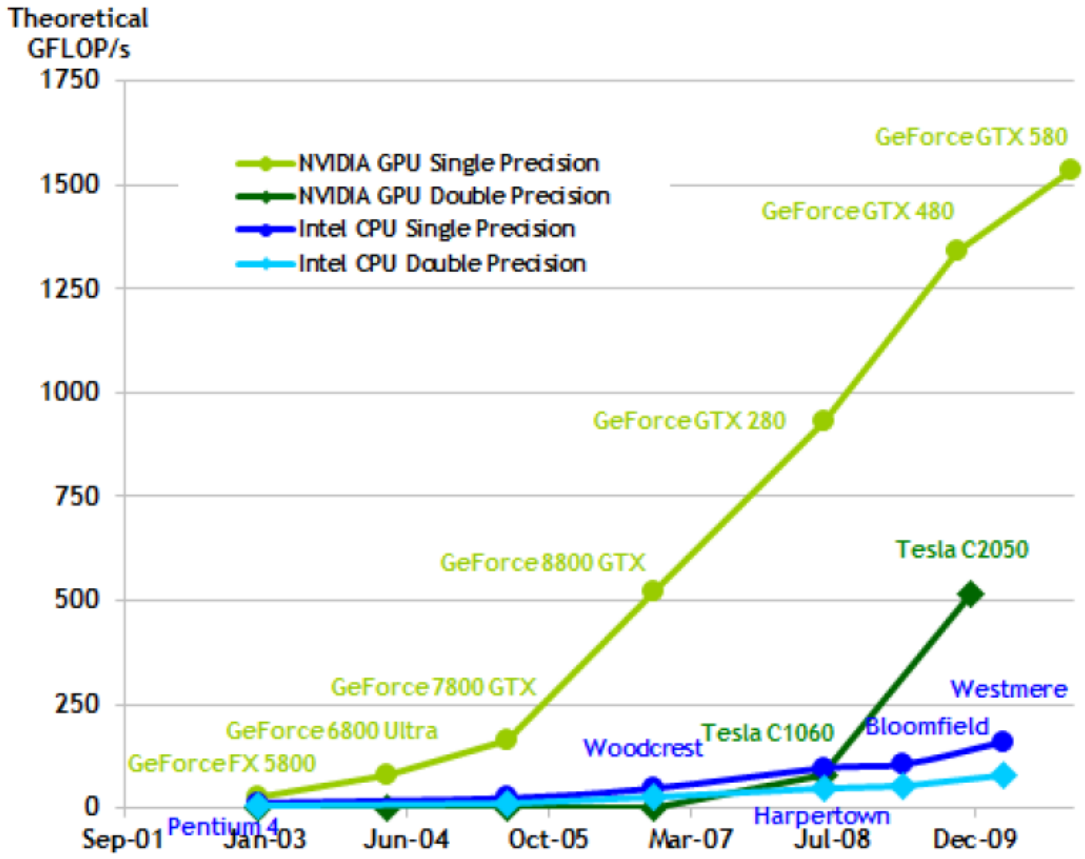


Figure 8 FLOP rate comparison of CPU and GPUs [18]

2.4 GPU Architecture

Both AMD and NVIDIA offer GP-GPU solutions, AMD (incorporating ATI) with FireStream and NVIDIA with CUDA architectures. NVIDIA is currently the more mature platform using the CUDA programming model and is now in its second generation, Fermi 20 series (following Tesla 10 series) which has been designed specifically for better performance in the GP-GPU realm. This includes the use of ECC memory (error checking and correction) and increased IEEE double precision floating point operations.

Figure 9 NVIDIA GP-GPU card [19]

GPU design architecture is much different from that of a conventional CPU

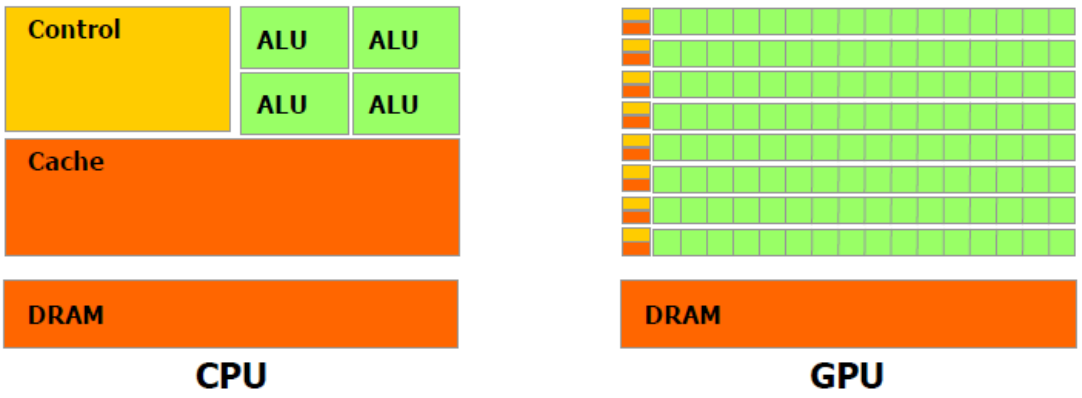


Figure 10 Comparison of CPU and GPU design [18]

CPU architectures are designed to deal with complicated flows of instructions with regular branches, where most of the time and effort is spent calculating the outcome of branches graphical interface user interaction

GPU stream processors, lockstep un-branching predictable calculation focussed. The same program is executed on each piece of data

As can be seen in the diagram above the actual structure of the GPU and CPU is physically very different The GPU has much simpler control modules and many more arithmetic units these are arranged in to stream multiprocessors, each consisting of many cores and shared memory.

2.5 Coding for GPUs

GPUs are referred to as ‘stream processors. This is because the data is not executed in serial one after the other but as a simultaneously by element in a stream.

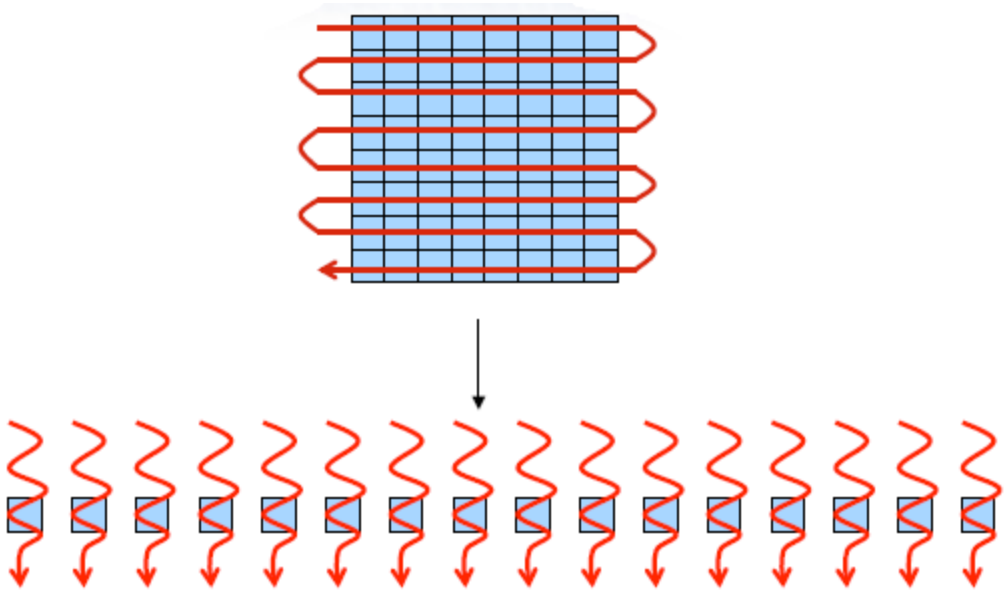


Figure 11 loop over data elements converted to stream of parallel elements [20]

This data parallel approach means the same program (kernel) is executed on each processor, and they all work in step performing the kernel on one data item at a time.

The Language used for writing these kernels is CUDA-C or CUDA-FOTRAN

The main body of the code is still executed normally on the host processor but using CUDA API calls the data is transferred to the memory on the GPU.

The CUDA kernel is then launched on the GPU using the chevron syntax **<<<...>>>** this syntax enables the layout of the threads to be mapped to the processors.

When the kernel is executing it uses this information along with its own processor id to calculate which element of data it should be working on. The arrangement of threads can be altered using this syntax which can be useful to optimise for coalesced memory accesses.

2.6 Development Strategies

Programming for GPU architectures requires much thought in advance. The most prominent considerations are threefold; primarily that a small kernel is required, since there is a limited amount of memory and restricted registers on the device. The second consideration is that of optimising memory access, both transferring to and from the device and of the device memory by the kernel. The third consideration is perhaps the most complex which is to ensure the independence of iterations within the kernel although some communication and synchronisation is possible it may impact significantly on performance.

Opposed to being a general purpose device like a system's main CPU a GPU is more specialised. This means that it implements a different instruction set when compared to the CPU. This difference is why it is not simply a case of telling the program to run on the GPU, code must be rewritten and recompiled for the GPU.

As much as this makes coding for GPUs trickier, it can have benefits in terms of speed also, for instance the GPU offers extra inbuilt functions for performing maths such as sin, cosine and many other geometry related calculations.

Although the cores of the GPU are fast and numerous the memory system is still subject to similar constraints as a conventional system. The memory offers very high bandwidth, but a latency of many [18] clock cycles. In order to best optimise for this situation memory allocation and access should be continuous and there should be many more times data than cores. This allows the processor to stream memory accesses in order to hide the latency. Without this, the start-up cost of each calculation would be prohibitively large.

These both contribute to the SIMD (Single Instruction – Multiple Data) like programming model for the GPU, which is a small kernel performing limited calculations on many data items.

The final consideration mentioned is that of memory transfer. The GPU is connected to the CPU over the PCI Express bus, which is (compared to accessing the main memory) very slow [18]. All data which is to be available to the GPU kernel must be explicitly transferred to and from the GPU. The organisation of these memory transfers can have a large impact on the performance of a code. It is also possible with some models to overlap the loading of data with the calculation.

2.7 PGI CUDA-FORTRAN

Several languages are available for developing on GP-GPUs. The most common for NVIDIA graphics cards is CUDA-C this provides an API for interfacing at a reasonably low level with the hardware.

Many scientific codes are however written and maintained in various flavours of FORTRAN. Until recently this meant that if a code was to be ported the kernel would be written in CUDA-C and calls to that kernel would be added to the FORTRAN program. This has many disadvantages, from the learning curve for developers not experienced with writing C code, to more complicated build and debugging procedures.

Enter Portland Group Inc (PGI) and the PGI CUDA-FORTRAN compiler. The objective of the new compiler was to provide FORTRAN programmers with the same access to the CUDA API but from a native interface. The CUDA-FORTRAN compiler is available for NVIDIA cards with compute capability 1.3 or higher [20]

However unlike CUDA-C which is free and can be used with the free GNU compiler, the PGI compiler suite is a commercial product for both research and industry and therefor commands a premium price.

Compiling with CUDA is as simple as renaming the source files **`*.CUF`** or using the **`-Mcuda`** compiler flag, other CUDA specific compiler flags are also available as are existing serial code optimisation flags.

2.8 PGI Accelerator Directives

Another tool available within the PGI directive suite is the Accelerator Directives. These provide an abstracted implicit high level interface to GPU parallelisation. Although very limited in their application directives can provide a quick OpenMP style approach to Acceleration.

```
!$acc region
do i = 1,n
   r(i) = a(i) * 2.0
end do
!$acc end region
```

As can be seen from the example above this is very similar in concept to OpenMP

```
!$OMP PARALELL DO
do i = 1,n
   r(i) = a(i) * 2.0
end do
!$OMP END PARALELL DO
```

The development approach for using compiler directives is highly restricted and not suitable for all cases, It's main benefit is enabling non expert programmers to utilise GPU acceleration in simple loops in scientific code [21] An interesting benefit is that the two can be used in different places within the same program.

2.9 Speedup

When considering parallelisation of the code it's important to quantify what the improvement in runtime speed is, and how this relates to the theoretical maximum speedup available. A useful rule guide for this is Amdahl's law [21] which is stated as :

$$S(n) \approx \frac{Time(n\ cores)}{Time(1\ core)} = \frac{1}{1 - p + \frac{p}{n}}$$

Where: n is the number of cores, S(n) is the speedup on n cores compared to the time taken on one core, and p is the proportion of the code which is parallelised.

It can be seen for instance that if 50% of the code were parallelised the maximum speedup with any number of processors would be

$$S(max) \approx \lim_{n\to\infty}\left(\frac{1}{1 - 0.5 + \frac{0.5}{n}}\right) = 2$$

An expansion to this formula can be considered which is more useful. In dispersion modelling the amount of time spent in these parallelisable regions is linear with the number of particles. In this way increasing the problem size will enable a larger speed up (Gustafson's Law) [22], if we make p (the proportion of the code running in parallel, a linear function of the problem size z

$$S(n) \approx \frac{1}{1 - p(z) + \frac{p(z)}{n}}$$

If the problem size were doubled for the example above, the result is now:

$$S(n) \approx \lim_{n\to\infty}\left(\frac{1}{1 - \frac{2}{3} + \frac{2}{3n}}\right) = 3$$

And for ten times the problem size we would have $S(max) = 11$, which is perhaps more useful than the pessimism implied by Amdahl's law.

2.10 Timing Methodology

The Ness machine has a queuing system which is used to submit jobs to back end machines which are not available interactively to help provide accurate timings. This is used for timing serial and OpenMP code. PGI CUDA-FORTRAN is not installed on Ness, although it now supports compute capability 1.3. For scientific use the

ECC memory of NVIDIA series 20 GPUs is essential, so timings for GPU accelerated code are performed on Fermi0 (see note previously also).

All runs are completed 3 times and the fastest wall time (total run time as reported by **`time xxx.x`**) run selected for analysis, if the times are highly inconsistent the process is repeated.

2.11 Profiling

Timing code execution can be done in many ways. In its most simplest this consists or running the code using time

```
time executable.x
```

This returns for example :

```
real 0m15.458s
user 1m1.442s
sys  0m9.27s
```

Of interest here is **`real`**, which indicates the total wall time between invocation and completion of the executable and **`user`**, which indicates the total amount of CPU time used, in this example it is more than **`real`**, since four processors were used.

This information although useful only gives the time taken for the complete execution. It is used primarily when examining the speedup of a whole code. It does not provide any information about where in the code time is being spent.

This is where profiling tools such as gprof are used. Gprof is short for GNU profiling tool and is a call graph profiler [23]. Gprof is used to produce a file which indicates which functions were called in a tree like fashion, the number of times they were called, and the amount of time spent in within each function.

Gprof is invoked by adding the **`-g`** flag to the compilation. It must be added to all files compiled and linked in the final executable.

The code is then run as normal, with the required input files. An output file gmon.out is created when the execution completes. The gmon.out file contains the information gprof can then use to create a readable profile file. To generate the file use:

```
gprof executablename.x > outputfile.txt
```

It's worth noting that the statistical nature of profiling means that routines which are very short should be interpreted with some caution, and profiling files used only as a guide. For these reasons manual timers are added to the NAME III OpenMP

implementation and were also added to the Simple Dispersion model code during parallelisation.

2.12 Development Machines Hardware

Two development machines are available within EPCC Ness, and Fermi0. Both are shared memory machines based on Intel Xeon architecture with NVIDIA CUDA cards installed.

	Ness	*Fermi0*
Processor	Intel Xeon E5504	Intel Xeon X5650
Clock speed	2.0 GHz	2.67 GHz
Number of cores	4	6
Number of processors	4	4
Total logical cores	16	24
Accelerator	NVIDIA M1060	NVIDIA C2050
Compute capability	1.3	2.0
PGI Compiler Version	10.0-0	11.2-1

It is important to note that although supported by the GPU on ness CUDA-FORTRAN is not implemented in PGI Compiler Version 10.0-0 [24] *so fermi0 must be used for all GPU development*

3 NAME III

3.1 Introduction

The NAME III is a code some 110,000 lines long, written in FORTRAN 90 and has been produced and provided by the met office. NAME III is available for research purposes under licence [25]. For this research the code has been supplied under a Non-Commercial Research licence[1] (Nov 2010- Nov 2011).

The NAME III model is very complex, which means the code is also very complicated, much more so than previous dispersion model codes accelerated with GP-GPUs. [26]

The code consists of a main control programme which accepts many parameters to determine the type of simulation being ran.

The modelling approach is broadly broken down into 3 main sections: particle propagation, physical process and atmospheric chemistry. Many input parameters

[1] See appendix

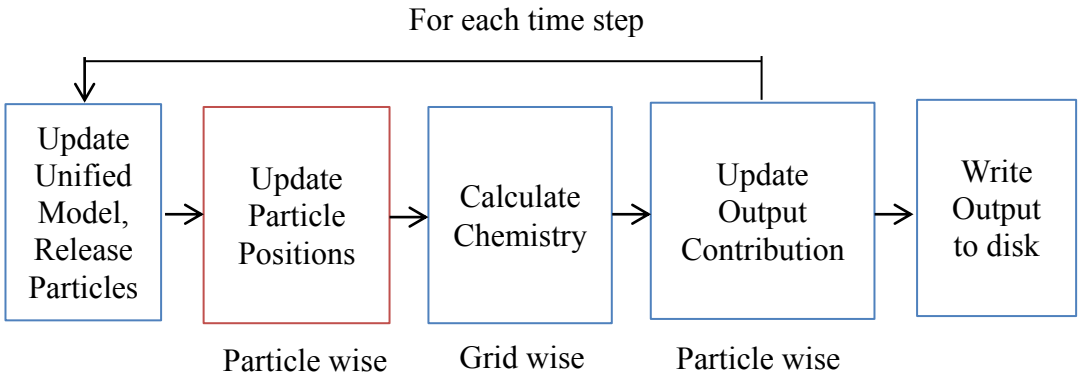
are required for each simulation such as the chemistry and physical properties of the particles being modelled.

The modular approach to the code enables features such as chemistry to be switched off to complete just a particle run for example.

Depending on the length and timescales particles may either be modelled as particles or as 'puffs' the movement of both is considered to be independent. Only the particles model has been parallelised so far, and it is exclusively this type of model that will be focused on. Each particle is representative of a very large number of particles, since simulating a real life number of particles would be incredibly time consuming.

The code can periodically load new data from the Unified Model to update wind and weather patterns, and periodically writes the calculated particle densities to disk.

On the most basic level the code works in a reasonable well defined block structure [9].



The output interval may be specified so that I/O is not performed on every time step, particle positions are updated many times each time step.

Since many particles (several million) are considered in each time step the runtime is very long, typically an emergency volcanic simulation takes about 1.5 hours[2] to execute at present by the Met Office. As the model is a statistical one the quality of the output increases with the number of particles, so for production runs many more particles are considered.

For the case or emergency response only the particle dispersion elements of the code are used, since the whole code is too time consuming. Many millions of particles are considered in the run, and runtime is approximately linear with number of particles.

[2] On Intel Xeon X5680 3.33GHz server

3.2 NAME III OpenMP

Some areas of the NAME III code have already been parallelised using OpenMP by Eike Mueller at the Met Office [27] and were implemented from version 6.0

The areas parallelised in this version are.

Particle loop

- Responsible for the evolution of the position particles over time.
- Parallelised by particle number, each particle is independent.

Particle update loop

- Calculates the concentration field from the particle's position for each particle at a given time step if it's still in motion.
- Calculated deposition field contribution for each deposited particle
- Parallelised by particle number, each particle is independent.
- Output field data is shared amongst the particles, so communication is required in the reduction calculation phase to prevent field points being summed incorrectly.

Chemistry Loop

- Operating over a 3d grid
    - Converts number of particles in grid box to concentration of chemical species
    - Updates concentrations with reaction rates for each species
    - Number of particles in each grid square updated according to concentration.
- Parallelised by grid square, grid squares are independent.

Paralell I/O has also been implemented for output requirements and reading MetData in the form or a dedicated I/O thread which operates reads while worker threads propagate particles.

The Particle Loop was found to parallelise very well :

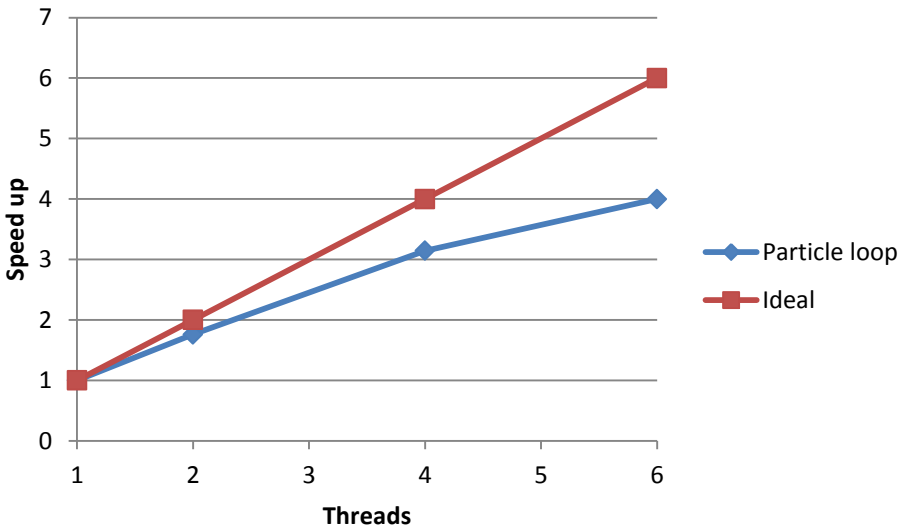


Figure 12 OpenMP scaling results for particle loop in complex model [28]

The Chemistry loop exhibited poor scaling on large thread counts, it was hypothesised that this might be due to memory access patterns which could be improved by reordering loops, and still constitutes an active area of research.

3.3 Porting NAME III to PGI Compilers

NAME III was provided by the Met Office on CD-ROM under licence[3] Executing the contracts for the code's release took much longer than anticipated this was not factored into the original risk analysis, and resulted in delays of several weeks.

The code is in FORTRAN 90 and is configured to use the Intel FORTRAN compiler. The development machines available with the EPCC do not implement the Intel compiler suite, so the code needed to be ported to use the PGI compilers.

Primarily this consisted of editing the makefile to use the alternative compiler and alter the flags appropriately.

The initial compiling flags used were:

```
Ifort -Wp, -macro=no =C -w -extended_source -nbs -Vaxlib
-auto -heap-arrays -sox
```

[3] See appendix

For the PGI compiler these were changed to

```
pgf90 -03 -MBackslash
```

Here the –Mbackslash tag is added to prevent backslashes being treated as escape characters, this is the default behaviour in the Intel FORTRAN compiler.

For analysing the code the GNU profiling tool gprof was used. This requires the addition of the `-g` compiler flag

As the NAME code is very complex and the errors reported by the compiler are of little use, much assistance from the Met Office was required to successfully port the code.

In summary the changes required were:

- `NWPFlow.F90`. PGI compiler doesn't support passing variables to functions in the form Function(/String, ListOfStrings/) this was altered to Function(item='value',..) instead
- `Makefile` IntelPortlandCompiler pre-processor variable added for OpenMP compatibility
- `All files` _Std and P64 removed from real constants throughout
- `System.F90` command line arguments read by FORTRAN 2003 intrinsics
- `Timers.F90` UseTimers variable name conflicted with one in pre-processor, was changed due to different handling in PGI as no `-Wp -macro=no` alternative to the Intel compiler flags was available

The code however failed to compile due to an error in the dependencies. This was tracked garbage binary data having been written to one of the object files.

Once the code was successfully compile a test benchmark was ran, this resulted in a segmentation error which proved hard to correct. After much investigation this too was resolved by spuriously declaring a variable in one of the subroutines.

It is not known why this data was written to the files, nor why it persisted in a make clean and a re make, nor why the declaration of this variable enabled the code to run.

The only solution was to create a new directory and load the whole NAME code from scratch and recompile. This suggests there could be a disk fault or other hardware related problem.

The porting took several additional weeks to complete than was expected in the original plan, and beyond even the worst case in the risk assessment. This resulted in a slight re-scoping of what would be possible within the dissertation.

3.4 Benchmarks

In order to profile the code and view performance in such a way as for it to be informative benchmark input files were provided. These consisted of data which targeted specific parts of the name code, and spent little time elsewhere.

3.4.1 Benchmakr_Particle.txt

This benchmark is designed to primarily utilise the lagrangian particle part of the NAME III code. It utilises a constant field for met data (requiring negligible I/O time), spends little time in I/O overall and nearly all the runtime within the particle loop.

The particle benchmark runs for 100,000 particles, comparing to several million for a real world disaster response.

The benchmark was compiled for serial execution and ran on NESS. Total runtime was 100.7, of which 73.0s were concerned with the benchmark seconds.

The call graph is summarised

Routine	*Calls*	*Time*	*Children time*	*Percentage*
Main	1	0.1	72.9	100
Runtosuspendorendofcase	1	3.65	65.8	95
Casemodule_runtorestrart dumporsuspendorendofcase	1	0	65.75	90
Casemodule_runtosynctime ormetflowupdateorendofcase	48	0	63.94	87
Casemodule_loopparticlepuffs andtimesteps	48	0.08	55.83	76
Casemodule_loopparticlepuffs andtimestepsub	2.2e8	2.81	52.68	72
Casemodule_oneparticletimestep	2.2e8	2.81	42.64	58
Other calls				

As can be seen from the call graph the main kernel would be Casemodule_loopparticlepuffsandtimestepssub, which is called 2.2x10^8 times performing the calculations within this subroutine consumes 72% of the runtime using Amdahl's law leads to a maximal speedup of 4.55x

However if the problem size is scaled to that of a realistic solution 20,000,000 particles, this would lead to a theoretical maximum speed up of 51x

3.4.2 Benchmark_CloudGamma.txt

This benchmark is also stripped down in a similar fashion to the particle benchmark, with the addition of the particles now being radioactive emitting. gamma radiation.

In each time step the total space is broken down into grid boxes, the distance to each particle from this grid box is then calculated and its contribution to the radioactive dose received in that grid box is calculated

Here the total runtime was 76.01s with17.64 concerned with code execution. Gprof produces very muddled profiles of this code with many <spontaneous> function calls Again the simplified call graph there is m:

Routine	*Calls*	*Time*	*Children time*	*Percentage*
Main	1	0.0	17.64	100
Runtosuspendorendofcase	1	0.3	17.28	97
Casemodule_runtorestartdumporendofcase	1	0	17.28	97
Casemodule_runtosynctimeormetflowupdateorendofcase	1440	0.0	17.28	97
Casemodule_loopparticlepuffsandtimesteps	1440	0.1	15.89	91
Casemodule_loopparticlepuffsandtimestepssub	1.4e5	0.01	15.89	91
Casemodule_calculateparticleresults	31884	0.01	15.25	87
Particlemodule_particlecloudgamma	31884	3.72	11.53	86
Coordinatesystemmodule_calculatedistancebetweentwopoints	1.6e8	3.89	4.77	49

To examine the data and the selection it's important to consider the way the code is implemented

```
For each particle
  ParticleCloudGamma(
  Loop over x coordinate
    Loop over y coordinate
      Loop over z coordinate
       Calculate distance between particles and grid box
              (CalcDistanceBetweenTwoPoints())
        Calculate gamma dose
              (Particle Fluence())
        Update output fields
      End Do
```

```
    End Do
  End do)
End
```

It's now clear that the kernel should be Selected as ParticleCoudGamma() which would result in a potential speed up of 7.14x

The CalculateDistanceBetweenTwoPoints could be implemented efficiently using optimised CUDA library calls.

The problem with selecting this as the kernel however is it would require output from the GPU every coordinate step, so transfers and calculation would have to be done in parallel to gain any speed up.

The second issue to be considered here, is that all new 3d grid data is transferred to the GPU for each particle, so in reality it may well be worthwhile moving this level of loop into the CUD kernel too, to avoid excessive time spent in data transfer.

3.5 Acceleration target selection

Summarising the results from the benchmarks the primary routines to be accelerated would be for both cases casemodule_loopparticlepuffsandtimestepssub which contains within it routines for moving the particle according to the lagrangian model, and calculating the radioactive decay over a 3d grid.

This outer loop however contains many other elements of calculation so would make the kernel very, and potentially too large.

4 Simple Dispersion Model

4.1 Introduction

The simple dispersion model is a prototype for the particle dispersion implementation in NAME. Since the main NAME code is highly complex this model was supplied by the Met Office in order to test acceleration techniques, which could be related to the NAME model.

The simple dispersion model was supplied in both C and FORTRAN variants.

4.2 About the code

The model simple stochastically propagates particles for a given number of time steps in a constant wind field. At given intervals the particles are converted to a concentration field on a 2 dimensional grid and then output to file.

The detailed pseudo code is:

```
Instantiate variables
Create array of random seed
Create array for x
Create array for y
Create array for grid(x,y)
For each output
     For each particle - integrate position:
          Calculate generic constants
          For each timestep
               Calculate random numbers from seed
               Increment particle position
          Update seed value to array
          End
     Convert particles to density
     Save density to file
End
```

By keeping the seed array constant, this enables the outputs to be compared whatever the acceleration method. The sequence of random numbers generated will always be the same, so that excluding rounding and floating point calculation differences the results will be identical regardless of the parallelisation method.

The random numbers are calculated from the seed by using a box-muller transformation [28], this calculates 2 independent, normally distributed random numbers from the seed thus

$$Z_1 = \sqrt{-2lnU_1}\cos(2\pi U_2)$$

$$Z_2 = \sqrt{-2lnU_1}\sin(2\pi U_2)$$

Where S is the seed value and Z_1 Z_2 are normally distributed random variables.

In the simple code Z_1 Z_2 correspond to x and y, U_1 U_2 corresponded to r, -r

The output is written to *.dat files as such

```
# Particle concentration

# Grid definition
  x_centre =         0.0000000000
  y_centre =         0.0000000000
  nx       =                  160
  ny       =                   80
  dx       =         0.2000000030
  dy       =         0.2000000030
```

```
# Data
  data[     79,     39] =          0.0024999999
  data[     79,     41] =          0.0012500000
  data[     80,     37] =          0.0049999999
  data[     80,     38] =          0.0112499986
  data[     80,     39] =          0.0287500042
  data[     80,     40] =          0.0199999996
  data[     80,     41] =          0.0112499986
  data[     81,     36] =          0.0024999999
  data[     81,     37] =          0.0275000036
```

Using the python script supplied by the met office these output density files can be converted to graphics

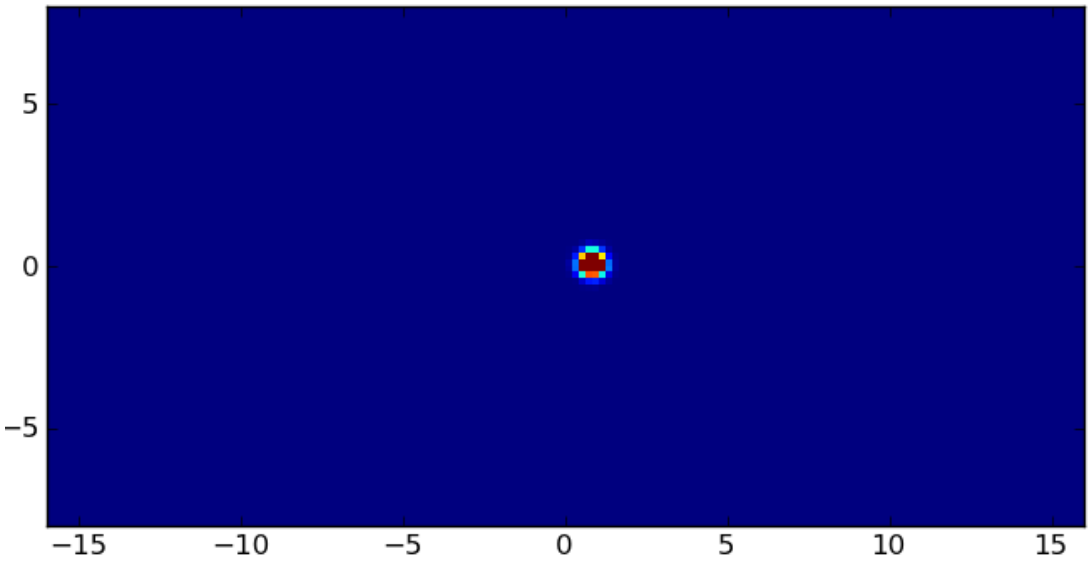


Figure 13 Particle density at first output step

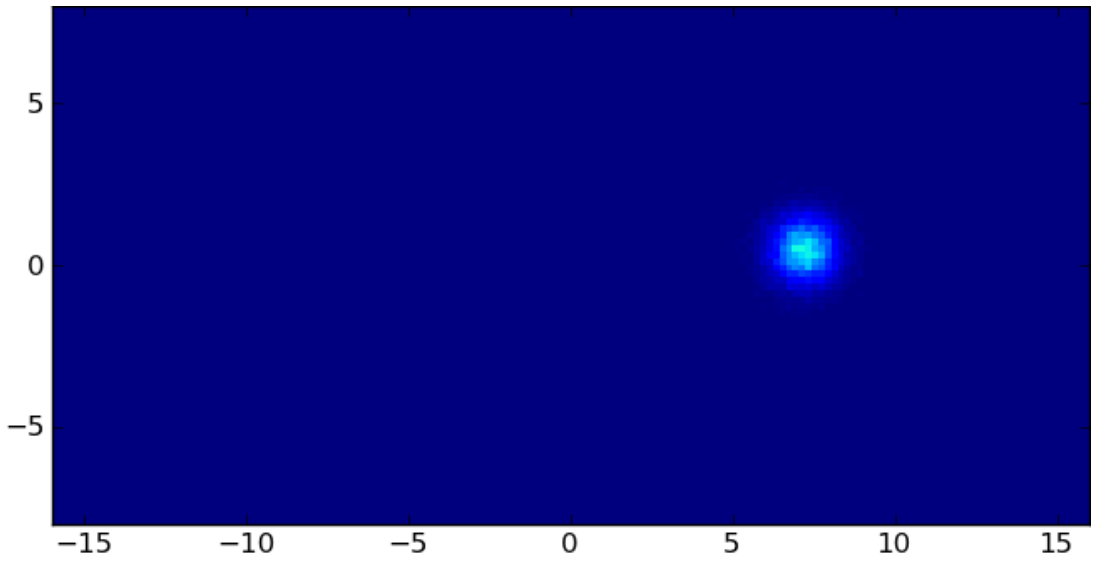


Figure 14 Particle density at final output step

Here the graphics show a wind field propagating in the positive x direction.

Initially the particles released from a point source are densely packed but over successive time steps the diffuse due to the random turbulent motion.

The uniform wind field means that the distribution maintains the same overall shape as it propagates.

4.3 Correctness

Conventionally when improving the speed of code it is expected that the output would remain the same. This however is not always possible since floating point arithmetic on computers is not commutative or associative, i.e..

$$a + (b + c) \neq (a + b) + c$$

$$a(b.c) \neq (a.b)c$$

For this reason a definition of 'correct enough' is required, for the simple dispersion model this was decided to be the least significant figure. This concept is easier to accept in the stochastic model considered here, since the movements are random already.

Each particle within the loop uses its own random number generator based on the pre-set seed array, this is required so that the same sequence of random numbers is generated if running in serial or parallel, so that the results are comparable.

For a production application where this consistency checking may not be necessary the GPU can generate random numbers with various distributions very quickly and this could be utilised as an alternative.

I wrote a code which in sequence loaded the output files from the original serial version of the code and the parallelised/accelerated code and compared the densities. If they were outside of this specified error then a report was produced detailing the errors.

For all the runs, even using single precision the error was below this specified threshold. The error checking code can be found in the appendix.

4.4 Profiling

The simple dispersion model was run with gprof enabled and a profile produced. This showed that 99.12% of the runtime was spent within the particle loop performing the integrate function, which is called for each particle.

```
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 99.12      0.92     0.92   100000     0.01     0.01  integrate
  1.08      0.93     0.01       10     1.00     1.00
                                                     particles2concentration
  0.00      0.93     0.00       10     0.00     0.00  int2char
  0.00      0.93     0.00       10     0.00     0.00
                                                     saveconcentration
```

It is therefore this function which would form the kernel for GPU acceleration and corresponds to Casemodule_loopparticlepuffsandtimestepssub in the NAME code. It must be noted that this is only true for the simple case of the Particle benchmark, since in this benchmark other processes performed in this loop ie, wet and dry deposition, chemistry and radioactivity calculation are not implemented.

4.5 Implementing OpenMP

As the particle iterations within the loop are independent parallelising the FORTRAN code with OpenMP is straightforward. OpenMP parallel do directives are added around the particle do loop. The default distribution of the loop data is utilised as a simple task farm pattern is acceptable. No additional barriers are needed as they are implied with the parallel do directives.

```
call cpu_time(t1)
! Loop over particles and update position
!$OMP PARALLEL DO default(none),                          &
     private(j),                                          &
          shared(ThreadCount,k,n,n_frames,U_x,U_y,        &
          x,y,seed,n_particles)
     Do j = 1, n_particles
          Call Integrate(K,dt,n/n_frames,                 &
               U_x,U_y,j,x(j),y(j),seed(j),errorflag)
     End Do
call cpu_time(t2)
looptime = looptime+t2-t1
```

All variables are declared as shared, as they are either global constants or arrays, except for j which is the loop variable.

This code snippet also shows that timing code has been introduced to aggregate the time spent within the loop.

The completed Open MP code was run for a variety of particle numbers: 10^3, 10^4, 10^5, 10^6, 10^7 to simulate a variety of problem sizes, these runs were completed on 1, 2, 4. 8 and 16 threads, with 16 being the maximum availability

Table 1 Total OpenMP run times for various problem sizes and thread numbers

	P= 1	*2*	*4*	*8*
10^3	0.064	0.168	0.139	0.152
10^4	0.061	0.308	0.157	0.086
10^5	5.629	2.880	1.645	0.936
10^6	56.820	31.570	14.910	8.060
10^7	547.700	316.789	154.800	80.245

This data can be plotted as a graph of speedup when compared to the single processor code examining strong scaling.

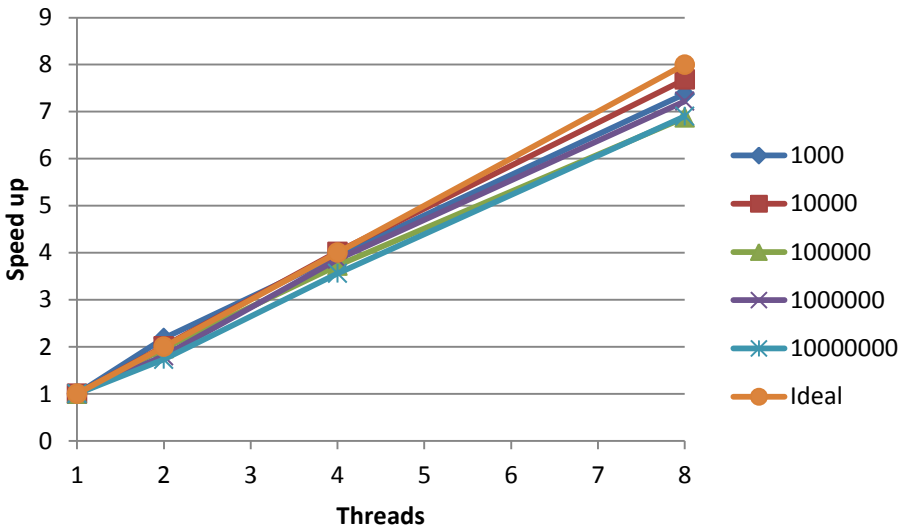


Figure 15 Graph of OpenMP speed up of particle loop for various thread numbers and problem sizes

When considering the speedup within the particle loop, this is virtually linear with the number of processors.

However when the code is looked at as a whole there is a tailing off in the speedup, this is due to the increased impact of the serial part of the code with increasing problem size. Both calculating the density and I/O operations are increased

As is noted for very small problem sizes (less than 10,000 particles) the code performs slower on higher thread counts, as operating the OpenMP costs more time than performing the code.

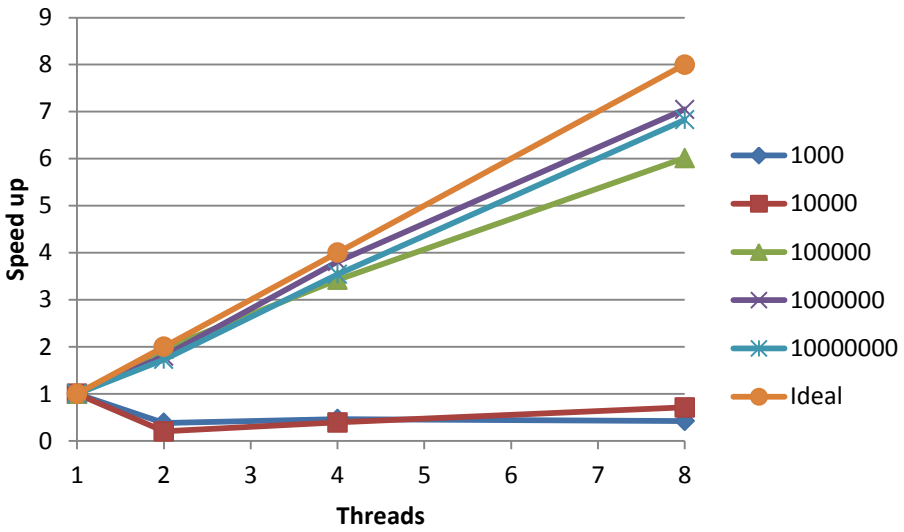


Figure 16 Graph of OpenMP speed up for all code various thread numbers and problem sizes

By keeping the problem size per thread fixed strong scaling can be investigated. Here 100,000 particles per processor are examined.

Table 2 Runtime for various thread numbers with 100,000 particles per thread.

Threads	*Total (s)*	*Loop(s)*
1	6.06	6.042
2	6.071	6.044
4	6.315	6.267
8	6.357	6.267

By plotting a graph of this strong scaling it can be seen that the code scales excellently

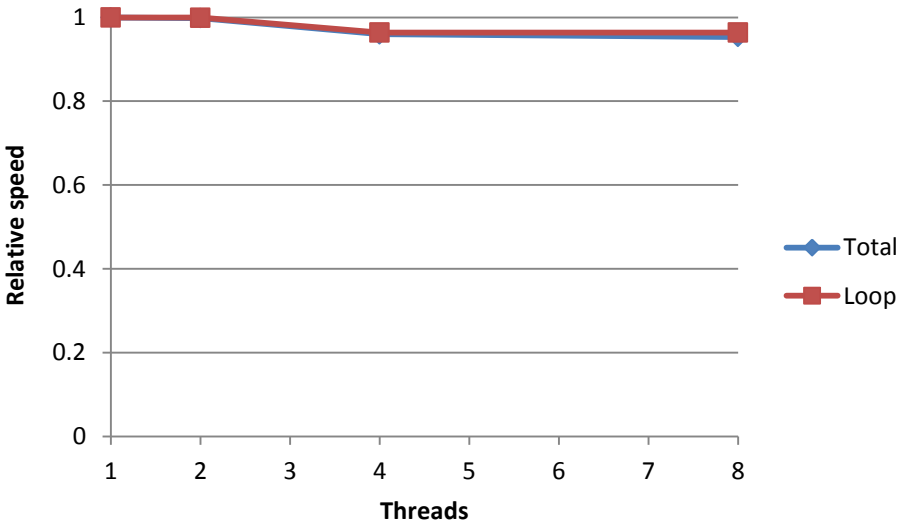


Figure 17 Strong scaling with 100,000 particles per thread

For comparison the NAME code was executed on the same processor counts with the same timing structure around the particle loop this resulted in a speedup as shown

Table 3 NAME OpenMP Particle Loop Benchmark for various thread counts

	P= 1	*2*	*4*	*8*
NAME	135.34	68.73	35.41	18.58

Which again plotted reveals a similar story scaling almost perfectly with the number of threads

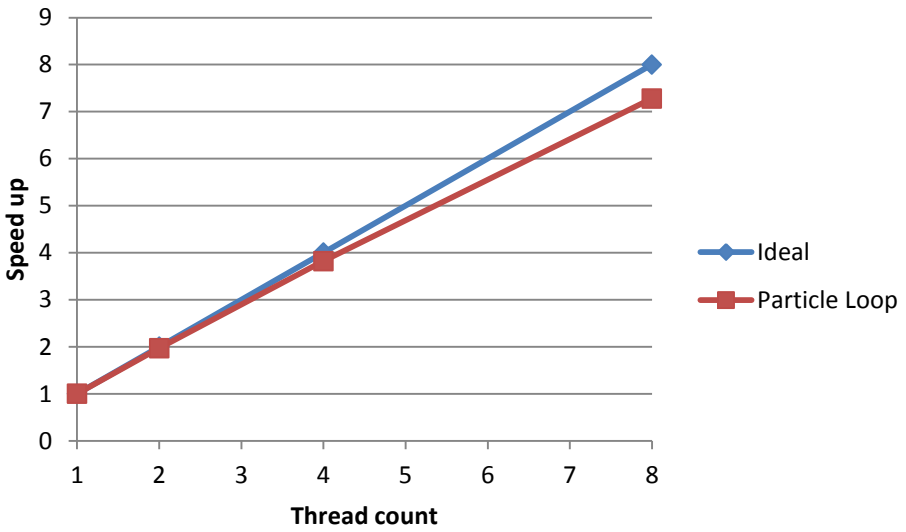


Figure 18 Speedup of Particle loop in Particle Benchmark with OpenMP

4.6 CUDA-C Simple Model Implementation

Having established the appropriateness of comparing the NAME particle loop and that of the simple dispersion code acceleration of the simple code was considered using CUDA. First the C version of the code was considered.

4.6.1 Transferring the arrays to the GPU

CUDA provides very similar calls as in conventional C.

cudaMalloc is used to create arrays on the device:

```
cudaMalloc((void **) &d_x, size);
```

Then data is transferred using the cudaMemcpy function

```
cudaMemcpy(d_x, x, size, cudaMemcpyHostToDevice);
```

with cudaMemcpyDeviceToHost available also

4.6.2 Altering the procedure call for Integrate to call the GPU kernel

In order to call a GPU kernel the chevron syntax is used to specify the way the cores calculations should be distributed amongst the cores. The call itself is altered so that rather than passing the item to the function the pointer to the array on the device is passed ie rather than **`x(i), d_x`** is passed. This enables each core on the GPU to

process one element from the array at a time in parallel with the other cores, the element to be processed by each core is calculated using nBlocks and blockSize.

```
cudaIntegrate<<<nBlocks,blockSize>>>
     (K,dt,novernframes,U_x,U_y,d_x,d_y,d_seed);
```

4.6.3 Converting the procedure to a GPU kernel and assigning it to the GPU

The **__global__** identifier is used for procedures to specify that they should be executed on the GPU and the variables required by the procedure to are altered in line with that for calling. What was not previously the particle index (from the loop index) is now calculated based on the thread layout using

```
unsigned int idx = blockIdx.x*blockDim.x+threadIdx.x;
```

4.6.4 Adding appropriate timing code

Timing code was added to sum the time spent :

- In total
- In the particle loop
- Transferring data to and from the GPU

This was done in the conventional C fashion of making calls to **clock()** however adding the timing raised an important consideration.

CUDA procedures return control to the calling program as soon as the call is made, they do not wait for the CUDA procedure to complete. It is imperative therefore that before a timing call is made the threads are synchronised. This is achieved using

```
cudaThreadSynchronize();
```

This synchronisation doesn't affect the overall runtime since the memory copy would force a synchronisation implicitly, it merely prevents the timer being executed until all threads on the GPU are complete.

4.7 CUDA-C Results

Unexpectedly the CUDA-C code takes a very long time to start up, this causes problems for the overlal speedup figures with look very poor in comparison to theose for the particle loop.

Table 4 Speed up figures for CUDA-C

	CUDA-C Total	*CUDA-C Loop*
1000	0.012	
10000	0.012	
100000	0.588	78.071
1000000	8.892	100.857
10000000	32.447	95.907

The time spend in the loops for 100 and 10000 particles was too small to report.

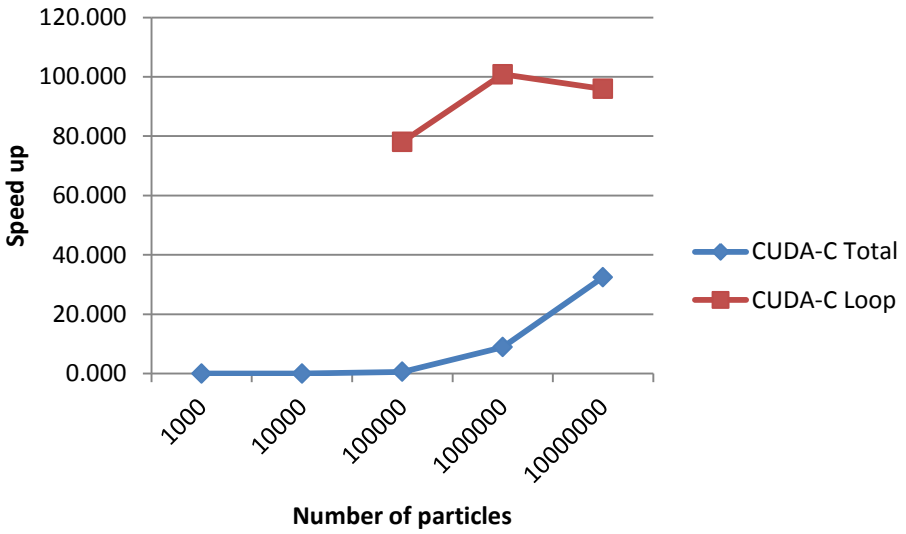


Figure 19 Speedup for CUDA-C Code for various problem sizes

4.8 CUDA-FORTRAN Simple Model Implementation

The FORTRAN version of the code was also accelerated using CUDA. This was more complex as the PGI FORTRAN compiler is relatively new and there is very little support available, compiler error messages are especially cryptic.

FORTRAN does however provide a very simple to use and logical expansion to deal with CUDA.

There are 2 options for compilation, the first of which is to simply rename the file to `*.CUF`. The second method is to use the `-Mcuda` compiler flag.

A very important and scarcely documented requirement is that the CUDA kernel *must* be in a separate module to the main code and include the **use cudafor** command.

The same steps for porting code using to the GPU are followed as previously:

4.8.1 Transferring the arrays to the GPU

CUDA FORTRAN provides a very simple method for allocating device memory and transferring data, the device keyword is simply added to the definition, then the array can be treated exactly the same as if it were local to the host

```
real, device, allocatable, dimension( : )  :: x_device
allocate(x_device(n_particles))
x_device = x
```

transferring data back from the device is as simple as **x=x_device**

4.8.2 Altering the procedure call for Integrate to call the GPU kernel

In exactly the same way as CUDA-C the chevron syntax is used to specify the way the cores calculations should be distributed amongst the cores. The call itself is again altered so that rather than passing the item to the function the array is passed as in a conventional procedure call. The array is already located on the GPU after the transfer, using the chevron syntax enables each CUDA core to calculate which array element it should process, all cores do this calculation in parallel and execute the Integrate function on their respective array elements in parallel, rather than one at a time in a loop over the array elements in the serial code.

```
Call Integrate<<<nBlocks,BlockSize>>>                    &
(K,dt,n/n_frames,U_x,U_y,x_device,y_device,seed_device)
```

4.8.3 Converting the procedure to a GPU kernel and assigning it to the GPU

In FORTRAN the **__global__** identifier is replaced with the global attribute to procedures to specify that they should be executed on the GPU. The conversion of the procedure follows that of CUDA-C but it is important to remember to convert from zero to 1 based arrays ie. The particle index is calculated as

```
idx=(blockIdx%x-1)*blockDim%x+threadIdx%x
```

4.8.4 Adding appropriate timing code

The same timing codes were added as in the C variant this time using **call cpu_time()**

Once again the CUDA threads must be synchronised to ensure the correct timing in FORTRAN this is done with

```
cudaError=cudaThreadSynchronize()
```

4.9 CUDA-FORTRAN Results

The CUDA-FORTRAN results clearly show the impact of data transfer to and from the GPU for very large problem sizes, with the speedup being less than for smaller ones. The overall increase in performance is still very large however at 50x

Table 5 Table 6 Speed up figures for CUDA-FORTRAN

	CUDA-FORTRAN Total	*CUDA-FORTRAN Loop*
1000	0.061598	13.11111
10000	0.058095	65.26882
100000	4.899043	77.62784
1000000	20.29286	82.33236
10000000	49.27132	80.15689

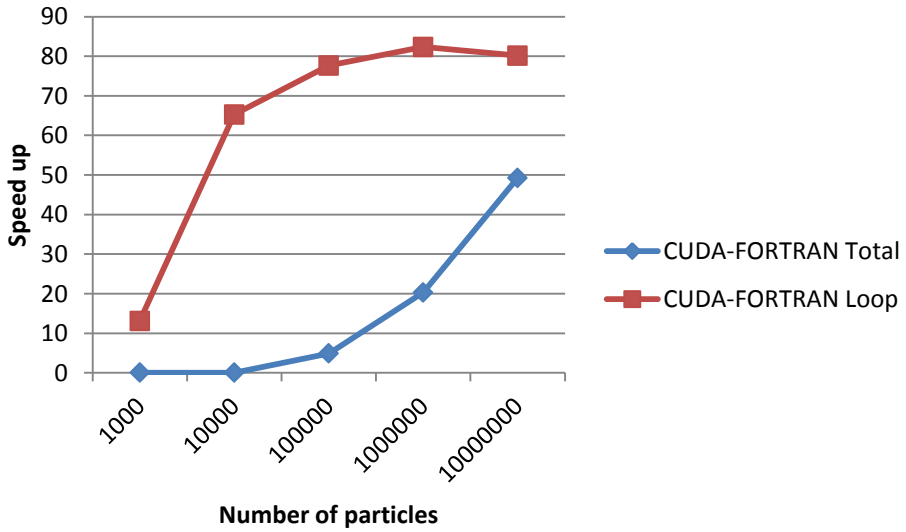


Figure 20 Speedup for CUDA-FORTRAN Code for various problem sizes

4.10 PGI FORTRAN Accelerator directives

PGI FORTRAN Accelerator directives can provide a very fast way to accelerate simple loop. This simplicity however comes at the cost of versatility.

For anything more than the simplest of loops the compiler fails to identify how to parallelise the loop, but perhaps the feature which makes this technique easy to rule out is that the compiler cannot accelerate loops which contain subroutine calls. This would mean that all subroutines would have to be in-lined manually which in many cases would make the code very hard to maintain.

With the advent of CUDA-FORTRAN it's easy to see that this method for accelerating FORTRAN code may rapidly lose favour for nearly all cases except where it's directive approach makes maintaining serial, parallel and accelerated codes simultaneously possible.

4.11 Analysis

CUDA acceleration offers massive speedups. Even without considering any further optimisation of thread layouts and compiler flags speedups of ~100x are observed

The total runtime doesn't tell the whole story here. In the Accelerated versions of the code timings were added to report the time spent within the loop, and the time spent performing data transfers. If these are plotted additionally the picture becomes clearer.

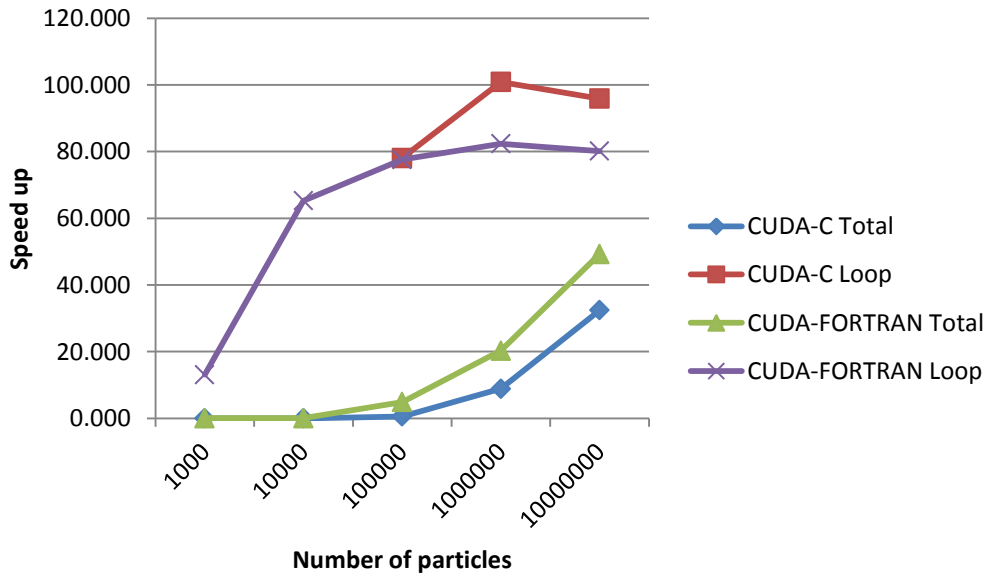


Figure 21 Comparison of overall and loop speedup for CUDA C and CUDA FORTRAN

This detailed breakdown reveals that the particle loop calculation time for CUDA-C and CUD-FORTRAN both offer very large speedups of over 80x for real problem sizes

CUDA-C performs faster memory transfers than FORTRAN is capable of, this difference widens as the size of the data being copied increases, up to nearly twice as fast for very large arrays (10,000 times original data set of 1,000).

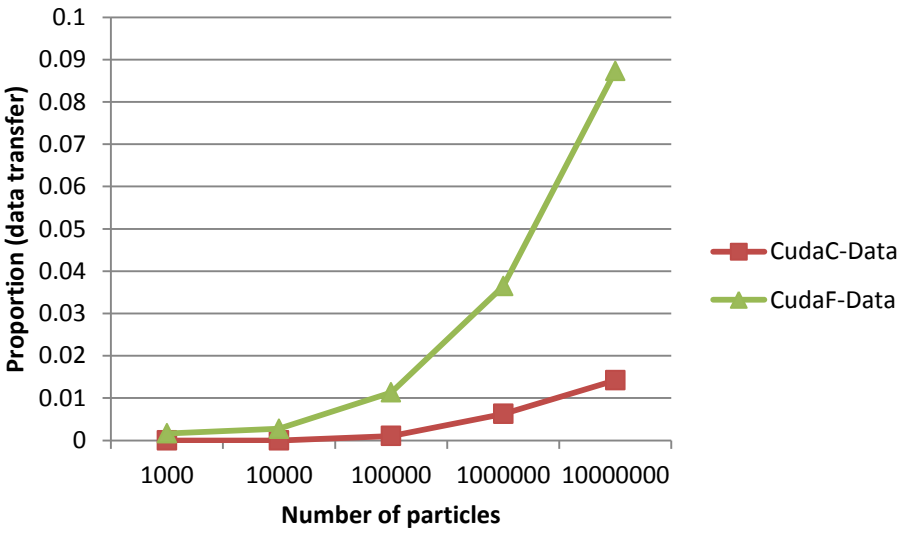


Figure 22 Comparison of proportion of runtime spent transferring data for CUDA-C and CUDA-FORTRAN codes

However overall The CUDA-FORTRAN code performs better. This is because serial portion of the FORTRAN code is much faster to execute than the C version, this is especially important for smaller particle numbers where the serial code has a much larger impact.

CUDA-C took erroneously longer to allocate (c malloc) the initial array on the Fermi0 machine. A trial run on Ness confirms the problem on a particle run of 1 million particles

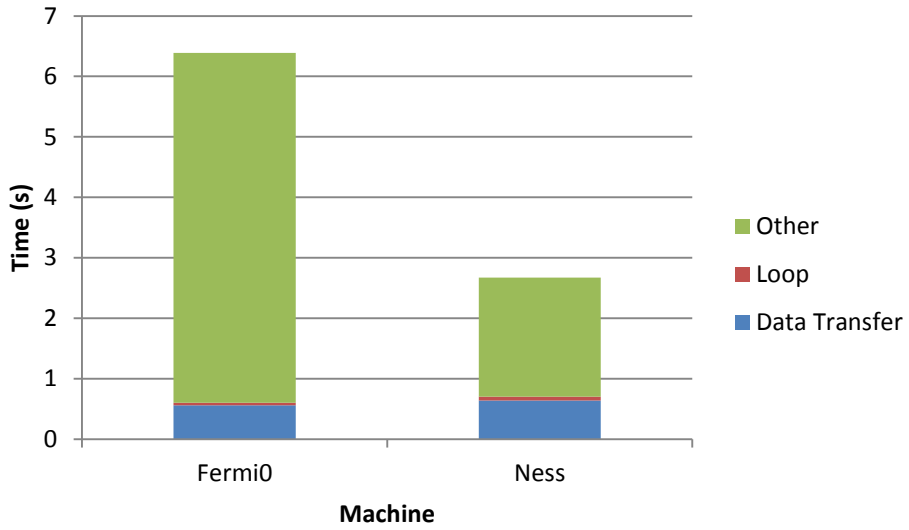


Figure 23 CUDA-C simple model takes nearly 3 times longer to allocate device arrays on Fermi0 compared to Ness

For very large problems therefore it might be tempting to consider the best implementation to be to maintain existing FORTRAN code and utilise CUDA-C for the accelerated portion in order to deliver the best of both worlds. Optimising the serial code should also be considered.

This option must be evaluated carefully however, maintaining C and FORTRAN operability is no simple task, and porting the kernel code from FORTRAN to c may in itself prove to be time consuming and error prone.

5 Towards accelerating NAME III

5.1 Code analysis

NAME III consists of many hundred thousands of lines of code. It utilises a very modern style of FORTRAN 90 programming implementing many derived types and a modularised structure.

The main functions of name are as follows [9], in this pseudo code items highlighted in blue form the basis of the simple model, the other areas are optional in a NAME III run.

```
Run to end of case
If Met data being used
     Update Met Data (eg import from UM)
End if
Run to sync time
Release new particles
If puffs
     Propagate Puffs for each time step
Else
     Parallel
          For each particle
               Propagate particles for each time step
          End for
     End parallel
End if
If Chemistry Required
     Convert particle densities to field concentrations
of species
     Parallel
          Loop over grid boxes
               Apply chemistry to field
               Update to particle densities
          End loop
     End Parallel
End if
Parallel
     For each particle
          Calculate contribution to output
     End for
End Parallel
Process output to write to disk
If end of case
     End
Else
     Goto start
End IF
```

5.2 Conformity of code to simple model

The particle loop from the NAME code is similar in concept to the Simple Model and as such certain extrapolations can be made as to the potential speedup from accelerating this region.

For situations such as that considered in the particle benchmark, which is broadly indicative of emergency response type codes, this give a good indication that utilising GPUs would give a great benefit. It should be noted however that many of the more complex use cases for NAME III would require much more advanced CUDA coding.

Before expanding these comparisons much must be considered:

5.3 Code design

CUDA acceleration works best on very simple kernels. Within the particle loop in NAME there are many procedure calls loops and conditional statements. These do not lend themselves well to CUDA implementation, in fact exiting a loop, for example when a particle is deemed to be removed from the system, is not supported at all.

CUDA requires a SIMD approach. In terms of the particle loop this would mean that the code should be written as if for one particle and every operation within the loop performed on that one particle. The parallelism is then added afterwards. Communication is available within the CUDA threads [18] but this has a large detrimental effect on performance as in many cases this prevents coalesced memory accesses.

Derived types are also unsupported in CUDA-FORTRAN [20] in all but the simplest fashion.

The largest problem to overcome however is simple that of size. There is a limited number of registers available on the CUDA device, the more of these each kernel requires the fewer kernels can be executed at a time, if very few kernels are implemented then the speed per core will come into play:

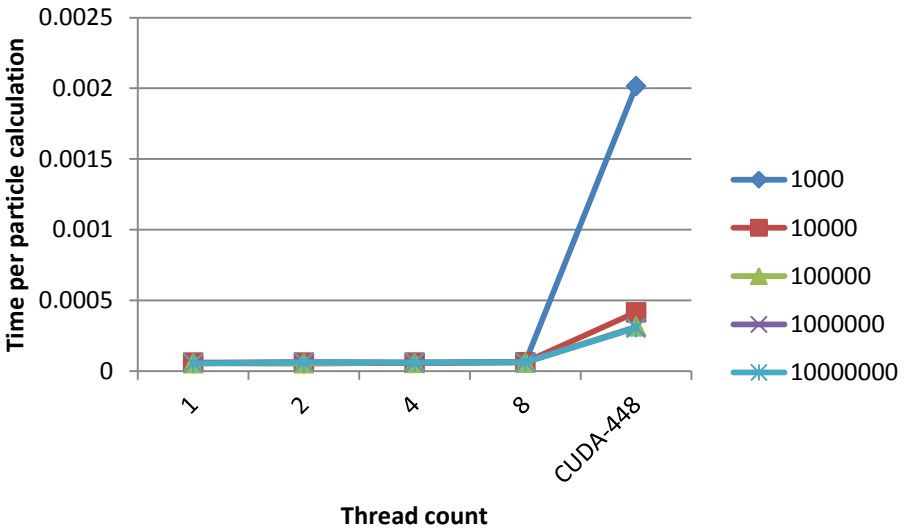


Figure 24 Real time per particle calculation

As is demonstrated above CUDA devices only gain their speed from massive parallelism, which must be achieved to make Acceleration worthwhile. The actual processing of each particle in the loop takes 6-30 times longer on the GPU than the CPU.

5.4 Acceleration potential

A best case scenario could be extrapolated from comparing the simple model and NAME Particle Benchmark 100,000 particles

Total runtime for NAME III benchmark	140.69s
Amount code parallelisable	94.6%
Time to run parallelisable code in serial	133.1s
Time to run non parallelisable code	7.61s
CUDA FORTRAN loop speedup for 100,000 particles from simple code	33.22x
Corresponding reduced time in NAME III for loop	4.01s
Total projected runtime for NAME III benchmark	11.62s

Total projected speedup from using CUDA-FORTRAN	12.1x

This would be the maximum possible speedup using the GPU as all other code within the particle loop except that of the lagrangian motion is ignored. Again here the speedup would be much improved for larger numbers or particles

Based on the linear properties of the NAME III code for 10 million particles the projection would be:

Total runtime for NAME III benchmark	14000
Amount code parallelisable	99.95%
Time to run parallelisable code in serial	13993
Time to run non parallelisable code	38s
CUDA FORTRAN loop speedup for 100,000 particles from simple code	70.5x
Corresponding reduced time in NAME III for loop	198s
Total projected runtime for NAME III benchmark	236s
Total projected speedup from using CUDA-FORTRAN	59x

As can clearly be seen the CUDA acceleration would have a massive impact in these large problems

5.5 Analysis of development issues

The modular approach to NAME's construction extends as far as targeting different hardware within the make files, by altering compile time variables the choice can be made between different hardware vendors, and different software vendors' compilers. There are also options for enabling OpenMP and for specifying the number of processors to be used.

This type of modularity is possible by these being compile time options. Since OpenMP uses directives these are simply ignored if the option is not enabled in the compiler.

Implementing CUDA however would require the code to be branched into Serial and CUDA versions (OpenMP can still be used with CUDA for the non-accelerated parts of the code). The easiest route to this would be to utilise a good version control system with support for branching such as Subversion [29]

Staff expertise may also be important, CUDA FORTRAN is a simple expansion of FORTRAN 95 so would be the simplest to learn for programmers new to CUDA but the lack of existing training material and support on the internet and in books would

require training courses to be undertaken and a support contract with the Portland Group for their compiler would be essential.

6 Further optimisations

6.1 CUDA Optimisations

Several optimisations can be made within the CUDA codes to improve performance. The first of which is altering the thread layout. This has more relevance for more complex memory access patterns than are present in the simple loop particle loop, but it's worth examining the impact this has on performance

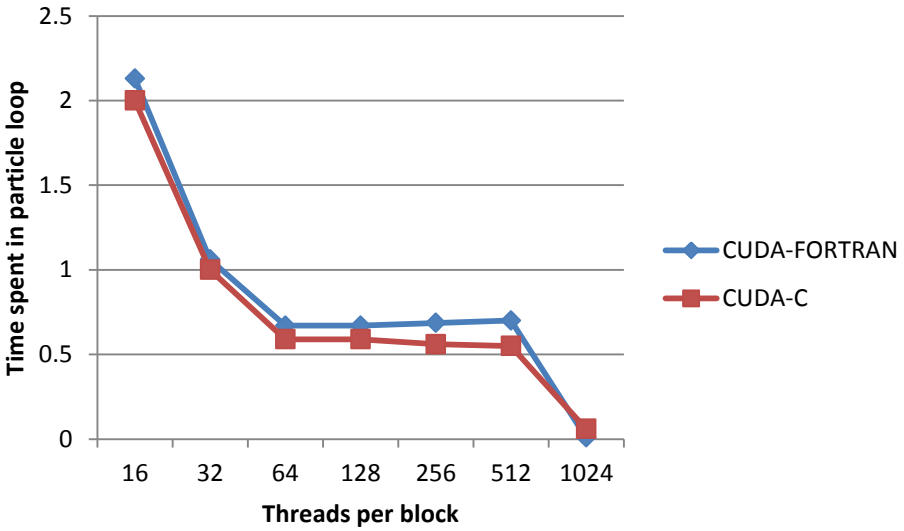


Figure 25 Performance variation with threads per block

As is evident here there's quite a variation in performance and there are no certain ways to calculate which thread layout will be best, a trial and error approach is needed. For the runs here 256 blocks per thread were used.

This was due to the maximum number of available threads per block on the Ness Development machine being 512. When upgrading to the newer Fermi architecture this is increased to 1024, which correspondingly offers a performance increase, shown here by the sharp down tick.

Use of CUDA fast maths is invoked with the compiler flag `-use_fast_math` or `-ta=nvidia,fastmah` for FORTRAN. This enables the GPU to perform calculations in a much simpler fashion at the expense of some accuracy. For the simple code this was below the level of precision required so this option can be enabled without issue and provides a small increase in performance.

Also CUDA optimisations can be used along with normal serial code optimisations such as the `-O3` of `-fast` compiler flags which all help to reduce the total runtime

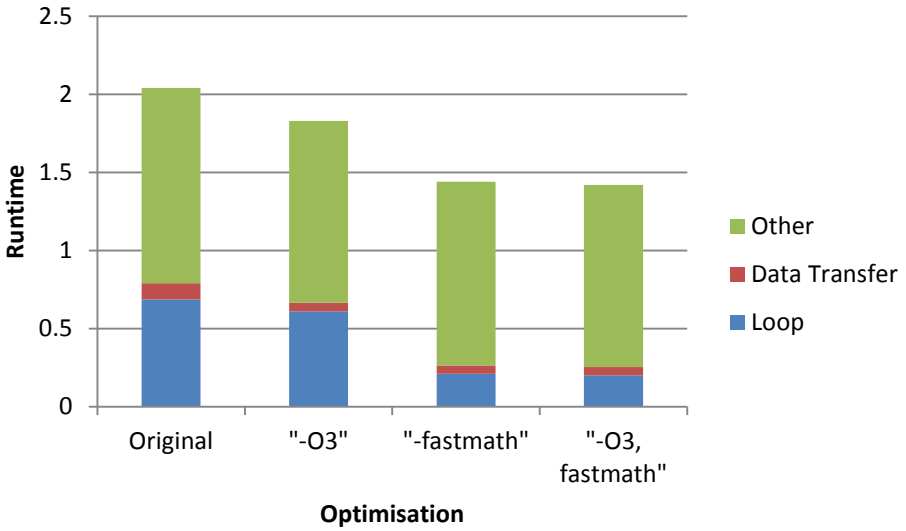


Figure 26 Utilising optimisation methods for 1 million particle run

Here the graph shows that the CUDA fast math flag cuts more than 2/3 from the loop runtime and serial optimisations shave 10% of the non-CUDA sections of the code.

6.2 Other areas for Optimisation

The Cloud_Gamma benchmark was also supplied and has not been previously discussed since a simple model was not available for comparison.

An investigation of the code however can lead to some understanding of the possibility for the optimisation of this area.

The loop over a 3d grid is much simpler, there are far fewer function calls, and the branching can all be completed outside the loop, allowing it to comprise exclusively of calculations, which is good for porting to CUDA.

The 3D layout of the grid performed poorly in OpenMP parallelisation, but CUDA in better equipped to deal with this type of memory layout using `cudaMalloc3D()` [] there are also geometric function libraries for CUDA which may be utilised cg provides an optimised distance function for calculating the Euclidean distance between two points.

The main issue is most likely that there problem size of the 3d grid would not be very large, this will lead to the data transfer time to and from the GPU having a significant impact since data is transferred for each particle in each time step.

7 Discussion simple model of results

The results from accelerating the simple model show a good agreement of this with the work of F. Molnár et al. [8] speed up of 90x was observed for using CUDA.

Performance for the simple code OpenMP doubled with a doubled number of processors indicating an agreement with observation of the OpenMP version of the NAME III code. This means there is near linear weak scaling as the runtime remained almost identical.

Strong scaling was also nearly linear in the simple model, which agrees with observations of the OpenMP NAME III code with speedup doubling with double the number of threads for a fixed problem size

Excellent strong and weak scaling together show that large problems on large systems would be most efficient.

There was little difference between the on GPU performance between using CUDA-C and CUDA-FORTRAN. This is most likely due to the very similar restricted implementation of functions on the GPU

CUDA-FORTRAN exhibited slower data transfer times than CUDA-C for exchanging data with the GPU. This is most likely due to the higher level of the interface provided for data transfer in the FORTRAN implementation, which does enable code to be developed more rapidly. Interfaces to nave CUDA-C type calls are also available for manual control of the data transfer, which may improve the performance at the expense of code readability.

8 Conclusions

Lagrangian particle dispersion models are excellent candidates for accelerating using GPUs, performance compared to serial execution can be expected to be about 100x faster.

To help draw concussions as to the impact this could have on NAME III it's informative to consider all the overall speed ups together:

Table 7 Particle loop speed ups for different parallelisation methods

	P= 2	*4*	*8*	*CUDA-C*	*CUDA-FORTRAN*
10^3	2.185	3.933	7.375	--	13.111
10^4	2.010	4.020	7.684	--	65.269
10^5	1.964	3.725	6.874	78.071	77.628
10^6	1.805	3.858	7.223	100.857	82.332

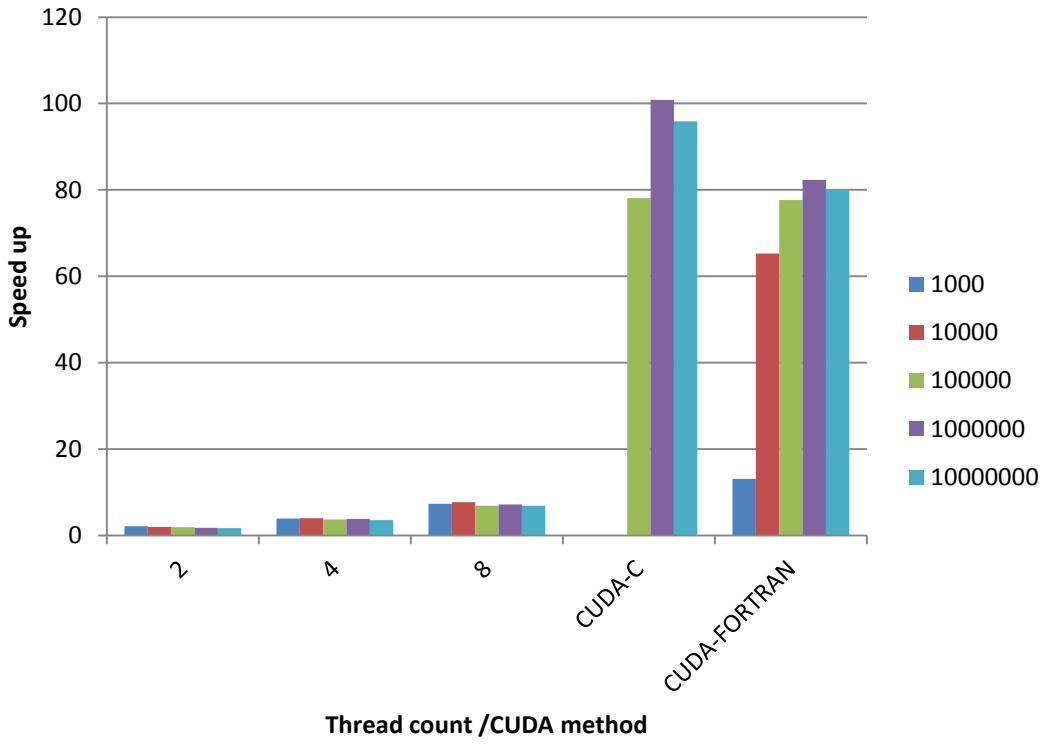


Figure 27 Particle loop speed up for OpenMP and CUDA compared

Certainly parts of the code which conform to this model would benefit from the acceleration and enable much larger runs to be completed in the same time as is presently allocated.

For the simple case used in emergency response, which is similar to the particle benchmark examined here, the runtime can be reduced to a small fraction, which would enable emergency services to respond much quicker.

The problems with adopting an accelerator model for NAME III are twofold and primarily concerned with the overall complexity of the code.

NAME III currently provides a one size fits all approach, both in terms of supported simulation models and supported hardware for running the code. In order to provide a simpler route to accelerating sections, development would have to be branched in both these directions. Complex sections of NAME III code could be replaced by smaller efficient kernels which could be used for specific sets of data, for example emergency response.

NAME III is very well structured and optimised for complex serial runs on complex processors. The whole premise of this model is contrary to that of using GPUs, were the NAME III code designed from the ground up in a GPU centric fashion NAME could be expected to perform excellently.

Given the code's current structure however and performance an alternative route to that of GPU acceleration might prove more efficient; that of Hybrid OpenMP and MPI. This proposition is reinforced when the speed up for the whole code is considered including data transfers and communication overheads:

Table 8 Overall speedups of different parallelisation methods

Particles	*P= 2*	*4*	*8*	*CUDA-C*	*CUDA-FORTRAN*
10^3	0.381	0.460	0.421	0.012	0.062
10^4	0.198	0.389	0.709	0.012	0.058
10^5	1.955	3.422	6.014	0.588	4.899
10^6	1.800	3.811	7.050	8.892	20.293
10^7	0.381	0.460	0.421	0.012	0.062

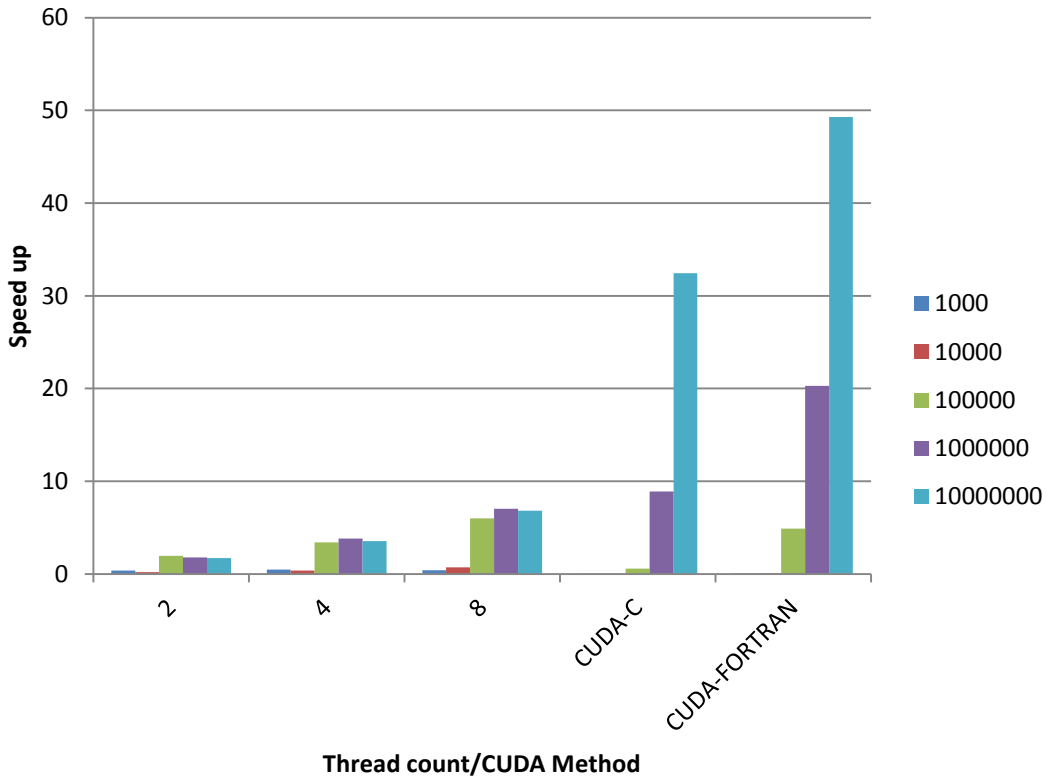


Figure 28 Overall speed up for OpenMP and CUDA compared for various particle numbers (problem sizes)

The graph of overall speed up shows that for all but very large problems (tens of millions of particles) the total speedup could easily be achieved with a small cluster, a 4x8 core system would give similar results to CUDA-FORTRAN for problem sizes of about 1 million particles for instance.

Although using this approach would not yield the maximum performance gains available for the particle loop demonstrated here, it is much more compatible with the features used and existing structure of the code. It may well prove that partially parallelising more of the code using this method results in shorter run times overall for a wider variety of use cases.

However typical problem sizes for real world runs of NAME III are of the order of tens of millions [2], and there is more than a doubling of performance when using the GPU for this large problem size when compared to one of the order of a million.

The independence of the loop operations would mean that there would not be overly complex communications, and that communications could be readily overlapped with calculations.

The hybrid model could also be implemented using the free GNU compiler which would be more appealing to academic institutions, or the existing Intel compiler providing a smoother transition route for existing users.

9 References

[1] Thomson D.J., Hort M. and Devenish B. Jones A.R., *U.K. Met Office's next-generation atmospheric dispersion model, NAME III*.

[2] Eike Müller, *Telephone Conversation*.

[3] © Met Office, *Email Communication*.

[4] Public Weather Service Outputs; Met Office Public Tasks, *http://www.metoffice.gov.uk/media/pdf/e/e/Public_Weather_Service_Outputs.pdf*.

[5] Met Office VAAC Website, *http://www.metoffice.gov.uk/aviation/vaac/index.html*.

[6] Met Office Website, *© Copyright EUMETSAT/Met Office*.

[7] J Nijdam, B Guo, D Fletcher, and T Langrish, "Lagrangian and Eularian models for simulating turbulent dispersion and agglomeration of droplets within a spray," , Melbourne, 2003.

[8] Benjamin Devenish, "Notes on Lagrangian stochastic models," 2009.

[9] Eike Müller, *NAME Code Structure*., 2011.

[10] F. Molnár Jr., T. Szakály, R. Mészáros, and I. Lagzi, "AIR POLLUTION MODELLING USING A GRAPHICS PROCESSING," 2009.

[11] Gordon E. Moore, "Cramming more components," *Electronics, Volume 38, Number 8*, 1965.

[12] Gordon E Moore, *ftp://download.intel.com/museum/Moores_Law/Video-Transcripts/Excepts_A_Conversation_with_Gordon_Moore.pdf*.

[13] Wikipedia, *http://en.wikipedia.org/wiki/Moore's_law*.

[14] CPU World, *http://www.cpu-world.com/Cores/Smithfield.html*.

[15] CPU World, *http://www.cpu-world.com/CPUs/Bulldozer/AMD-Opteron%206276.html*.

[16] CDC Star/Wikipedia, *http://en.wikipedia.org/wiki/CDC_STAR-100*.

[17] CRAY, *http://www.cray.com/Products/XE/Technology.aspx*.

[18] CPU World, *http://www.cpu-world.com*.

[19] Coolest Gadgets, *http://www.coolest-gadgets.com/20071122/ageia-ut3-mod-kit-released/*.

[20] NVIDIA, *CUDA-C Programming Guide*, http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf ed.

[21] Digital content producer, *http://blog.digitalcontentproducer.com*.

[22] EPCC, *Advanced Topics Course Notes*.

[23] Portland Group Inc, *CUDA Fortran Programming Guide and Reference*.: http://www.pgroup.com/lit/whitepapers/pgicudaforug.pdf.

[24] Portland Group Inc, *http://www.pgroup.com/resources/cudafortran.htm*.

[25] Wikipedia, *http://en.wikipedia.org/wiki/Amdahl%27s_law*.

[26] Wikipedia, *http://en.wikipedia.org/wiki/Gustafson%27s_Law*.

[27] Free Software Foundation inc, *http://sourceware.org/binutils/docs-2.21/gprof/index.html*.

[28] Portland Group Inc, *Compiler release notes*.: http://www.pgroup.com/doc/pgiwsrn108.pdf.

[29] Met Office Dispersion Model, *http://www.metoffice.gov.uk/research/modelling-systems/dispersion-model*.

[30] Tsuneo MATSUNAGA, Hitoshi MUKAI Jiye ZENG, "Using NVIDIA GPU for Modelling the Lagrangian Particle Dispersion," , Ottawa, 2010.

[31] Eike Müller, *Parallelisation with OpenMP*., 2010.

[32] Eric Weisstein, *Box-Muller Transformation*.: http://mathworld.wolfram.com/Box-MullerTransformation.html.

[33] Apache Subversion, *http://subversion.apache.org/*.

[34] Met Office Website, *http://www.metoffice.gov.uk/research/modelling-systems/unified-model*.

10 Appendices

10.1 NAME III Code Licence

SOFTWARE LICENCE FOR NON-COMMERCIAL USE

LICENCE PARTICULARS

LICENSEE DETAILS

Licensee name:	**University of Edinburgh, EPCC**
Licensee address:	James Clerk Maxwell Building, Mayfield Road, Edinburgh, EH9 3JZ, UK
Licensee contact details:	Name of main contact: **Xu Guo** Telephone number: **+44 (0)131 651 3530** Email address: **xguo@epcc.ed.ac.uk** Fax number: **+44 (0)131 650 6555**

MET OFFICE DETAILS

Met Office address:	**FitzRoy Road, Exeter, Devon, EX1 3PB, United Kingdom**
Met Office contact details:	Name of main contact: **Matthew Hort** Telephone number: **+44 (0)1392 886242** Email address: **matthew.hort@metoffice.gov.uk** Fax number: **+44 (0)1392 885681**
Licence Reference Number:	**L0665**

For the purposes of this section of the Licence Particulars, the following words shall have the following meaning:

"**Educational Use**" shall mean use of the Software:

- in the course of teaching which takes place at a school, college or university; or
- use, by a school, college or university, for the generation of teaching materials; or
- use, in conjunction with the published results, know-how or outputs arising from Non-Commercial Research, for the provision of occasional, un-marketed and nominally remunerated consultancy by a school, college or university;

"**Non-Commercial Research Use**" shall mean original investigation to which all of the following conditions apply:

- neither it nor its results have been commissioned exclusively by any person and/or organisation in the private sector;
- the results, know-how and outputs arising from it will not be used, displayed, presented or made available in such a way as to directly or indirectly create, build, further or promote a commercial relationship between the licensee and a third party or in such a way that a third party could use all or part of the results, know-how or outputs to further his/her/its commercial interests;

- the results, know-how and outputs arising from it will be made public as soon as reasonably practicable, without restriction of access and at a fee which does not exceed the cost of delivery;
- it does not involve the use of live (real-time) data feeds;
- its scope and nature will be summarised in writing to the satisfaction of the Met Office and approved in writing by the Met Office before a licence is granted;
- the applicable licence will be time limited and may not be used for activities which go beyond the authorised research.

THE LICENCE TERMS

Permitted Use of the Software	**Delete as appropriate**	
Educational Use		NO
Non-Commercial Research Use	YES	

Licence Term	1 year commencing on 5th November 2010
Special Conditions (if any) *Note: Anything documented in this section will override the Terms in accordance with clause 2.7.* There are no special conditions.	**Notes for key changes:**

DESCRIPTION OF THE SOFTWARE

Software name	**Software description**	**Version**	**Release date**	**Supply date**
NAME III	Met Office atmospheric dispersion model	**6.0**		Nov 2010

Form of the Software	**Delete as appropriate**	
Source Code	YES	
Object Code / Executable	YES	

SUPPORTING DOCUMENTATION

Description of supporting documentation required to run the Software (if any):	Model documentation appropriate to the version of NAME supplied (as included in the standard model build).

These Licence Particulars are subject to the Software Licence attached and you acknowledge that you have read, understood, and agree to be bound by those terms and conditions for and on behalf of the Licensee.

……	……
SIGNATURE	LICENSEE NAME
……	
PRINT NAME	
……	……
JOB TITLE	DATE

Signed by a duly authorised representative of the Met Office for and on behalf of the Secretary of State for Defence of the United Kingdom of Great Britain and Northern Ireland ("**Met Office**"):

……	
SIGNATURE	
……	
PRINT NAME	
……	……
JOB TITLE	DATE

THE MET OFFICE

SOFTWARE LICENCE FOR NON-COMMERCIAL USE

The Licensee's attention is drawn in particular to the provisions of clauses 6 and 8.

AGREED TERMS:-

1. Interpretation

1.1. The definitions and rules of interpretation in this clause apply in these Terms and in the Licence Particulars.

"**Business Day**" shall mean a day other than a Saturday, Sunday or public holiday in England.

"**Commercial Benefit**" shall mean the receipt of any revenue or credit by the Licensee, excluding the receipt by the Licensee of a research grant, arising from the use by the Licensee of the Software.

"**Insolvency Event**" shall mean in relation to the Licensee any of the following events:

(a) a meeting of the creditors of that person being held for an arrangement or composition with or for the benefit of its creditors (including a voluntary arrangement) being proposed by or in relation to that person;

(b) a charge holder, receiver, administrative receiver or other similar person taking possession of or being appointed over or any distress, execution or other process being levied or enforced (and not being discharged within 7 days) on the whole or a material part of the assets of that person;

(c) that person ceasing to carry on business, stops paying its debts as they fall due or being deemed to be unable to pay its debts;

(d) that person or its directors or the holder of a qualifying floating charge giving notice of their intention to appoint, appointing or making an application to the court for the appointment of, an administrator;

(e) a petition being presented (and not being discharged within 28 days) or a resolution being passed or an order being made for the administration or the winding up, bankruptcy or dissolution of that person; or

(f) the happening in relation to that person of an event analogous to any of the above in any jurisdiction in which it is incorporated or resident or in which it carries on business or has assets.

"**Intellectual Property Rights**" shall mean all patents, rights to inventions, utility models, copyright and related rights, trade marks, service marks, trade, business and domain names, rights in trade dress or get-up, rights in goodwill or to sue for passing off, unfair competition rights, rights in designs, rights in computer software, database right, topography rights, moral rights, rights in confidential information (including know-how and trade secrets) and any other intellectual property rights, in each case whether registered or unregistered and including all applications for and renewals or extensions of such rights, and all similar or equivalent rights or forms of protection in any part of the world.

"**Licence**" shall mean this licence consisting of the Licence Particulars and the Terms.

"**Licence Particulars**" shall mean the document entitled as such which contains the Met Office's confirmation of the terms of the Licence.

"**Licence Term**" shall mean the period for which the Licence is granted as specified in the Licence Particulars commencing on the date upon which the Licence Particulars are signed by both the Met Office and the Licensee.

"**Licensee**" shall mean the person identified in the Licence Particulars.

"**Modify**" shall mean the Licensee making changes and/or improvements to the Software in its completed and final format.

"**Permitted Use**" shall mean the use which the Met Office shall permit the Licensee to make in respect of the Software as detailed in the Licence Particulars.

"**Purpose**" shall mean shall mean the purpose for which the Licensee wishes to secure a licence of the Software which is detailed in the Licence Particulars.

"**Software**" shall mean the Software specified in the Licence Particulars, any documentation required to run the Software which is specified in the Licence Particulars and any Upgrades which the Met Office may provide to the Licensee from time to time.

"**Terms**" shall mean these terms and conditions.

"**Upgrade**" shall mean the Met Office amending, improving or upgrading the Software in its completed and final format.

1.2. Headings in these conditions shall not affect their interpretation.

1.3. A **person** includes a natural person, corporate or unincorporated body (whether or not having separate legal personality).

1.4. References to a party or parties shall mean a party or parties to the Licence.

1.5. A reference to a statute or statutory provision is a reference to it as it is in force for the time being, taking account of any amendment, extension, or re-enactment and includes any subordinate legislation for the time being in force made under it.

1.6. Any phrase introduced by the terms **including**, **include**, **in particular** or any similar expression shall be construed as illustrative and shall not limit the sense of the words preceding those terms.

1.7. A reference to writing includes faxes and email.

2. Grant of Licence

2.1. The Met Office grants the Licensee a worldwide, non-exclusive, organisation-wide, non-transferable licence for the Licence Term to use the Software solely for the Purpose, and to the extent permitted by the Permitted Use. No right or licence is granted by the Met Office to the Licensee except as expressly set out in this clause 2.

2.2. The Licence is granted on a free of charge basis.

2.3. For the purposes of clause 2.1, **use** of the Software shall be restricted to use of the Software in the form which is specified in the Licence Particulars.

2.4. The Licensee may:

2.4.1. copy, run and store the Software both on the Licensee's computer system and at its site notified to the Met Office in writing on the date of this Licence; and

2.4.2. make back-up copies of the Software for its lawful use. The Licensee shall record the number and location of all copies of the Software and take steps to prevent unauthorised copying.

2.5. If the Licensee wishes to renew the Licence, the Licensee shall submit a written request to the Met Office no less than 3 months prior to expiry of the Licence Term. The Licensee acknowledges and agrees that the Met Office's decision on whether to renew the Licence shall be at the Met Office's sole discretion.

2.6. The Met Office will endeavour to provide the Licensee with the Software within one calendar month of the date upon which the Licence Particulars are signed by both the Met Office and the Licensee. However, time shall not be of the essence in respect of the supply of the Software by the Met Office.

2.7. If any of these Terms are inconsistent with any term set out in the Licence Particulars, the Licence Particulars shall prevail.

3. Intellectual Property Rights

3.1. All Intellectual Property Rights and all other rights in the Software and any Upgrade shall be owned by the Met Office (on behalf of the Crown) and its licensor(s). All Software licensed to the Licensee remains the property of the Met Office (on behalf of the Crown) and, if applicable, its licensor(s) and the Licensee's right to use the Software for the Permitted Use will not give it any ownership rights or other interest in any of the Software.

3.2. The Licensee agrees to take all reasonable steps to prevent any damage to or infringement of the Met Office's Intellectual Property Rights.

3.3. The Licensee shall make acknowledgement to the "Met Office" in any reproduction of data, publication of papers, reports, literature to customers, or presentations arising out of the use of the Software using the following acknowledgement:

"Material produced using Met Office Software"

3.4. The Licensee shall reproduce on any copy of the Software and accompanying documentation a Crown Copyright acknowledgement in the following form:

where the reproduction will be wholly within the UK:

"© Crown copyright [followed by year of first publication], the Met Office"

OR

where the reproduction will occur outside the UK:

"© British Crown copyright [followed by year of first publication], the Met Office"

3.5. The words "Met Office" and the Met Office device and logos are registered trade marks in the United Kingdom, the European Union, the United States of America and other countries. These trade marks are the property of the Secretary of State for Defence of the United Kingdom of Great Britain and Northern Ireland. The Licensee may not use any trade mark, service mark, logo, corporate or business name of the Met Office without the Met Office's prior consent in writing.

3.6. The Licensee shall ensure that the Software in its possession is secure and that adequate technological security measures are taken to ensure that the Software is not accessed or used by unauthorised persons. The Licensee shall notify the Met Office immediately if the Licensee becomes aware of any unauthorised use of the Software by anyone or of any actual or potential infringement of the Met Office's Intellectual Property Rights in the Software. The Licensee shall permit the Met Office at any time to check that the use of the Software is in accordance with the terms of this Licence.

3.7. Subject to the provisions of clause 3.1, ownership of the output resulting from the Licensee's use of the Software shall belong to the Licensee.

3.8. Save where the parties otherwise agree in writing, the Licensee grants to the Met Office a non-exclusive, perpetual, irrevocable, royalty-free and fully paid-up worldwide right and licence to produce, re-produce, copy, publish, develop, adapt, offer for sale, sell and/or distribute or otherwise use any Intellectual Property Rights created by the Licensee using the Software for Official Duties and/or for Non-Commercial Research Use for the full duration of such rights, such right and licence to include the right to sub-licence or otherwise transfer any and all of the aforesaid such rights to any third party. For the purposes of this clause 3.9, "Official Duties" means government funded activities, which an organisation is required to undertake for statutory or legal reasons or in support of governmental or intergovernmental requirements, which may include but is not limited to the area of aviation, defence or public safety and "Non-Commercial Research Use" has the meaning which is given in the Licence Particulars.

3.9 To the extent that the any Intellectual Property Rights arise or are created, produced, developed or acquired by Licensee in the development of modifications and/or improvements to the Software, they are hereby assigned, as a continuing obligation, into the name of the Met Office (on behalf of the Controller of Her Majesty's Stationery Office and Queen's Printer (in respect of non-registrable Intellectual Property Rights), and/or the Secretary of State for Defence of the United Kingdom of Great Britain and Northern Ireland (in respect of registrable Intellectual Property Rights)), to hold unto itself absolutely, with full title guarantee for the full duration of such rights, wherever in the world enforceable.

3.10 The Licensee shall execute and/or shall procure that any other person they shall engage in producing developments, modifications and/or improvements to the Software, including without limitation all staff, officers, employees, students, agents, or sub-contractors executes, all documents and assignments and does all such things necessary in order to be able to give effect to the provisions of **Clause 3.9**

3.11 The Met Office hereby grants to the Licensee, as a continuing obligation, a perpetual, irrevocable, non-exclusive, royalty-free and fully paid-up worldwide right and licence to produce, re-produce, copy, publish, develop, adapt, distribute or otherwise use all its interest in the Software modifications and/or improvements assigned to the Met Office pursuant to **Clause 3.9**, for any purpose (including without limitation commercial exploitation) for the full duration of such rights, such right and licence to include the right to sub-licence any and all of the aforesaid such rights to any third party.

4. Upgrades

4.1. The Met Office may Upgrade the Software during the Licence Term and may, at its option, choose to supply the Licensee with such Upgrade(s).

4.2. In the event that the Met Office chooses to supply an Upgrade to the Licensee, the Licence Particulars shall be amended so as to refer to the upgraded version of the Software. The Licensee agrees that its acceptance of such an Upgrade shall be deemed as acceptance of this amendment.

4.3. Nothing in these Terms shall oblige the Met Office to Upgrade the Software or supply such Upgrades to the Licensee.

5. Licensee obligations and restrictions

5.1. The Licensee shall:

5.1.1. only make use of the Software to the extent necessary for the Purpose, and in accordance with, the Permitted Use;

5.1.2. on or before the date on which the Licence Particulars are signed by the parties and in any event within seven (7) days of that date, supply an abstract of the description of the work to be undertaken, which shall include as a minimum the project title and a description of the work which the Licensee intends on carrying out using the Software (the "**Work**");

5.1.3. on request and within a reasonable time, provide the Met Office with an annual report on the results obtained using the Software and a description of any module the Licensee has developed pursuant to clause **Error! Reference source not found.**. This report shall contain as a minimum, a description of the work undertaken and the results achieved in the previous year (or any part thereof) and a description of any future work plan;

5.1.4. provide to the Met Office, free of charge, a copy of any papers publishing the results of the Work **OR** where nothing is published, a summary of the use which was made of the Software and any conclusions or findings;

5.1.5. provide to the Met Office, free of charge, suggested revisions to the Software to resolve problems, improve portability and improve efficiency;

5.1.6. be entitled to Modify the Software; and

5.1.7. where possible, permit the Met Office to inspect and have access to any premises, and to the computer equipment located there, at or on which the Software is being kept or used, and any records kept pursuant to this Licence, for the purposes of ensuring that the Licensee is complying with the terms of this Licence, provided that the Met Office provides reasonable advance notice to the Licensee of such inspections, which shall take place at reasonable times.

5.2. Save as expressly provided for in this Licence, the Licensee shall not:

5.2.1. distribute, licence, transfer, assign, sell or disclose to or otherwise forward the Software or any associated documents, software, documentation, or other information to any third party including subsidiary companies without the express prior written permission of the Met Office; or

5.2.2. (subclause deleted); or

5.2.3. use the Software to generate a Commercial Benefit; or

5.2.4. electronically transfer the Software over a network to a computer system that is not owned or controlled by the Licensee or otherwise transfer any rights to the Software without the express prior written permission of the Met Office; or

5.2.5. do anything that may bring the name of the Met Office into disrepute or which damages or dilutes the goodwill associated with the name and trade marks of the Met Office; or

5.2.6. (and shall not permit any third party) to copy, adapt, reverse engineer, decompile, disassemble, modify, adapt or make error corrections to the Software in whole or in part except to the extent that any reduction of the Software to human readable form (whether by reverse engineering, decompilation or disassembly) is necessary for the purposes of integrating the operation of the Software with the operation of other software or systems used by the Licensee, unless the Met Office is prepared to carry out such action at a reasonable commercial fee or has provided the information necessary to achieve such integration within a reasonable period, and the Licensee shall request the Met Office to carry out such action or to provide such information (and shall meet the Met Office's reasonable costs in providing that information) before undertaking any such reduction.

6. Warranties

6.1. The Met Office warrants that:

6.1.1. it is authorised by the Controller of Her Majesty's Stationery Office and any relevant third party to grant licences for exploitation of UK Crown Copyright works; and

6.1.2. to the best of its knowledge and belief it is the owner of the Intellectual Property Rights in the Software or that it is duly licensed to use the Intellectual Property Rights in the Software.

6.2. The Software is supplied and used entirely at the risk of the Licensee and the Licensee accepts responsibility for the selection of the Software to achieve its intended results.

6.3. Other than the warranties expressly set out in these Terms and/or in the Licence Particulars, the Met Office excludes all warranties or representations (express or implied) including any in respect of the accuracy, compatibility, performance or fitness for purpose of the Software to the fullest extent permitted by applicable law.

6.4. The Met Office does not:

6.4.1. make any representations as to the compatibility of the Software

with the Licensee's computer operating systems and platforms; or

6.4.2. warrant or undertake that the Software will satisfy the Licensee's requirements; or

6.4.3. warrant that the Software will be without errors or defects; or

6.4.4. warrant that the operation of the Software shall be uninterrupted.

7. Confidentiality

7.1. Subject to the provisions of clauses 7.2 and 7.3, each party:

7.1.1. shall treat as strictly confidential and use solely for the purposes contemplated by the Licence all information, whether technical or commercial, obtained or received by it as a result of entering into or performing its obligations under the Licence and relating to the negotiations relating to, or the provisions or subject matter of, the Licence or the other party ("confidential information"); and

7.1.2. shall not, except with the prior written consent of the party from whom the confidential information was obtained publish or otherwise disclose to any person any confidential information.

7.2. Each party may disclose confidential information which would otherwise be subject to clause 7.1 if (but only to the extent that) it can demonstrate that:

7.2.1. such disclosure is required by law or by any securities exchange or regulatory or governmental body having jurisdiction over it, wherever situated, and whether or not the requirement has the force of law;

7.2.2. the confidential information was lawfully in its possession prior to its disclosure by the other party (as evidenced by written records) and had not been obtained from the other party; or

7.2.3. the confidential information has come into the public domain other than through its fault or the fault of any person to whom the confidential information has been disclosed in accordance with clause 7.1.

7.3. Each party may for the purposes contemplated by the Licence disclose confidential information to the following persons or any of them, provided that it procures the compliance of each such person with confidentiality obligations which are no less onerous than those set out in this clause 7:

7.3.1. its professional advisers, auditors, bankers and insurers, acting as such; and

7.3.2. its directors, officers, senior employees, and permitted sub-contractors.

8. Limitation of liability - THE LICENSEE'S ATTENTION IS PARTICULARLY DRAWN TO THIS CLAUSE

8.1. This clause 8 sets out the entire financial liability of the Met Office (including any liability for the acts or omissions of its employees, agents, consultants, and subcontractors) to the Licensee in respect of:

8.1.1. any breach of the Licence including any deliberate personal repudiatory breach;

8.1.2. any use made by the Licensee of the Software or any part of it; and

8.1.3. any representation, statement or tortious act or omission (including negligence) arising under or in connection with the Licence.

8.2. Nothing in these Terms limits or excludes the liability of the Met Office:

8.2.1. for death or personal injury resulting from negligence; or

8.2.2. for any damage or liability incurred by the Licensee as a result of fraud or fraudulent misrepresentation by the Met Office; or

8.2.3. any other matter for which it would be illegal or unlawful for the Met Office to exclude or attempt to exclude its liability.

8.3. Subject to clause 8.2, the Met Office shall not have any liability to the Licensee (howsoever arising, including any liability in tort) under or in connection with the Licence for any:

8.3.1. loss of income or revenue;

8.3.2. loss of business;

8.3.3. loss of opportunity;

8.3.4. loss of profits or contracts;

8.3.5. loss of anticipated savings;

8.3.6. loss of data;

8.3.7. loss of or damage to reputation or goodwill;

8.3.8. wasted management and/or other staff and/or office time;

in each case whether direct, indirect, special and/or consequential loss or damage; or

8.3.9. for any other indirect, consequential and/or special loss or damage.

8.4. Subject to clause 8.2, the Met Office's total liability in contract, tort (including negligence or breach of statutory duty), misrepresentation, restitution or otherwise arising in connection with the performance, or contemplated performance, of the Licence shall be limited to the sum of one thousand pounds (£1,000).

8.5. Subject to clause 8.6, the Licensee shall keep the Met Office fully and effectually indemnified against all actions, claims, proceedings, costs and/or damages, together with all legal costs or expenses that the Met Office may incur as a result of licensing with the Licensee, including but not limited to:

8.5.1. any other claim by a third party made against the Licensee resulting from the use or modification of the Software by the Licensee; or

8.5.2. the Licensee's negligent or unlawful acts or omissions, or its wilful misconduct; or

8.5.3. any breach by the Licensee of any of the terms of this Licence.

8.6. The indemnity in clause 8.5 shall not apply to the extent that the Met Office has contributed to its own loss or damage by its negligence, or unlawful acts or omissions, or its wilful misconduct.

8.7. The Licensee acknowledges and agrees that the Licence is granted to the Licensee on a free of charge basis having due regard to the exclusions, waivers and limitations set out in this clause 8 and in clause 6.

9. Termination

9.1. Subject to prior termination under clause 9.2, 9.3 or 9.4 either party may terminate the Licence by giving to the other party not less than thirty (30) days' written notice.

9.2. Without prejudice to any other rights or remedies which the parties may have, either party may terminate the Licence without liability to the other immediately on giving notice to the other if the other party commits a material breach of any of the terms of the Licence and (if such a breach is remediable) fails to remedy that breach within 30 days of that party being notified in writing of the breach.

9.3. Notwithstanding any other provision of this Licence, the Met Office reserves the right to refuse a Licence or terminate an existing Licence immediately on Ministry of Defence and/or Met Office security grounds.

9.4. The Met Office may terminate the Licence without liability to the Licensee immediately on notice to the Licensee if:

9.4.1. the Licensee breaches the terms of the licence which is granted by the Met Office in clause 2;

9.4.2. the Licensee fails to supply the Met Office with the abstract which is referred to in clause 5.1.2;

9.4.3. the Licensee ceases or threatens to cease to carry on research and/or educational activities; or

9.4.4. the Licensee is subject to an Insolvency Event.

9.5. Upon termination of this Licence by the Met Office pursuant to clause 9.2, 9.3 or 9.4, the Licensee shall remain liable to pay any expenses the Met Office may have incurred or have agreed to incur in connection with this Licence.

9.6. On termination or expiry of the Licence (however arising):

9.6.1. the Licensee shall either return or destroy the Software and its accompanying documentation and shall erase all copies of the Software under his/her control and stored in any medium;

9.6.2. all rights granted to the Licensee under this Licence shall cease and the Licensee shall cease all activities authorised by this Licence; and

9.6.3. the accrued rights and liabilities of the parties as at termination and the continuation of any provision expressly stated to survive or implicitly surviving termination, shall not be affected.

9.7. On termination of the Licence (however arising), the following clauses shall survive and continue in full force and effect; clause 3, clause 5, clause 6, clause 7, clause 8, clause 9, clause 10, clause 11, clause 14, clause 15, clause 16, clause 17, clause 19 and clause 21.

10. Freedom of Information

10.1. The Licensee acknowledges that the Met Office is subject to the requirements of the Freedom of Information Act 2000, the Environmental Information Regulations 2004 and the Reuse of Public Sector Information Regulations 2005 (together the "Disclosure Legislation") and shall assist and co-operate with the Met Office to enable the Met Office to comply with the Disclosure Legislation and any requests which reference the Disclosure Legislation ("Requests").

10.2. The Licensee shall:

10.2.1. transfer each relevant Request to the Met Office as soon as practicable after receipt and in any event within five (5) days of receiving the relevant Request;

10.2.2. provide the Met Office with a copy of all information which is available to it in the form that the Met Office requires within five (5) days (or such other period as the Met Office may specify) of the Met Office requesting the relevant information; and

10.2.3. provide all necessary assistance requested by the Met Office to enable the Met Office to respond to a Request within the time periods set out in the Disclosure Legislation.

10.3. The Met Office shall be responsible for determining in its absolute discretion whether the information:

10.3.1. is exempt from disclosure in accordance with the provisions of the Disclosure Legislation; and

10.3.2. is to be disclosed in response to a Request,

and in no event shall the Licensee respond directly to a Request unless it is expressly authorised to do so by the Met Office.

10.4. The Licensee acknowledges that the Met Office may be obliged under the Disclosure Legislation to disclose information:

10.4.1. without consulting with the Licensee; or

10.4.2. following consultation with the Licensee and having taken its views into account.

11. Data Protection

11.1. The Licensee authorises the Met Office to retain and process personal data provided by the Licensee to the Met Office in connection with the grant of this Licence provided that the Met Office processes that personal data in accordance with the Data Protection Act 1998.

11.2. For the purposes of this clause 11, the term 'personal data' will have the meaning given to it in the Data Protection Act 1998.

12. Force majeure

12.1. The Met Office shall have no liability to the Licensee under the Licence if it is prevented from, or delayed in performing, its obligations under the Licence or from carrying on its business by acts, events, omissions or accidents beyond its reasonable control ("Force Majeure"), including (without limitation) strikes, lock-outs or other industrial disputes (whether involving the workforce of the Met Office or any other party), failure of a utility service or transport network, act of God, war, riot, civil commotion, malicious damage, sabotage, epidemics, compliance with any law or governmental order, rule, regulation or direction, accident, breakdown of plant or machinery, significant power outages and/or IT failures, fire, flood, storm or default of suppliers or subcontractors.

12.2. Where delay has arisen due to Force Majeure, the date on which the Met Office's obligations are to be fulfilled shall be extended for a period of time equal to the time lost.

13. Variation

No variation of the Licence shall be valid unless it is in writing and signed by or on behalf of each of the parties.

14. Waiver

14.1. A waiver of any right under the Licence is only effective if it is in writing and it applies only to the circumstances for which it is given. No failure or delay by a party in exercising any right or remedy under the Licence or by law shall constitute a waiver of that (or any other) right or remedy, nor preclude or restrict its further exercise. No single or partial exercise of such right or remedy shall preclude or restrict the further exercise of that (or any other) right or remedy.

14.2. Unless specifically provided otherwise, rights arising under the Licence are cumulative and do not exclude rights provided by law.

15. Severance

15.1. If any provision of the Licence (or part of any provision) is found by any court or other authority of competent jurisdiction to be invalid, illegal or unenforceable, that provision or part-provision shall, to the extent required, be deemed not to form part of the Licence, and the validity and enforceability of the other provisions of the Licence shall not be affected.

15.2. If a provision of the Licence (or part of any provision) is found illegal, invalid or unenforceable, the provision shall apply with the minimum modification necessary to make it legal, valid and enforceable.

16. Entire agreement

16.1. The Licence constitutes the entire understanding and agreement between the parties in connection with and about the subject matter of the Licence and supersedes all earlier and other understandings and agreements between the parties and all earlier representations by any party about such subject matter.

16.2. Each party warrants that they have not entered into the Licence in reliance upon any representation, warranty, promise, term, condition, obligation or statement which is not expressly set out in the Licence. If a party has given any representation, warranty, promise, or statement then (except to the extent that it has been set out in the Licence), the party to whom it is given hereby waives any rights or remedies which it may have in respect of it.

16.3. Nothing in this clause shall limit or exclude any liability for fraud.

17. Assignment

17.1. The Licensee shall not, without the prior written consent of the Met Office, assign the Licence or all or any of its rights or obligations under the Licence to any third parties.

17.2. The Met Office may at any time assign all or any part of its rights and benefits under the Licence to a third party.

17.3. Each party that has rights under the Licence is acting on its own behalf and not for the benefit of another person.

18. Relationship of the parties

Nothing in the Licence shall constitute, or be deemed to constitute, a partnership between the parties nor, except as expressly provided, shall it constitute, or be deemed to constitute, either party the agent of the other for any purpose.

19. Rights of third parties

Except as expressly set out in the Licence, no person who is not a party to the Licence shall have any rights under the Contracts (Rights of Third Parties) Act 1999 to enforce any term of the Licence.

20. Notices

20.1. Any notice (which term shall in this clause include any other communication) required to be given under the Licence or in connection with the matters contemplated by it shall, except where otherwise specifically provided, be in writing in the English language.

20.2. Any notice to the Licensee and/or to the Met Office shall be addressed as set out in the Licence Particulars.

20.3. Notices may be:

20.3.1. personally delivered, in which case it shall be deemed to have been given upon delivery at the relevant address if it is delivered not later than 17.00 hours on a Business Day, or, if it is delivered later than 17.00 hours on a Business Day or at any time on a day which is not a Business Day, at 08.00 hours on the next Business Day; or

20.3.2. if within the United Kingdom, sent by first class pre-paid post, in which case it shall be deemed to have been given two Business Days after the date of posting; or

20.3.3. if from or to any place outside the United Kingdom, sent by pre-paid airmail, or by air courier in which case it shall be deemed to have been given seven Business Days after the date of posting in the case of airmail or two Business Days after delivery to the courier, in the case of air courier; or

20.3.4. sent by electronic mail, in which case, it shall be deemed to be given when received but any notice despatched by electronic mail after 17.00 hours on any Business Day or at any time on a day which is not a Business Day shall be deemed to have been given at 08.00 on the next Business Day; or

20.3.5. sent by fax, in which case it shall be deemed to have been given at the time of transmission.

21. Governing law and jurisdiction

21.1. The Licence, and any dispute or claim arising out of or in connection with it or its subject matter or formation (including non-contractual disputes or claims), shall be governed by, and construed in accordance with, the law of England and Wales.

21.2. The parties irrevocably agree that the courts which are located in the United Kingdom shall have exclusive jurisdiction to settle any dispute or claim that arises out of, or in connection with, the Licence or its subject matter or formation (including non-contractual disputes or claims).