Computational Speedup in Spatial Bayesian Image Modeling via GPU Computing

Timothy D. Johnson

Department of Biostatistics
University of Michigan

Outline

Problem 1: Group fMRI Analysis

- Hierarchical Bayesian model [1]
- Level 1 — Likelihood
  - separate model for each subject (typically 10 - 20)
  - image intensities — finite mixture with spatially correlated weights
  - several components required to model each activation region
  - component means — finite mixture about activation centers
- Level 2
  - Activation centers — finite mixture model about population centers
- Level 3
  - Population centers modeled as a homogeneous Poisson process

I will concentrate on Level 1 as it is the most computationally intense

[1] Xu, Johnson, Nichols, Nee. *Biometrics 2009*

Problem 1: Group fMRI Analysis

- y_v — intensity at voxel v. Approximately 250K voxels
- x_v — (x, y, z) spatial location
- $\phi(y; m, v)$ — Gaussian density w/mean m and variance v

Likelihood and weights

All voxels assumed to be conditionally independent

$$\pi(y_v \mid \cdot) = w_0 \phi(y_v; 0, 1) + \sum_{j=1}^{J} w_j \phi(y_v; \theta_j, \sigma_j^2)$$

$$\begin{aligned} w_0 &\propto m \\ w_j &= \frac{\phi_3(x_v; \eta_j, \Psi_j)}{m + \sum_{k=1}^{J} \phi_3(x_v; \eta_k, \Psi_k)} \end{aligned}$$

where $\phi_3(x; \eta, \Psi)$ is the Gaussian density in $\mathbb{R}^3$ at x

Problem 2: Modeling of MS Lesion Location

Definition

Suppose that $Y = \{Y(\xi) : \xi \in S\}$ is a non-negative random field such that with probability one, $\xi \to Y(\xi)$ is a locally integrable function. If $[X \mid Y] \sim PP(S, Y)$, then X is said to be a *Cox process* driven by Y.

Density

The density of $X_B = X \cap B$, $B \subset S$, w.r.t. the dist. of a unit-rate Poisson process, is given by

$$\pi(x) = \mathbb{E}\left[\exp\left(|B| - \int_B Y(\xi)d\xi\right)\prod_{\xi \in x} Y(\xi)\right].$$

Furthermore, the intensity function is given by $\rho(\xi) = \mathbb{E}[Y(\xi)]$.

Problem 2: Modeling of MS Lesion Location

Definition

If $Y(\xi)$ is a Gaussian Process, then the Cox process driven by $\exp[Y(\xi)]$ is said to be a Log Gaussian Cox Process.

- The dist of (X, Y) completely determined by

$$m(\xi) = \mathbb{E}(Y(\xi)) \qquad c(\xi, \eta) = Cov(Y(\xi), Y(\eta)).$$

- We will assume $c(\xi, \eta)$ is translation invariant and isotropic

$$c(||\xi - \eta||) = \sigma^2 r(\alpha||\xi - \eta||).$$

- A useful family of corr functions is the *power exponential family*:

$$r(\alpha||\xi - \eta||) = \exp(-\alpha||\xi - \eta||^\delta).$$

Problem 2: Modeling of MS Lesion Location

- Model the locations of Multiple Sclerosis lesions in 217 subjects

Problem 2: Modeling of MS Lesion Location

- Model the locations of Multiple Sclerosis lesions in 217 subjects
- Intensity function $\lambda(x_v) = \exp[Y(x_v))]$
  - $Y(\cdot)$ is a Gaussian process in $\mathcal{B} \subset \mathbb{R}^3$
  - mean of process is μ
  - stationary correlation function $r(||x - z||) = \sigma^2 \exp(-\rho||x - z||^\delta)$.

Problem 2: Modeling of MS Lesion Location

- Model the locations of Multiple Sclerosis lesions in 217 subjects
- Intensity function $\lambda(x_v) = \exp[Y(x_v))]$
  - $Y(\cdot)$ is a Gaussian process in $\mathcal{B} \subset \mathbb{R}^3$
  - mean of process is μ
  - stationary correlation function $r(||x - z||) = \sigma^2 \exp(-\rho||x - z||^\delta)$.
- $\mathcal{B}$ is discretized on a grid ($91 \times 109 \times 91$)

Problem 2: Modeling of MS Lesion Location

- Model the locations of Multiple Sclerosis lesions in 217 subjects
- Intensity function $\lambda(x_v) = \exp[Y(x_v))]$
  - $Y(\cdot)$ is a Gaussian process in $\mathcal{B} \subset \mathbb{R}^3$
  - mean of process is μ
  - stationary correlation function $r(||x - z||) = \sigma^2 \exp(-\rho||x - z||^\delta)$.
- $\mathcal{B}$ is discretized on a grid $(91 \times 109 \times 91)$
- $Y(\cdot)$ is approx by a finite dimen Gaussian r.v. $\widetilde{Y}$
  - The length of $\widetilde{Y}$ is $902,629$
  - $\widetilde{Y} = \mu + \sigma C^{1/2} Z, \quad Z \sim N(0, I)$
  - The covariance matrix is thus 902629×902629

Problem 2: Modeling of MS Lesion Location

- Model the locations of Multiple Sclerosis lesions in 217 subjects
- Intensity function $\lambda(x_v) = \exp[Y(x_v))]$
  - $Y(\cdot)$ is a Gaussian process in $\mathcal{B} \subset \mathbb{R}^3$
  - mean of process is μ
  - stationary correlation function $r(||x - z||) = \sigma^2 \exp(-\rho||x - z||^\delta)$.
- $\mathcal{B}$ is discretized on a grid ($91 \times 109 \times 91$)
- $Y(\cdot)$ is approx by a finite dimen Gaussian r.v. $\widetilde{Y}$
  - The length of $\widetilde{Y}$ is $902, 629$
  - $\widetilde{Y} = \mu + \sigma C^{1/2} Z, \quad Z \sim N(0, I)$
  - The covariance matrix is thus 902629×902629
- obviously, this matrix is too large to handle computationally

Problem 2: Modeling of MS Lesion Location

- The structure of most stationary correlation matrices is Toeplitz in 1D, block Toeplitz in 2D and nested block Toeplitz in 3D.

[2] Wood and Chan (1994) *JCGS*

Problem 2: Modeling of MS Lesion Location

- The structure of most stationary correlation matrices is Toeplitz in 1D, block Toeplitz in 2D and nested block Toeplitz in 3D.
- These matrices can be embedded in circulant, block circulant and nested block circulant matrices. [2]
    - C fully specified by single row/column: called the base, c

[2] Wood and Chan (1994) *JCGS*

Problem 2: Modeling of MS Lesion Location

- The structure of most stationary correlation matrices is Toeplitz in 1D, block Toeplitz in 2D and nested block Toeplitz in 3D.
- These matrices can be embedded in circulant, block circulant and nested block circulant matrices. [2]
  - C fully specified by single row/column: called the base, c
- Let $C = F\Lambda F^H$ denote it's eigenvector/value decomposition

[2] Wood and Chan (1994) *JCGS*

Problem 2: Modeling of MS Lesion Location

- The structure of most stationary correlation matrices is Toeplitz in 1D, block Toeplitz in 2D and nested block Toeplitz in 3D.
- These matrices can be embedded in circulant, block circulant and nested block circulant matrices. [2]
  - C fully specified by single row/column: called the base, c
- Let $C = F \Lambda F^H$ denote it's eigenvector/value decomposition
- F turns out to be the DFT matrix and $\Lambda = \sqrt{n}\,\mathrm{diag}(Fc)$.
  - c is of length $n = 2 \cdot 91 \times 2 \cdot 109 \times 2 \cdot 91 (= 7,221,032)$
  - for any vector v, $Fv \leftrightarrow \mathrm{DFT}(v)$ and $F^H v \leftrightarrow \mathrm{IDFT}(v)$.

[2] Wood and Chan (1994) *JCGS*

Problem 2: Modeling of MS Lesion Location

- The structure of most stationary correlation matrices is Toeplitz in 1D, block Toeplitz in 2D and nested block Toeplitz in 3D.
- These matrices can be embedded in circulant, block circulant and nested block circulant matrices. [2]
  - C fully specified by single row/column: called the base, c
- Let $C = F \Lambda F^H$ denote it's eigenvector/value decomposition
- F turns out to be the DFT matrix and $\Lambda = \sqrt{n}\,\mathrm{diag}(Fc)$.
  - c is of length $n = 2 \cdot 91 \times 2 \cdot 109 \times 2 \cdot 91 (= 7,221,032)$
  - for any vector v, $Fv \leftrightarrow \mathrm{DFT}(v)$ and $F^H v \leftrightarrow \mathrm{IDFT}(v)$.
- Thus, we can use the FFT which is extremely fast: $O(n \log_2 n)$.

[2] Wood and Chan (1994) *JCGS*

Problem 2: Modeling of MS Lesion Location

For example

$$\begin{aligned} C^{1/2}v &= F\Lambda^{1/2}F^{H}v \\ &= F\sqrt{n}\,\mathrm{diag}(\sqrt{Fc})F^{H}v \\ &= \sqrt{n}\,\mathrm{DFT}(\sqrt{\mathrm{DFT}(c)} \odot \mathrm{IDFT}(v)) \end{aligned}$$

Problem 2: Modeling of MS Lesion Location

For example

$$
\begin{aligned}
C^{1/2}v &= F\Lambda^{1/2}F^{H}v \\
&= F\sqrt{n}\,\mathrm{diag}(\sqrt{Fc})F^{H}v \\
&= \sqrt{n}\,\mathrm{DFT}(\sqrt{\mathrm{DFT}(c)} \odot \mathrm{IDFT}(v))
\end{aligned}
$$

or solving $Cx = b$:

$$
\begin{aligned}
x &= C^{-1}b \\
&= F\Lambda^{-1}F^{H}b \\
&= \frac{1}{\sqrt{n}}\mathrm{DFT}(\mathrm{IDFT}(b) \oslash \mathrm{DFT}(c))
\end{aligned}
$$

Problem 2: Modeling of MS Lesion Location

For example

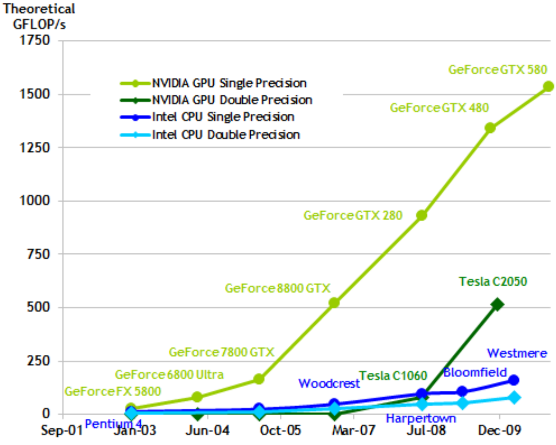
$$
\begin{aligned}
C^{1/2}v &= F\Lambda^{1/2}F^{H}v \\
&= F\sqrt{n}\,\mathrm{diag}(\sqrt{Fc})F^{H}v \\
&= \sqrt{n}\,\mathrm{DFT}(\sqrt{\mathrm{DFT}(c)} \odot \mathrm{IDFT}(v))
\end{aligned}
$$

or solving $Cx = b$:

$$
\begin{aligned}
x &= C^{-1}b \\
&= F\Lambda^{-1}F^{H}b \\
&= \frac{1}{\sqrt{n}}\mathrm{DFT}(\mathrm{IDFT}(b) \oslash \mathrm{DFT}(c))
\end{aligned}
$$

Although the FFT algorithm is fast, the base c in this problem is very large. Hence, overall the MCMC algorithm is quite slow.

General Purpose Graphical Processing Unit Computing



[3]

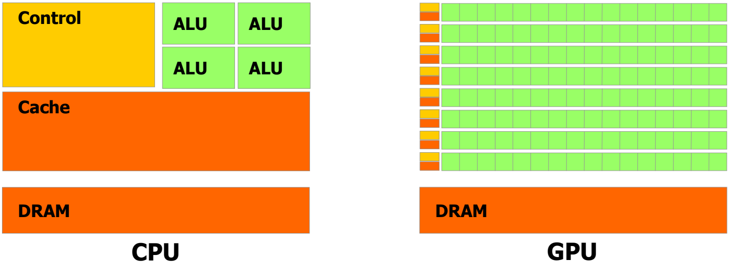
Tesla C2050 GPU Single Precision — 1 TFLOP/s

Tesla M2090 GPU Single Precision — 1.33 TFLOP/s

[3] Source: NVIDIA CUDA C Programming Guide, Version 4.0

General Purpose Graphical Processing Unit Computing

How do GPUs achieve such stunning performance gains over CPUs?



[4]

More transistors are dedicated to data processing rather than data caching and control flow. GPUs typically have hundreds of cores. (Tesla C2050 — 448 cores)

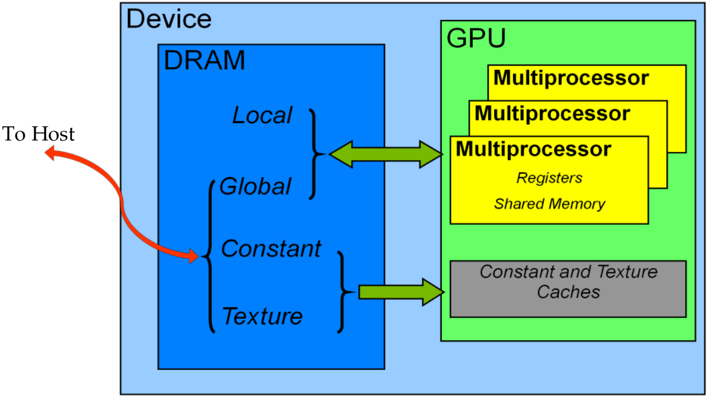
[4] Source: NVIDIA CUDA C Programming Guide, Version 4.0

GPU Memory Bandwidth Limitations

Key to performance with HDD: reduce memory transfers

$\leftarrow$ More memory

Faster transfer $\rightarrow$



5

---

[5] Source: NVIDIA CUDA C Best Practices Guide, Version 4.0

How is the GPU exposed in C/C++ ?

C/C++ interface (CUDA and OpenCL)

- allocate memory on the GPU — special malloc function
- copy data from CPU to GPU memory — special function
- call kernel function—executes on GPU, N times in parallel
- copy GPU output to CPU — special function
- free GPU memory

Kernel Functions

- kernels—C function, executed on GPU, N times in parallel by N threads

Kernel Functions

- kernels—C function, executed on GPU, N times in parallel by N threads
- threads—smallest functional units. Each thread executes the kernel on a separate piece of data. Each thread has its own register memory, invisible to all other threads

Kernel Functions

- kernels—C function, executed on GPU, N times in parallel by N threads
- threads—smallest functional units. Each thread executes the kernel on a separate piece of data. Each thread has its own register memory, invisible to all other threads
- blocks—threads are partitioned into blocks. Threads within a block can cooperate with one another and can share memory. Threads in different blocks cannot cooperate.

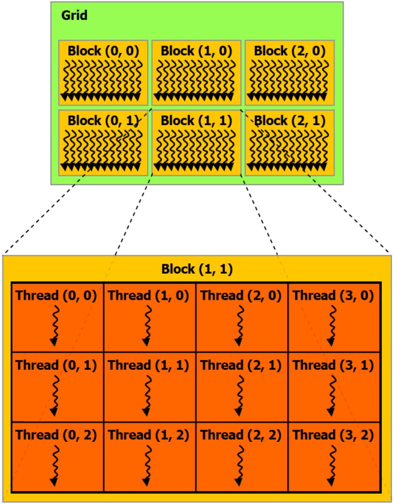
Kernel Functions

- kernels—C function, executed on GPU, N times in parallel by N threads
- threads—smallest functional units. Each thread executes the kernel on a separate piece of data. Each thread has its own register memory, invisible to all other threads
- blocks—threads are partitioned into blocks. Threads within a block can cooperate with one another and can share memory. Threads in different blocks cannot cooperate.
- grid—blocks of threads are partitioned onto a grid.

Kernel Functions

- kernels—C function, executed on GPU, N times in parallel by N threads
- threads—smallest functional units. Each thread executes the kernel on a separate piece of data. Each thread has its own register memory, invisible to all other threads
- blocks—threads are partitioned into blocks. Threads within a block can cooperate with one another and can share memory. Threads in different blocks cannot cooperate.
- grid—blocks of threads are partitioned onto a grid.
- The host interface (the C program) calls the kernel
  - specifies the number of threads/block and the number of blocks/grid
  - passes memory pointers where data have been loaded onto the GPU.

How is the GPU exposed in C/C++ ?



[6]

[6] Source: NVIDIA CUDA C Programming Guide, Version 4.0

Example

Typical C code

- loop over voxels, *i*
  - loop over mixture comp, *j*
    - calculate *j*th comp contrib. to likelihood at voxel i
  - end mixture comp loop
- end voxel loop

Example

Typical C code

- loop over voxels, *i*
  - loop over mixture comp, *j*
    - calculate *j*th comp contrib. to likelihood at voxel i
  - end mixture comp loop
- end voxel loop

Parallel code

- allocate GPU memory
- data copy CPU → GPU
- call kernel()
- data copy GPU → CPU
- free GPU memory

kernel code (voxels processed in parallel)

- load shared memory
- loop over mixture comp, *j*
  - calculate *j*th comp contrib. to likelihood at voxel i
- end mixture comp loop

Gain in Computational Efficiancy

Model	Processor	Iterations (x 1000)	Memory (Mb)	Time (hours)	Factor
groupFMRI	CPU	5		229.52	
	GPU	5	175	1.20	191
LGCP	CPU	120		595.00	
	GPU	120	311	4.75	125

Caveats

- reduce data transfers
- beware of conditional branching (e.g. if-else)
- bank conflicts (memory)
  - for shared memory, a bank conflict occurs when two or more threads within a warp try to access different bytes of memory within the bank
    - a warp is a group of 32 threads—minimum size of data processed. A hardware constraint. Blocks of threads are further divided into warps.
  - if two or more threads try to access the same byte, no bank conflict occurs
  - may need to pad arrays to avoid bank conflicts
  - in my opinion, most difficult aspect of GPGPU computing

Conclusions

- GPGPU computing can substantially improve performance
- Likelihood based approaches can also gain from GPGPU computing
- In general, the larger the problem, the greater the practical gains.