Energy-aware metrics for benchmarking heterogeneous systems

Simon McIntosh-Smith
Dept. of Computer Science
University of Bristol
Woodland Road, Bristol, BS8 1UB, UK
simonm@cs.bris.ac.uk

Terry Wilson
Dept. of Computer Science
University of Bristol
Woodland Road, Bristol, BS8 1UB, UK
terry.wilson@gmail.com

Jon Crisp
Department of Biochemistry
University of Bristol
Woodland Road, Bristol, BS8 1TD, UK
jccrisp@gmail.com

Amaurys Ávila Ibarra
Department of Biochemistry
University of Bristol
Woodland Road, Bristol, BS8 1TD, UK
amaurys.avilaibarra@bris.ac.uk

Richard B. Sessions
Department of Biochemistry
University of Bristol
Woodland Road, Bristol, BS8 1TD, UK
r.sessions@bris.ac.uk

ABSTRACT

With the advent of heterogeneous computing systems consisting of multi-core CPUs and many-core GPUs, robust methods are needed to facilitate fair benchmark comparisons between different systems. In this paper we present a benchmarking methodology for measuring a number of performance metrics for heterogeneous systems. Methods for comparing performance and energy efficiency are included. Consideration is given to further metrics, such as associated runnings costs and even carbon emissions. We give a case study for these metrics applied to BUDE, a molecular mechanics-based docking application that has been ported to OpenCL at the University of Bristol.

Keywords

Heterogeneous computing, GPU, benchmarking, energy aware software, OpenCL, molecular docking

1. INTRODUCTION

Several trends in semiconductor physics have combined with trends in media-rich applications to give rise to highly parallel computer architectures, the two most obvious examples being multi-core CPUs and the latest fully programmable many-core graphics processors (GPUs). Some of the major trends in semiconductor physics are familiar, such as the exponential increase in available transistors as described by Moore's Law [11]. But some trends are more recent. For example, per device power consumption is now bounded, where previously this did trend upwards, tracking Moore's Law until mid way through the first decade of the twenty first century [18, 19]. Similarly, in the past chip voltage has been steadily reducing with each silicon process generation. Reducing voltages has been one of the main mechanisms for keeping device power consumption in check, giving rise to today's chip voltages of 1.0V or lower. However voltage cannot be reduced forever, and indeed the closer the voltage gets to a threshold that is determined by the physics of CMOS-based transistors, the worse they behave. Currently this lower threshold is 0.7V, and so we are close to losing voltage as a power reduction mechanism too.

These clashing semiconductor physics trends mean we will see processors with greater parallelism and greater heterogeneity in the future, as microprocessor architects seek creative ways to harness the potential of many more transistors in ever harsher power consumption regimes. Heterogeneous, massively parallel processors are thus likely become ubiquitous. They will be used everywhere from HPC systems to the smartphones in our pockets. Today's GPUs are a signpost for what is to come, with hybrid CPU-GPU processors soon to become the norm. In HPC we will need to embrace this next major architectural paradigm in order to maximise the performance benefits we receive from future systems.

GPUs were the first massively parallel processor technologies to be explored for use in HPC. The term 'General-Purpose computation on Graphics Processing Units,' or GPGPU, was first coined by Mark Harris as early as 2002 [7]. By 2006 the first fully programmable GPUs emerged, with Nvidia's introduction of CUDA and its G80 architecture. Since then the floodgates have opened, with multiple competing GPU architectures from major vendors and even an emerging open standard for GPU programming, OpenCL [8].

2. RELATED WORK

GPUs are fast and low-cost, so it should be no wonder that they have already found their way into HPC systems, with many applications being ported to CUDA or more recently OpenCL. One of the earliest classes of applications to be successfully ported to GPUs is the class to which BUDE belongs: molecular mechanics codes. These N-body algorithms are potentially well suited to GPUs because of the massively parallel nature of the problems they are solving. Early examples of molecular mechanics codes ported to GPU-like architectures include GROMACS in 2005 [20], NAMD in 2007 [16], Folding@Home [13] and Amber [2].

In terms of molecular docking codes, Sukhwani and Herbordt at Boston University were among the first to adopt GPUs, porting the PIPER production-level docking code initially to FPGAs and more recently GPUs [17]. Very few other docking codes have been ported yet but we expect this to change rapidly over the next few years.

Measuring power efficiency has only recently received significant attention in HPC. The Top500 now records energy consumption for systems being listed [10], and the Green500 was created in 2007 to rank systems by energy efficiency [14]. The challenge for both the Top500 and Green500 is the accuracy and consistency of the power consumption measurements — verifying these is much more difficult than LINPACK performance, and differences in how power consumption is measured or even estimated can potentially have a large impact on the metric. The power consumption given for some systems in the Top500 is the maximum power rating while in other cases the figures are actual measurements during their Top500 LINPACK runs. Additionally there is some confusion regarding whether power consumption includes system cooling in some cases (for example cooling systems integrated into the racks).

One project attempting to make system power measurements more accurate and robust is PowerPack [5]. PowerPack combines hardware probes with a software monitoring system to give a detailed view of power consumption at a component level inside a node (CPU, memory, disk, mainboard, network etc). The PowerPack project is showing great promise but its adoption is being hampered by the need for fairly intrusive hardware modifications to achieve its per component level of accuracy. Future server hardware may include more built-in monitors making PowerPack's approach much more widely applicable.

3. BUDE: A MOLECULAR MECHANICS-BASED DOCKING ENGINE

At the University of Bristol we have been developing a molecular mechanics-based docking engine called BUDE since 2001 [6]. BUDE uses a molecular mechanics-like empirical free-energy forcefield to predict the binding energy of two molecules. We have recently used BUDE to design inhibitors of human elastase [3] which have the potential to become drugs for the treatment of emphysema. Designed compounds are ranked in terms of their predicted binding affinity using the Evolutionary Monte Carlo (EMC) search method. Promising ligand molecules are synthesised in the laboratory and their real binding affinities experimentally determined. In the docking procedure, BUDE is given two molecules, one protein representing a drug target, and one ligand (potential drug). These two molecules are manipulated to see how well they fit or 'dock' for different poses (different positions of the ligand relative to the protein). Figure 1 illustrates a successful docking operation.

Like many molecular mechanics-based codes, BUDE is an

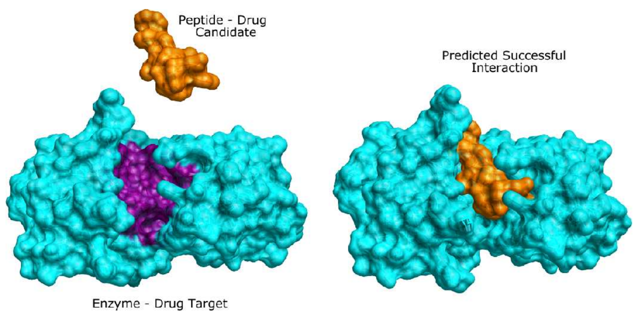


Figure 1: The molecular docking of an enzyme with a peptide.

ideal target for massively parallel implementations. BUDE has previously been ported to ClearSpeed systems, an early GPU-like accelerator developed specifically for HPC [9].

BUDE docking simulations may process an entire library of potential ligands which can contain millions of candidate molecules. Each of these ligands is docked with the protein in many different orientations or 'poses', where a pose describes a unique translation and rotation of the ligand in relation to the protein. Assuming modifications to pose rotation of 5° in each of the three spatial dimensions, there are 3.7×10^5 unique poses before translation is taken into account. Each pose is independent of all others, enabling poses to be calculated in parallel. BUDE is thus massively parallel in two dimensions: there are millions of independent ligands to be simulated, and each ligand has potentially hundreds of thousands of independent poses to test. BUDE does not search the entire solution space but instead its genetic algorithm-like EMC approach creates successive generations of candidate solutions from the best candidates from previous generations. After several such evolutionary phases BUDE typically finds answers very close to the optimal yet will have only searched around 1% of the total solution space.

There is one obvious further level of parallelism that can be exploited. Each docking operation is itself an N-body problem, where the energies of interaction between all atoms in the protein and all atoms in the ligand are calculated. The interaction energies between these atom pairs may be calculated in parallel, and indeed, this (and the corresponding force calculation) is the most common method of parallelism for molecular dynamics code ports to GPUs.

3.1 BUDE's Forcefield

BUDE describes the properties of the atoms of the 20 standard amino acids in a 'combined-atom' *forcefield*: each atom except for hydrogen (i.e. the heavy atoms) is modelled as a sphere with a specific radius and hydrophobicity or hydrophilicity potential, and hydrogen atoms are included in the volume of the heavy atoms they are attached to [15]. BUDE's forcefield parameters were collated empirically from experimental data, and are continually being modified and improved to predict accurate binding free-energies.

In order to calculate theoretical binding energies, BUDE uses these *forcefield* parameters and the equations described below (illustrated graphically in figure 2), which were modified from the *forcefield* developed for protein folding studies with the program RAFT [6]. The energy calculated by BUDE approximates to a free energy, as described by:

$$E_{complex} = E_{steric} + E_{electrostatic} + E_{desolvation} \tag{1}$$

E_{steric} is a repulsion caused by overlap of the spheres, $E_{electrostatic}$ is the electrostatic energy from charge-charge interactions and $E_{desolvation}$ is derived empirically for each amino acid from experimentally-determined solvation energies [21].

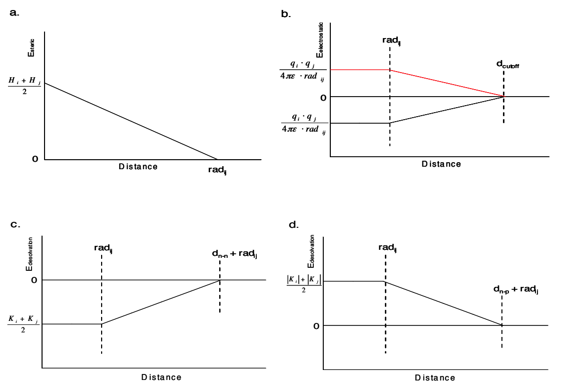


Figure 2: The functions used by BUDE to calculate theoretical energies. In all cases, rad_{ij} is the sum of the radii of atoms i and j. The x axis is the distance between the two atoms, $dist_{ij}$. In figure a), H is the hardness of each atom. In figure b), the red line illustrates the case where q_i and q_j are like charges and the black line illustrates the case where the charges are opposite. The distance after which no interaction is assumed to occur is d_{cutoff}, q is the charge on each atom and ε is taken to be 1. In figures c) and d), K is the desolvation potential of each atom. d_{n-n} or d_{n-p} is the distance over which the interaction is assumed to occur.

$$\text{When } dist_{ij} < rad_{ij}, \quad E_{steric} = \left(\frac{H_i + H_j}{2}\right)\left(1 - \frac{dist_{ij}}{rad_{ij}}\right) \tag{2a}$$

$$\text{When } dist_{ij} > rad_{ij}, \quad E_{steric} = 0 \tag{2b}$$

where $dist_{ij}$ is the distance between the two atoms, rad_{ij} is the sum of the radii of the atoms, and H_i and H_j are the hardness of the interacting atoms, as specified by the *forcefield* parameters.

When $dist_{ij} < rad_{ij}$,

$$E_{electrostatic} = \frac{q_i \cdot q_j}{4\pi\varepsilon \cdot rad_{ij}} \tag{3a}$$

When $rad_{ij} < dist_{ij} < d_{\text{cutoff}}$,

$$E_{electrostatic} = \left(\frac{q_i \cdot q_j}{4\pi\varepsilon \cdot rad_{ij}}\right)\left(1 - \frac{dist_{ij} - rad_{ij}}{d_{\text{cutoff}} - rad_{ij}}\right) \tag{3b}$$

When $dist_{ij} > d_{\text{cutoff}}$,

$$E_{electrostatic} = 0 \tag{3c}$$

where q_i and q_j are the charges on the two interacting atoms and the value of ε, the permittivity, is 1. d_{cutoff} is defined as 4 Å where the atom has a partial charge (i.e. for hydrogen bonding atoms) and 6 Å in the case of formal charge-charge interactions.

For polar – polar atom interactions,

$$E_{desolvation} = 0 \tag{4a}$$

For polar – non-polar or non-polar – polar interactions, when $dist_{ij} < rad_{ij}$,

$$E_{desolvation} = \frac{|K_i| + |K_j|}{2} \tag{4b}$$

where K_i and K_j are the values for desolvation potentials of the interacting atoms.

When $rad_{ij} < dist_{ij} < d_{n-p}$,

$$E_{desolvation} = \left(\frac{|K_i| + |K_j|}{2}\right)\left(1 - \frac{dist_{ij} - rad_{ij}}{d_{n-p}}\right) \tag{4c}$$

where d_{n-p} is the cutoff distance for non-polar – polar or polar – non-polar interactions.

When $dist_{ij} > d_{n-p}$,

$$E_{desolvation} = 0 \tag{4d}$$

For non-polar – non-polar interactions, when $dist_{ij} < rad_{ij}$,

$$E_{desolvation} = \frac{K_i + K_j}{2} \tag{4e}$$

When $rad_{ij} < dist_{ij} < d_{n-n} + rad_{ij}$,

$$E_{desolvation} = \left(\frac{K_i + K_j}{2}\right)\left(1 - \frac{dist_{ij} - rad_{ij}}{d_{n-n}}\right) \tag{4f}$$

where d_{n-n} is the cutoff distance for non-polar – non-polar interactions.

When $dist_{ij} > d_{n-n}$,

$$E_{desolvation} = 0 \tag{4g}$$

These calculations amount to a very computationally-intensive atom-atom inner loop for BUDE. Indeed this is more computationally-intensive than a typical N-body, atom-atom code, which helps to make BUDE even more suitable for parallelisation, effectively increasing the granularity of each independent task. Another useful characteristic of BUDE is its use of single precision floating point, making it suitable for most contemporary GPUs, even those which do net yet support double precision as fully as single precision.

Another important observation about this set of atom-atom calculations is that at first glance they contain a high degree of conditional execution dependant on the distances between the atom pair. Atoms closer together are treated differently to atoms further apart, as different forces come into play at different separation distances or with different kinds of atoms. On further examination some of the different distance cases contain common subexpressions which may be calculated just once for any atom-atom pairing, then modified further for a particular circumstance. We shall examine the implications of the remaining distance-dependant conditional execution on our data-parallel implementation in the following section.

3.2 A many-core parallelisation of BUDE using OpenCL

With a computationally-intensive atom-atom kernel and an abundance of ligand and pose-level parallelism, we were able to design a parallel version of BUDE that calculates many independent poses simultaneously, essentially running fast-yet-serial N-body kernels on each parallel processing element. We decided to use OpenCL for porting BUDE to GPUs for a number of reasons:

1. We wanted to port BUDE just once and then run the ported code on all potentially interesting GPUs for benchmarking purposes. While our initial hardware is from Nvidia we also wish to evaluate AMD's FireStream GPUs in due course.

2. One of the attractive features of OpenCL is the potential to run the same parallel code on multi-core host CPUs. BUDE has not yet been ported to OpenMP and so an OpenCL version using the host as the target device is of tremendous interest if the performance is good enough. Ultimately this will also enable us to make direct, like-for-like comparisons between GPUs and CPUs running the same OpenCL code.

3. OpenCL could also potentially support heterogeneous execution, running the OpenCL kernel on both GPUs and multi-core host CPUs at the same time. In this way we should achieve maximum aggregate performance, harnessing all available execution resources in a heterogeneous, multi- and many-core system.

BUDE's N-body kernel is computationally intensive, using an advanced empirical free energy forcefield, as described in the previous section. In addition, with protein molecules typically consisting of O(10^3) atoms and ligands typically O(10^2) atoms, the dataset for a real docking problem is fortuitously small. Indeed it is trivial to store all the necessary data for docking one ligand with one protein permanently either in the on-board memory of a GPU or in the on-chip cache of a CPU. Better yet, with a pose-parallel implementation, the data representing both the protein and the ligand is shared by all parallel processing elements (PEs). Unique to each PE will be a transformation matrix defining the pose to be tested on that PE. Transformation matrices are just twelve single precision floating point numbers that specify a translation and rotation in 3D space, and so once the protein and ligand atom information has been transferred to

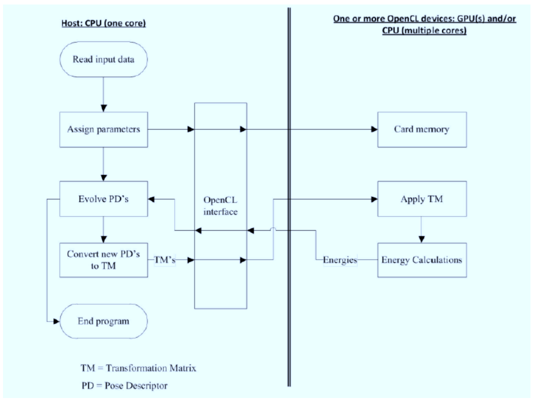


Figure 3: High-level description of BUDE

the OpenCL device in use, only $12 \times 4 = 48$ bytes of information has to be transferred per PE in order to initiate a significant amount of computation. Atoms are represented with just 40 bytes of information — three single precision floating point numbers representing the atom's (x, y, z) position in space, six single precision numbers representing the atom's radius, various charge parameters and so on, and a final four bytes of atom type information. Hence a simulation involves sending less than 50kBytes to the OpenCL device for the molecular information as a one off cost, and then one 48 byte transformation matrix per PE for each batch of pose calculations to be performed. For an OpenCL device with n PEs, $n \times 48$ bytes of transformation matrices will be transferred per batch – a trivial amount of data.

Data returned from the calculations is even more trivial. Each PE will be calculating a fitness function for how well its particular ligand pose has docked with the protein under consideration. This fitness value is ultimately a single 32-bit floating point number, and so for n PEs we will return just $4n$ bytes per batch of n poses that have been calculated in parallel.

4. BENCHMARKING METHODOLOGY

We had a number of systems we wished to test in terms of their performance and energy efficiency. The BUDE test problem we were using as a benchmark was for docking a 53 atom ligand to the 1,719 atom human prion protein [12]. A run of a single such BUDE simulation takes of the order of an hour on one core of a contemporary x86 processor. This is long enough to amortise any short duration noise and aid the repeatability of results. Nevertheless, we adopted a standard benchmarking approach of running every simulation five times, looking for any outliers, and taking the mean of the five runs on each platform.

We used the same FORTRAN host code on all platforms and the same OpenCL code across the range of systems that supported this programming language. We also used the same compilers wherever possible: gcc 4.3.3 and gfortran 3.4.6, both with -O3 optimisation levels selected. We used the latest Nvidia drivers on systems using their GPUs: NVIDIA UNIX x86_64 Kernel Module 260.24, released Thu Sep 9 17:01:12 PDT 2010. This corresponds to CUDA toolkit 3.1

and Nvidia's OpenCL release 1.1. We tried a number of experiments using OpenCL on the host CPU as an alternative to using OpenMP. For these experiments we used AMD's OpenCL SDK v2.2.

To ensure accuracy when measuring power consumption across the diverse platforms being measured, we decided to use a discrete power measuring device which would also allow us to measure power consumption 'at the wall' for each system. After considering several options we decided to use a 'Watt's Up? Pro' device [4]. This equipment measures total system power very accurately, to within ±1.5%, and records samples at a user-specified rate into an internal memory that can then be read back via a USB port for later analysis. Given that simulation runs were of the order of hours for single cores and minutes for GPUs we set the power consumption sampling rate for one second intervals. Being able to use the same measuring equipment for all the devices under test is an important element of this methodology, allowing us to guarantee that power consumption is being measured in a consistent fashion.

On multi-core CPUs we ran independent identical copies of the BUDE benchmark on each core, for example running eight simultaneous BUDE simulations on our eight core test machines. We would also run a single BUDE simulation on each GPU under test, for example running two simultaneous simulations on our twin GPU test machines. This allowed us to measure performance and power consumption on fully-loaded test machines. Given BUDE's massively parallel design and minimal data streaming requirements we would expect the simultaneous BUDE simulations to be very well behaved and exhibit close to linear speedup.

In addition to measuring performance and power consumption on the platforms under test we were interested in calculating carbon emissions per simulation. Carbon emissions are likely to be targeted for increased taxation in the UK and around the world in coming years. One may intuitively expect that carbon emissions are directly proportional to energy use, but this turns out not to be the case. Power generation is variable in terms of its carbon intensity, with the mix of energy sources such as coal and gas-fired power stations, nuclear and renewable, changing all the time, depending on the time of day, weather, demand etc. In the UK the iDEaS project at the University of Southampton delivers real-time information on energy carbon intensity for the UK's national grid [1]. We used this information to form estimates of carbon emissions per simulation and also to understand how this metric can vary over time. We believe these kinds of metrics — energy efficiency, cost per simulation, and carbon emissions per simulation — will become increasingly important in HPC over the coming decade.

4.1 Systems under test

The first system under test was a Supermicro 1U dual GPU server with two Intel 5500 series 2.4 GHz Xeon 'Nehalem' quad-core processors, 24 GBytes of DRAM and two PCI Express x16 gen 2 slots, each holding an Nvidia C2050 'Fermi' GPU. We benchmarked both the host CPUs and the Fermi GPUs in this system. This system is representative of the most popular high-end GPU-based servers being used in HPC today.

The second system under test was a workstation containing an Intel E8500 3.16 GHz dual core CPU and a previous generation Nvidia GPU in consumer form, the GTX280. We wanted to compare how well the new Fermi GPUs performed on the BUDE benchmark in relation to prior GPUs. This workstation is representative of a typical desk-side machine running a moderately up-to-date GPU for scientific experiments.

Because OpenCL also supports running on a multi-core host CPU we decided to try a workstation based on an AMD Phenom II X3 720 running at 2.8 GHz. This three core machine sported Ubuntu 10.04.1, had 4 GBytes of DRAM, and used AMD's Stream SDK v2.2. This system should be typical of the lower end of parallel machines, and so it is still useful and interesting to see what it can do in relation to the higher core count and GPU accelerated systems in this benchmarking comparison.

The final system under test was an Intel-based Core2Duo SU9400 'Penryn' 1.4 GHz laptop with 4 GBytes of DRAM. Sometimes BUDE can be used for small simulations on a researcher's personal computer, and with most laptops now being at least dual core, we wanted to see the effect of using OpenCL to run on both cores at once, compared to the single core the original FORTRAN code would use.

4.2 Problems encountered

During this work it became clear that OpenCL is still a maturing technology. Nvidia's drivers were very stable but we had a mixed experience with AMD's OpenCL SDK v2.2. We found that these drivers worked reasonably well targeting a host CPU, even Intel CPUs, but in a number of attempts to use AMD/ATI 4870 GPUs we would consistently suffer system crashes. In the end we gave up on trying to benchmark AMD GPUs with the current drivers and hope to return to these after their next software release and using more contemporary AMD GPUs.

5. RESULTS

First we had to form some measure of how optimal our OpenCL code was when running on a GPU. Using Nvidia's tools we were able to determine the 'utilisation' factor for BUDE's OpenCL kernel, which turned out to be 0.5 on the GTX280. We also recorded a branch divergence ratio of 300:1. Anecdotally, utilisations of 0.6–0.7 are good, so we were satisfied with this level of GPU efficiency.

Satisfied that our OpenCL code was reasonably optimal we moved on to measuring performance and energy efficiency across our range of test systems. The results using the representative high-end GPU server were about what we expected. The Fermi GPUs performed exceptionally well — when both GPUs were used at once they delivered 6.0X greater performance that the eight Nehalem cores in the same node. The other systems under test also performed roughly as one would expect. The GTX280 GPU performed very well, at about 40% slower than the Fermi-based C2050's. The three core AMD Phenom II X3720 also performed well at just 45% slower than the much more expensive and power hungry quad core Nehalems. As expected the laptop-optimized Core2Duo SU9400 is the slowest by far, but achieved a better than expected 2.3X speedup us-

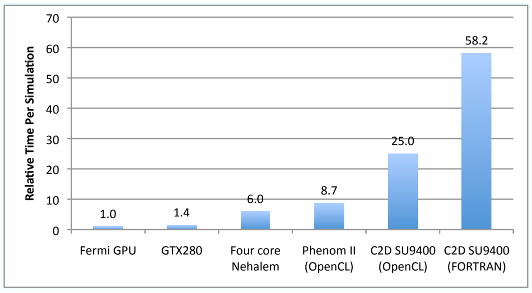


Figure 4: Normalised graph of time per simulation, lower is better.

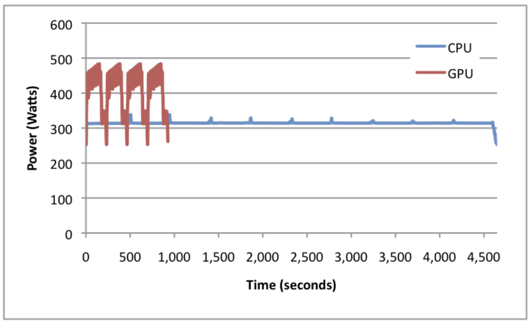


Figure 5: Power consumption profile for the dual Fermi GPUs compared to the eight Nehalem cores. Both runs complete eight BUDE simulations

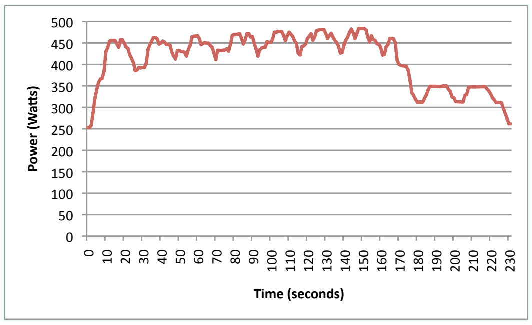


Figure 6: Power consumption profile for the dual Fermi GPU server to complete two BUDE simulations

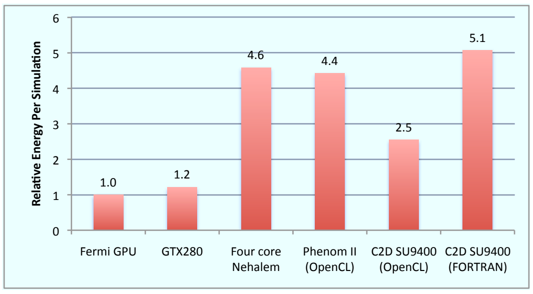


Figure 7: Normalised graph of energy used per simulation, lower is better.

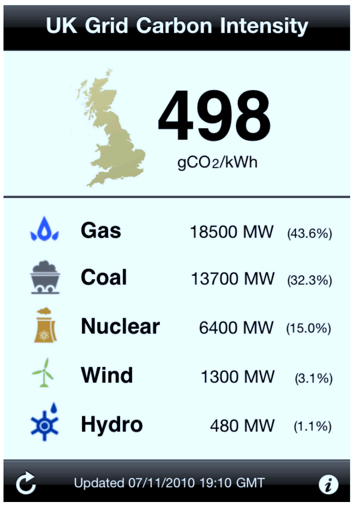
ing OpenCL on both cores compared to using the original FORTRAN code on just a single core. Figure 4 shows the normalised time per simulation of each system in the test.

In terms of energy efficiency, the dual Fermi GPU system would average 413.1W and consume 0.11kWh of energy to complete eight simulations. On the eight Nehalem cores the system would average 313.9W but would of course run for much longer, resulting in 0.40kWh of energy being used to complete the same eight simulations. Hence the GPUs were 3.6X better than the eight cores in terms of performance per watt. Figure 5 shows the power consumption profile of the system in the two different cases: using both Fermi GPUs and using all eight Nehalem cores. In both cases a total of eight BUDE simulations were completed.

It is worth breaking out the GPU power profile and zooming in to a single pair of BUDE simulations, see figure 6. The profile is slightly noisy because both GPUs are being used in parallel and the simultaneous BUDE simulations are slightly offset. However, there are some distinct features which we can correlate to our understanding of BUDE's execution on the GPUs. The peaks correspond to the GPUs running the atom-atom kernel, while the troughs correspond to the host cores being used to generate new populations of potential ligands based on the best ligands from the previous population. At the end of the simulations there is a results gathering and reporting phase that currently runs only on the host cores and so the GPUs are unused for a period of time before BUDE completes.

Comparing the relative energy used per simulation the results take on a different shape. As expected the GPUs do very well. But relatively speaking the AMD Phenom II X3720 and Intel Core2Duo SU9400 do well, posting similar energy efficiencies to the Intel Nehalem system. We should note that of course the Nehalem system is a high-end server optimized primarily for performance, whereas the Phenom and Core2Duo systems under test were workstation and laptop systems where power consumption has been more of a focus. Figure 7 shows the normalised energy used per simulation for each system in the test.

There is one important caveat to these result which is that the Fermi GPUs are known to have quite a high idle power draw which would be negatively impacting the Nehalem CPU results. We did not have time to uninstall the GPUs from the system and rerun the CPU results but this could


UK Grid Carbon Intensity

498

gCO2/kWh

Gas	18500 MW	(43.6%)
Coal	13700 MW	(32.3%)
Nuclear	6400 MW	(15.0%)
Wind	1300 MW	(3.1%)
Hydro	480 MW	(1.1%)

Updated 07/11/2010 19:10 GMT

Figure 8: An instantaneous measurement of the UK national grid's carbon intensity

have yielded a non-trivial improvement to the performance per watt results of the CPUs. Conversely, the host CPUs would be consuming idle power while the OpenCL code was running on the GPUs, though this should be a smaller effect. This is an issue we would hope to address in future work.

In the one instance where we were able to measure OpenCL performance using the host CPU as the OpenCL target we saw surprisingly good results. The original FORTRAN code for BUDE would only use one host core by default, but running the OpenCL code on the C2D SU9400 we saw a speedup of 2.3X, more than the theoretical maximum 2X we were aiming for. We are slightly suspicious of this result and investigations are continuing, but there is one potential explanation: BUDE is very vectorizable, and it is possible that the OpenCL code on the host is able to exploit the SSE SIMD instruction set in the x86 architecture. We have yet to confirm if this is actually what is happening here.

From these results we can also consider the carbon emissions associated with each simulation. Average carbon emissions for electricity generation for the UK's national grid are around 500g of CO_2 per kWh, but this can vary depending on the time of day and the prevailing weather. For example, between the hours of 9am and 6pm on a week day electricity demand is typically at its peak with a correspondingly high carbon intensity. But during hours of lower demand, the mix of power generation can be up to 25% more carbon efficient, especially in the small hours of the night, for example between 3am and 6am.

Using data from the iDEaS project at the University of Southampton [1] it is possible to see this effect over a 24 hour window. The graph in figure 8 shows an instantaneous view of the mix of gas, coal, nuclear, wind and hydro power generation for the UK's national grid (the UK was consuming a total of 40.4 GigaWatts with an average carbon intensity of 498gCO_2/kWh). Figure 9 shows how the total power demand changed over a 24 hour period, and how

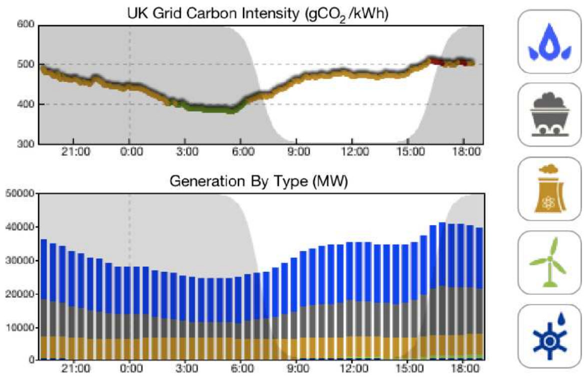


Figure 9: A 24-hour view of the carbon intensity of power generation in the UK

this affected the mix of the different sources.

6. FUTURE WORK

We are now in a strong position to benchmark our portable BUDE OpenCL code on a wider range of systems. We hope to test some of AMD's latest GPUs in the near future, and also to measure OpenCL performance on the Nehalem CPUs within our Fermi GPU cluster.

BUDE should scale extremely well to many GPUs, and so we would like to benchmark Dell's new dense GPU chassis, the 3U PowerEdge C410x which can accomodate up to 16 GPUs.

BUDE's OpenCL code is already quite optimal but we know there is room for improvement. In particular BUDE only makes 32-bit memory accesses when there is potential for it to load up to 128-bits at a time, improving load/store efficiency on most GPU architectures. It should also be possible to make use of OpenCL's implicit vector datatypes to unroll BUDE's inner loop four times. This should improve performance on AMD GPUs and perhaps aid vectorization on x86-SSE architectures. With the GPU code now running so fast, what little remains on the host is becoming the bottleneck. We will investigate porting some of the remaining functionality into OpenCL to mitigate this pinch point.

Finally we would also hope to try running BUDE heterogeneously using both the host cores and multiple GPUs to run the OpenCL kernel all at the same time. This arrangement should yield the maximum possible system performance and energy efficiency. Further development work on BUDE will be required to make this possible.

7. CONCLUSIONS

We have shown that real applications can be ported to OpenCL and achieve impressive performance gains on the latest GPUs and even on multi-core CPUs. OpenCL is still a maturing technology but is rapidly gaining ground as it enables applications to be ported once and run on many platforms. Molecular docking applications of the class represented by BUDE are particularly well suited to parallel implementations in general, and to GPUs in particular. And GPUs are able to deliver not just the best performance for our application, but also the most energy efficient execution. As docking applications evolve to use more computationally-intensive atom-atom kernels and searching more poses and potential ligands, they should become even more ideal matches to many-core architectures. We have demonstrated that it is possible to accurately compare energy efficiency between different systems with careful selection of the test equipment. We have shown how new metrics for per simulation energy consumption, cost and even carbon emissions are now possible and yield valuable insight. We believe these energy, running-cost and carbon-aware metrics will become increasingly important over the coming decade in HPC.

8. ACKNOWLEDGMENTS

The authors would like to thank the Advanced Computing Research Centre at the University of Bristol for providing access to the GPU servers in the Blue Crystal supercomputer and for supporting our work.

9. REFERENCES

[1] http://www.ideasproject.info/.

[2] D. Case, I. T.E. Cheatham, T. Darden, H. Gohlke, R. Luo, J. K.M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. Woods. The Amber biomolecular simulation programs. *J. Computat. Chem.*, 26:1668–1688, 2005.

[3] J. Crisp. *Design, Synthesis and Evaluation of peptide-based protease inhibitors*. PhD thesis, University of Bristol, 2009.

[4] Electronic Educational Devices. Watts up? Pro datasheet.

[5] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. Cameron. PowerPack: Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel and Distributed Systems*, 21(5):658–671, 2010.

[6] N. Gibbs, A. R. Clarke, and R. B. Sessions. Ab initio protein structure prediction using physicochemical potentials and a simplified off-lattice model. *Proteins: Structure, Function, and Bioinformatics*, 43(2):186–202, 2001.

[7] M. Harris. http://gpgpu.org/about, 2002.

[8] Khronos Group. OpenCL - the open standard for parallel programming of heterogeneous systems, November 2010.

[9] S. McIntosh-Smith and R. B. Sessions. An accelerated, computer assisted molecular modeling method for drug design. In *International Supercomputing*, June 2008.

[10] H. Meuer. http://www.top500.org/.

[11] G. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, April 1965.

[12] A. Nicoll, C. Trevitt, M. Tattum, E. Risse, E. Quarterman, A. Ibarra, C. Wright, G. Jackson, R. Sessions, M. Farrow, J. Waltho, A. Clarke, and J. Collinge. Pharmacological chaperone for the structured domain of human prion protein. In *Proc. Natl. Acad. Sci. U. S. A.*, pages 17610–17615, 2010.

[13] V. Pande, I. Baker, J. Chapman, S. Elmer, S. Khaliq, S. Larson, Y. Rhee, M. Shirts, C. Snow, E. Sorin, and B. Zagrovic. Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. *Biopolymers*, 68(1):91—-109, 2003.

[14] S. Sharma, C. hsing Hsu, and W. chun Feng. Making a case for a green500 list. In *Proc. of the Workshop on High-Performance, Power-Aware Computing*, 2006.

[15] S. Smith. *Cyclic peptides as protease inhibitors: development of a combinatorial library approach guided by computational modelling*. PhD thesis, University of Bristol, 2005.

[16] J. E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco, and K. Schulten. Accelerating molecular modeling applications with graphics processors. *Journal of Computational Chemistry*, 28(16):2618–2640, Sep 2007.

[17] B. Sukhwani and M. C. Herbordt. GPU acceleration of a production molecular docking code. In *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, GPGPU-2, pages 19–27, New York, NY, USA, 2009. ACM.

[18] H. Sutter. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's Journal*, 30(3), March 2005.

[19] H. Sutter. The free lunch is over online: http://www.gotw.ca/publications/concurrency-ddj.htm, August 2009.

[20] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. C. Berendsen. GROMACS: Fast, flexible, and free. *Journal of Computational Chemistry*, 26(16):1701–1718, Oct 2005.

[21] R. Wolfenden, L. Andersson, P. M. Cullis, and C. C. B. Southgate. Affinities of amino-acid side-chains for solvent water. *Biochemistry*, 20(4):849–855, 1981.