Some of the *What?, Why?, How?, Who?* and *Where?* of Graphics Processing Unit Computing for Bayesian Analysis

(@March 15th 2010 – its a moving target!)

Marc A. Suchard (UCLA), Chris Holmes (University of Oxford)
&
Mike West (Duke University)

Over the last 20 years or so, a number of Bayesian researchers and groups have invested a good deal of time, effort and money in parallel computing for Bayesian analysis. The growth of "small research group" to "institutionally supported" cluster computational facilities has had a substantial impact on a number of areas of Bayesian analysis, enabling analyses that are otherwise practically infeasible. Parallel computing has also motivated new approaches to simulation and optimisation-based Bayesian computations that aim to maximally exploit the "master-slave" and "embarrassingly parallel" computational model [e.g., 3, 4, 6]. In more recent years, increasingly prevalent multi-core CPUs in standard servers, desktops and laptops have engendered some interest in relatively simple and easy multi-threading of existing Bayesian analysis code, whether implemented in low level languages (C/C++) or through parallelisation facilities in environments such as R and Matlab®. Much progress in research and in advancing the use of Bayesian methods in increasingly computationally challenging problems has resulted. As we look ahead, however, the potential impact of parallel computation on both immediate research and the development of more broadly useful software is clearly – to some of us – *dramatically* enhanced by the advent of scientific computation using desktop and laptop graphical processing units (GPUs). Major new opportunities for orders-of-magnitude speed-up in computation are emerging through GPU programming, and the technology is cheap, both to purchase and run, and easily available.

We have been exploring these opportunities and developing a base of experience and examples in Bayesian analysis that bear out this "opportunistic" view; each of us is now firmly committed to exploiting GPU computation as a norm in our research. Current directions for technological developments include emerging-technology GPUs that are squarely aimed at the scientific computing community, clusters of GPUs, and integration of GPU and CPU processing in increasingly nimble commodity machines that enable massive parallelisation on the desktop. This will be a big part of the compute environment for us all in a short few years, and it seems clear (again to us!) that this technology promises to impact far more profoundly on statistical computation and software development than cluster facilities for parallelisation ever have or are ever likely to.

Some of these experiences and examples may be of broader interest to the Bayesian communities and ISBA members, in particular.

What?

GPUs are dedicated numerical processors designed for rendering 3-dimensional computer graphics. They are the graphics card "engines" in high-end graphics computers and gaming machines. In essence, a GPU consists of hundreds of processor cores on a single chip, and each core can be programmed to apply the same numerical operations simultaneously to each element of large data arrays – the so-called single instruction, multiple data (SIMD) paradigm. Since the same operations (called "kernels") function simultaneously, GPUs can achieve extremely high arithmetic intensity so long as we can enable sufficiently fast and efficient transfer of required in-

put data "onto" the processers and, correspondingly, of the ouput data "off" the processors.

As an extension to common programming languages, CUDA [7, 9] opens up the GPU to general purpose computing, and the computational power of these units has increased to the stage where they can process data intensive problems many orders of magnitude faster than conventional CPUs. The development of open-source libraries (OpenCL: *www.khronos.org/opencl*) is advancing, and in the coming year or two can be expected to simply mushroom as broader ranges of computational scientists press for, and themselves develop, supporting software tools.

GPUs are graphics processing work-horses in many standard desktop and laptop computers, as well as high-end graphical workstations. Among manufacturers, the NVIDIA corporation is way ahead in technology and in addressing the increasing interest for scientific computation. The current NVIDIA GPUs include GTX and Tesla varieties; these are inexpensive, commodity parallel machines that can be installed – singly or in small multiples – in many desktops and laptops, as well as in small cluster arrangements. NVIDIA's next-generation (2010 release expected) Fermi GPU promises substantial increases in numbers of cores, in processing speed per core, in on-card memory shared by the hundreds of cores for fast data access, input and output, as well being more heavily targeted towards computational uses in addition to graphics.

Why?

For scientific computing, GPU utility emerges when computations are inherently "massively parallel," i.e., the computation can exploit parallelization across many (hundreds or thousands of) GPU cores simultaneously. This structure emerges in many statistical models in Bayesian analysis, while in others we may capitalize on GPU architecture with appropriately restyled computational strategies. Among our own interests has been developing effective code for Bayesian mixture models for data in several tens of dimensions, with hundreds of mixture components and with large data sets – millions to tens of millions of observations. In such contexts, MCMC or posterior mode search computations are intensive, but massively dominated by the within-iterate (whether MCMC or EM, or other) calculations. In these *"moderate-to-large p, large k, very large n"* problems, massive fragmentation of calculations induced by conditional independencies are ideally suited to GPU parallelization. GPU machines have potential to define major speed-up – on cheaply and easily accessible hardware – for these computations as a routine, and the potential is realised; some of our recent examples show that first-version implementations of MCMC and Bayesian EM in standard Bayesian mixture models enables scale-ups on desktop personal computers that are simply not achievable using multi-threaded CPU desktops and simply impractical across distributed-memory computing clusters. Scale-ups of 100-fold in raw processing time are dramatic in terms of the ability to run analyses routinely, and – as this typical benchmark uses just first-generation GPUs and supporting software tools – this is just the start. Recent advanced Monte-Carlo methods such as population-MCMC, SMC samplers and particle-MCMC [e.g. 1] are naturally aligned to GPU computation, and effective parallel implementations allow us to realize their advantages in terms of increased mixing and exploration of high-dimensional and complex target densities for little overheard.

Novel, GPU-oriented approaches to modifying existing algorithms and software design not only provide the opportunity for vast speed-up, however; critically, they also enable statistical analyses that presently will simply not be considered due to compute time and other limitations in traditional computational environments. In one of our motivating application areas for *"massive mixtures"*, that of routine analyses of many, very large data sets in experimental biology studies [2], laboratory culture is hugely resistant to the notion of accessing institutional clusters – for reasons of cost and access, and also due to the norms and established practice of "computing in the lab." Software for GPU-enabled desktops will provide the opportunity

for complex, practically relevant Bayesian analyses to move more aggressively into practice as a result.

How?

GPU programming is fundamentally an exercise in parallel programming. As such, key concepts include those of clearly and explicitly identifying the major, core compute demands of any specific model analysis, and the inherent "bottlenecks" that limit the ability to achieve efficiencies via parallelization. MCMC algorithms, for example, are intrinsically *serial* algorithms, but can still benefit (potentially massively in large-scale, highly structured problems) from GPU parallelisation if the "per iterate" computations can be parallelised. That GPU cores share memory on the unit means that data stored "locally" can be quickly and efficiently accessed, so consideration must also be given to the basic programming issues of simply *moving data around*.

For NVIDIA GPUs, CUDA is a parallel computing technology, and programming language, that enables access to GPU computing via modifications of standard computing (in C/C++). OpenCL is an emerging library that provides access to GPUs from several hardware providers. Both require low-level programming for which researchers new to GPU programming should develop basic facility. On the near horizon stand higher-level, flexible interfaces for GPU programming, such as Thrust (*code.google.com/p/thrust*) that provides many standard parallel algorithms in an easier-to-use form than CUDA. We currently recommend prototyping using Thrust and then re-implementing critical code parts in CUDA for raw speed and performance.

To capitalise on the opportunities for Bayesian computations that are offered by GPUs – in the near term – requires some investment of time and effort (though limited money!) to adapt standard code to the GPU environment. This involves relatively modest changes in programming perspectives and strategy for algorithm implementation, and can rely on the already established base of experience in a number of groups. The required investment in developing programming skills certainly challenges researchers for whom low-level programming has never been a focus, However, for researchers and statistical programmers aiming to transform the efficiency, potential impact and broader use of Bayesian models and methods, the investment will be extremely worthwhile. As a rough guide we estimate that a programmer proficient in a language such as C or C++ should be comfortable with programming in CUDA within around 4 weeks.

Our own groups have defined a base of experiences that others may find useful – both as entry points for perspective, and possibly also in terms of access to existing code and examples. For example, we have defined some initial code and manuscripts with overtly "tutorial" flavour, focusing on some more-or-less standard Bayesian model contexts that will be of broad use and appeal. We have developed these to show the flow of the analyses so as to engage researchers that may be interested in developing in this direction in related or other classes of statistical models.

Who & Where?

Some links to groups involved in GPU computation for statistical research and application generally, as well as specific groups and projects in Bayesian analysis that provide resources – papers, examples, software – include those noted and linked below, among others. The current authors are committed to working towards the merging of their own sites to provide a central resource for the field.

- `www.stat.duke.edu/gpustatsci`
  Massively parallel GPU-based computing for statistical science, and Bayesian analysis broadly, at Duke University, headed by Mike West. This site links to articles and software with a focus on GPU in Bayesian analysis broadly. Readers can find there the tutorial-level *"How to?"* material, as well as code, for efficient GPU analysis of Bayesian mixture models described in [12]. The site is developing

to include additional GPU software for a range of complex Bayesian models (e.g., for the large-scale, spatial (and 3D spatio-temporal) models of [5]) and to add links and resources of broader interest to the Bayesian community.

- www.oxford-man.ox.ac.uk/gpuss/
  Bayesian GPUSS, the GPU stochastic simulation group at Oxford University, headed by Chris Holmes. Readers can find links to current research papers and online resources as well as a series of tutorial papers and coded examples, including the overview paper on advanced Monte Carlo methods for Bayesian inference [8]. The site aims to maintain a list of papers published in the field relevant to statistical computation using GPUs.

- www.biomath.ucla.edu/msuchard/
  Marc Suchard's group at UCLA. This site provides access to some of the first detailed GPU work in Bayesian analysis in challenging problems in computational biology and high-dimensional optiminization [10, 11, 14].

- brainarray.mbni.med.umich.edu/Brainarray/rgpgpu/:
  R+GPU. This R package off-loads several common data-summary tools from R to the GPU.

- www.stat.psu.edu/~mharan/
  Murali Haran's group at PSU. This site provides a GPU example involving slice sampling [13].

- gpgpu.org/
  The community site, resource and networking arena for researchers interested in *General-Purpose computation on Graphics Processing Units*.

- www.nvidia.com/page/home.html
  www.nvidia.com/object/cuda_home_new.html
  The NVIDIA and NVIDIA/CUDA site for CUDA progamming of GPUs; keep up with the latest technology and software advances, and access scientific computing networks.

- www.khronos.org/news/C124/
  The Kronos Group web site with information and resources related to the OpenCL library that brings general purpose GPU programming to a variety of hardware platforms.

- code.google.com/p/thrust/
  Thrust, Code at the Speed of Light. Thrust is a CUDA library of parallel algorithms with a high-level interface to enhance developer productivity.

References

[1] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo (with discussion). *J. Royal Statistical Soc. B*, to appear, 2010.

[2] C. Chan, F. Feng, M. West, and T.K. Kepler. Statistical mixture modelling for cell subtype identification in flow cytometry. *Cytometry, A*, 73:693–701, 2008.

[3] C. Hans, A. Dobra, and M. West. Shotgun stochastic search in regression with many predictors. *Journal of the American Statistical Association*, 102:507–516, 2007.

[4] C. Hans, Q. Wang, A. Dobra, and M. West. SSS: High-dimensional Bayesian regression model search. *Bulletin of the International Society for Bayesian Analysis*, 24:8–9, 2007.

[5] C. Ji, D. Merl, T.B. Kepler, and M. West. Spatial mixture modelling for unobserved point processes: Application to immunofluorescence histology. *Bayesian Analysis*, 4:297–316, 2009.

[6] B. Jones, A. Dobra, C.M. Carvalho, C. Hans, C. Carter, and M. West. Experiments in stochastic computation for high-

dimensional graphical models. *Statistical Science*, 20:388–400, 2005.

[7] D. Kirk and W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan-Kaufman, 2009.

[8] A. Lee, C. Yan, M.B. Giles, A. Doucet, and C.C. Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. Technical report, http://arxiv.org/abs/0905.2441, 2009.

[9] NVIDIA-CUDA. Nvidia cuda compute unified device architecture: Programming guide version 2.0. http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf, 2008.

[10] M.A. Suchard and A. Rambaut. BEAGLE: broad-Platform Evolutionary Analysis General Likelihood Evaluator. http://beagle-lib.googlecode.com, 2009.

[11] M.A. Suchard and A. Rambaut. Many-core algorithms for statistical phylogenetics. *Bioinformatics*, 25:1370–1376, 2009.

[12] M.A. Suchard, Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West. Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics*, to appear:–, 2010.

[13] M.M. Tibbits, M. Haran, and J.C. Liechty. Parallel multivariate slice sampling. Technical report, Department of Statistics, PSU, 2009.

[14] H. Zhou, K.L. Lange, and M.A. Suchard. Graphical processing units and high-dimensional optimization. Technical report, Department of Human Genetics, UCLA, 2009.