GPU-parallel Gibbs sampling of a hierarchical model of hybrid vigor in RNA-seq experiments

Will Landau

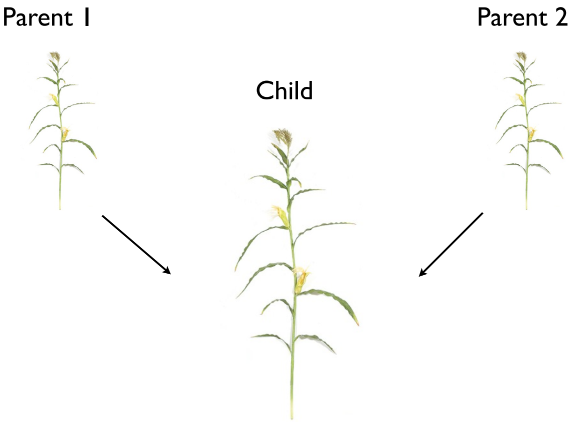
Iowa State University

October 10, 2013

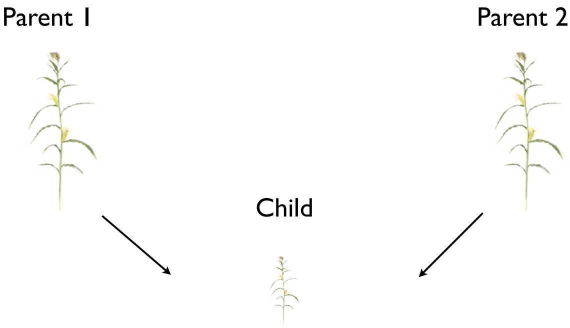
Outline

Outline

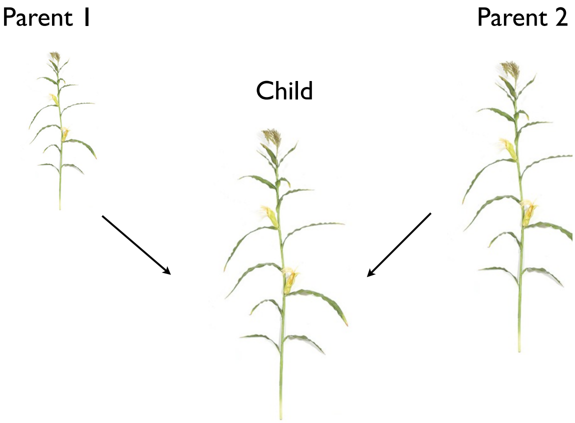
High-parent heterosis: child's trait surpasses both parents



Low-parent heterosis: child's trait is weaker than in each parent



Mid-parent heterosis: child's trait is different than average of parents



High-parent heterosis in gene expression

	Parent 1			Child		Parent 2	
Gene 1	100	225	0	70	279	300	106
Gene 2	0	1	1	50	501	2	7
Gene 3	3	4	2	700	900	0	0
Gene 4	893	400	760	5	5	1000	513
...	...	...	...	...	...	...	...
Gene 34897	10	13	6	819	761	902	912

Low-parent heterosis in gene expression

	Parent 1			Child		Parent 2	
Gene 1	100	225	0	70	279	300	106
Gene 2	0	1	1	50	501	2	7
Gene 3	3	4	2	700	900	0	0
Gene 4	893	400	760	5	5	1000	513
...	...	...	...	...	...	...	...
Gene 34897	10	13	6	819	761	902	912

Mid-parent heterosis in gene expression

	Parent 1			Child		Parent 2	
Gene 1	100	225	0	70	279	300	106
Gene 2	0	1	1	50	501	2	7
Gene 3	3	4	2	700	900	0	0
Gene 4	893	400	760	5	5	1000	513
...	...	...	...	...	...	...	...
Gene 34897	10	13	6	819	761	902	912

Outline

The model

$$\mu(n, \phi_g, \alpha_g, \delta_g) = \begin{cases} \phi_g - \alpha_g & \text{sample } n \text{ from parent 1} \\ \phi_g + \delta_g & \text{sample } n \text{ from child} \\ \phi_g + \alpha_g & \text{sample } n \text{ from parent 2} \end{cases}$$

$$y_{g,n} \overset{\text{ind}}{\sim} \text{Poisson}(\exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g)))$$

$$c_n \overset{\text{ind}}{\sim} \text{N}(0, \sigma_c^2)$$

$$\sigma_c \sim \text{U}(0, \sigma_{c0})$$

$$\varepsilon_{g,n} \overset{\text{ind}}{\sim} \text{N}(0, \eta_g^2)$$

$$\eta_g^2 \overset{\text{ind}}{\sim} \text{Inv-Gamma}\left(\text{shape} = \frac{d}{2}, \ \text{rate} = \frac{d \cdot \tau^2}{2}\right)$$

$$d \sim \text{U}(0, d_0)$$

$$\tau^2 \sim \text{Gamma}(\text{shape} = a_\tau, \text{rate} = b_\tau)$$

The model

$$\mu(n, \phi_g, \alpha_g, \delta_g) = \begin{cases} \phi_g - \alpha_g & \text{sample } n \text{ from parent 1} \\ \phi_g + \delta_g & \text{sample } n \text{ from child} \\ \phi_g + \alpha_g & \text{sample } n \text{ from parent 2} \end{cases}$$

$$y_{g,n} \overset{\text{ind}}{\sim} \text{Poisson}(\exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g)))$$

$$\phi_g \overset{\text{ind}}{\sim} \text{N}(\theta_\phi, \sigma_\phi^2)$$

$$\theta_\phi \sim \text{N}(0, \gamma_\phi^2)$$

$$\sigma_\phi \sim \text{U}(0, \sigma_{\phi 0})$$

$$\alpha_g \overset{\text{ind}}{\sim} (1 - I(\alpha_g)) \cdot \pi_\alpha + I(\alpha_g) \cdot (1 - \pi_\alpha) \cdot \text{N}(\alpha_g \mid \theta_\alpha, \sigma_\alpha^2)$$

$$\theta_\alpha \sim \text{N}(0, \ \gamma_\alpha^2)$$

$$\sigma_\alpha \sim \text{U}(0, \ \sigma_{\alpha 0})$$

$$\pi_\alpha \sim \text{Beta}(a_\alpha, \ b_\alpha)$$

$$\delta_g \overset{\text{ind}}{\sim} (1 - I(\delta_g)) \cdot \pi_\delta + I(\delta_g) \cdot (1 - \pi_\delta) \cdot \text{N}(\delta_g \mid \theta_\delta, \sigma_\delta^2)$$

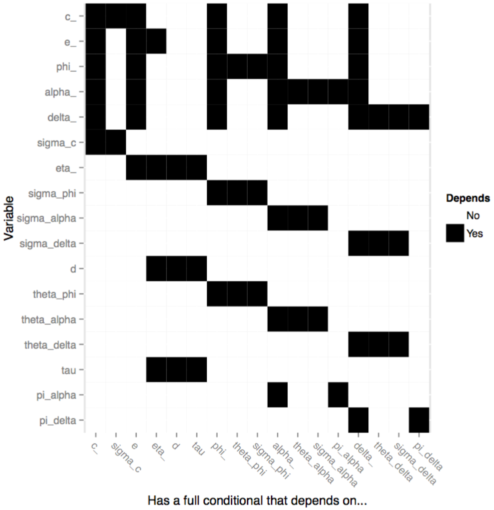
$$\theta_\delta \sim \text{N}(0, \gamma_\delta^2)$$

$$\sigma_\delta \sim \text{U}(0, \sigma_{\delta 0})$$

$$\pi_\delta \sim \text{Beta}(a_\delta, b_\delta)$$

Outline

Partition parameters by conditional independence.



Use these partitions as Gibbs steps.

$$y_{g,n} \stackrel{\text{ind}}{\sim} \text{Poisson}(\exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g)))$$

- From the appropriate full conditional distributions, sample the following:

1. $c_1, \ldots, c_N$
2. $\tau, \pi_\alpha, \pi_\delta$
3. $d, \theta_\phi, \theta_\alpha, \theta_\delta$
4. $\sigma_c, \sigma_\phi, \sigma_\alpha, \sigma_\delta, \eta_1^2, \ldots, \eta_G^2$
5. $\varepsilon_{1,1}, \varepsilon_{1,2}, \ldots, \varepsilon_{1,N}, \varepsilon_{2,N}, \ldots, \varepsilon_{G,N}$
6. $\phi_1, \ldots, \phi_G$
7. $\alpha_1, \ldots, \alpha_G$
8. $\delta_1, \ldots, \delta_G$

- and then repeat.

Estimated heterosis probabilities

$$\mu(n, \phi_g, \alpha_g, \delta_g) = \begin{cases} \phi_g - \alpha_g & \text{sample } n \text{ from parent 1} \\ \phi_g + \delta_g & \text{sample } n \text{ from child} \\ \phi_g + \alpha_g & \text{sample } n \text{ from parent 2} \end{cases}$$

$$y_{g,n} \stackrel{\text{ind}}{\sim} \text{Poisson}(\exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g)))$$

Consider one chain with M iterations.

$$P(\text{high-parent heterosis in gene } g) \approx \frac{1}{M} \sum_{i=1}^{M} I(\delta_g^{(i)} > |\alpha_g^{(i)}|)$$

$$P(\text{low-parent heterosis in gene } g\ ) \approx \frac{1}{M} \sum_{i=1}^{M} I(\delta_g^{(i)} < -|\alpha_g^{(i)}|)$$

$$P(\text{mid-parent heterosis in gene } g\ ) \approx \frac{1}{M} \sum_{i=1}^{M} I(\delta_g^{(i)} \neq 0)$$

Outline

The single instruction, multiple data (SIMD) paradigm

- SIMD: apply the same command to multiple places in a dataset.

```
for(i = 0; i < 1e6; ++i)
  a[i] = b[i] + c[i];
```

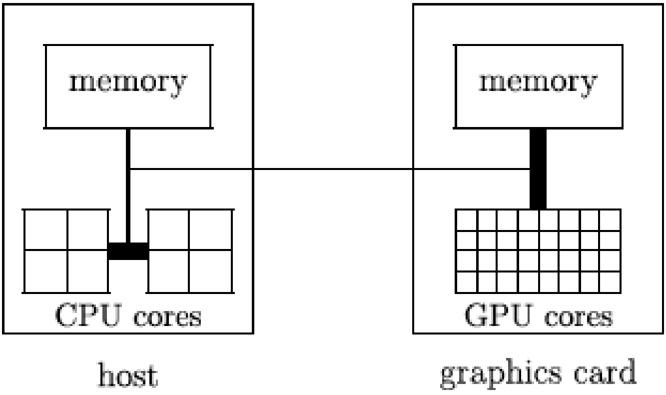
- On CPUs, the iterations of the loop run sequentially.
- With GPUs, we can easily run all 1,000,000 iterations simultaneously.

```
i = threadIdx.x;
a[i] = b[i] + c[i];
```

- We can similarly *parallelize* a lot more than just loops.

CPU / GPU cooperation

- The CPU ("host") is in charge.
- The CPU sends computationally intensive instruction sets to the GPU ("device") just like a human uses a pocket calculator.



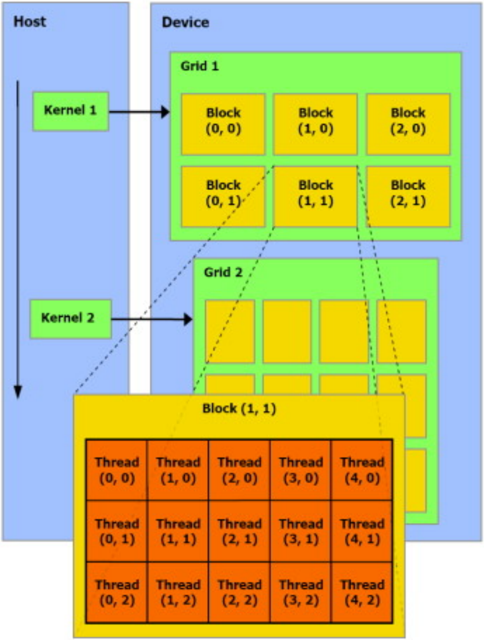
How GPU parallelism works

1. The CPU sends a command called a **kernel** to a GPU.
2. The GPU executes several duplicate realizations of this command, called **threads**.
    - These threads are grouped into bunches called **blocks**.
    - The sum total of all threads in a kernel is called a **grid**.

- Toy example:
    - CPU says: "Hey, GPU. Sum pairs of adjacent numbers. Use the array, (1, 2, 3, 4, 5, 6, 7, 8)."
    - GPU thinks: "Sum pairs of adjacent numbers" is a kernel.
    - The GPU spawns 2 blocks, each with 2 threads:

Block	0		1	
Thread	0	1	0	1
Action	$1+2$	$3+4$	$5+6$	$7+8$

- I could have also used 1 block with 4 threads and given the threads different pairs of numbers.


Host
Device
Grid 1
Kernel 1
Block (0, 0)
Block (1, 0)
Block (2, 0)
Block (0, 1)
Block (1, 1)
Block (2, 1)
Grid 2
Kernel 2
Block (1, 1)
Thread (0, 0)
Thread (1, 0)
Thread (2, 0)
Thread (3, 0)
Thread (4, 0)
Thread (0, 1)
Thread (1, 1)
Thread (2, 1)
Thread (3, 1)
Thread (4, 1)
Thread (0, 2)
Thread (1, 2)
Thread (2, 2)
Thread (3, 2)
Thread (4, 2)

Parallel reductions

- A **reduction** is an operation on a vector that produces a scalar.
- Repeatedly apply a binary operator to pairs of elements in the vector to get the scalar.
- Let's take the pairwise sum of the vector,

$$(5, 2, -3, 1, 1, 8, 2, 6)$$

using 1 block of 4 threads.

Pairwise summation: an example reduction

5 2 -3 1 1 8 2 6

6

Thread 0

Pairwise summation: an example reduction

5 2 -3 1 1 8 2 6

6 10

Thread 1

Pairwise summation: an example reduction

5	**2**	**-3**	**1**	**1**	**8**	**2**	**6**
6	**10**	**-1**					

Thread 2

Pairwise summation: an example reduction

5 2 -3 1 1 8 2 6

6 10 -1 7

Thread 3

Pairwise summation: an example reduction

5	2	-3	1	1	8	2	6
6	10	-1	7				

Synchronize threads

Pairwise summation: an example reduction

5 2 -3 1 1 8 2 6

6 10 -1 7

5

Thread 0

Pairwise summation: an example reduction

5 2 -3 1 1 8 2 6

6 10 -1 7

5 17

Thread 1

Pairwise summation: an example reduction

5	**2**	**-3**	**1**	**1**	**8**	**2**	**6**
6	**10**	**-1**	**7**				
5	**17**						

Synchronize Threads

Pairwise summation: an example reduction

5 2 -3 1 1 8 2 6

6 10 -1 7

5 17

Thread 0

22

Tons of opportunity for GPU parallelism across genes!

$$y_{g,n} \overset{\text{ind}}{\sim} \text{Poisson}(\exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g)))$$

- Sample collections of conditionally independent parameters in parallel:
  - ϕ_g's
  - α_g's
  - δ_g's
  - $\varepsilon_{g,n}$'s
  - η_g's

Example: ϕ_g's

$$
\begin{aligned}
y_{g,n} &\overset{\text{ind}}{\sim} \text{Poisson}(\exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g))) \\
\phi_g &\overset{\text{ind}}{\sim} \text{N}(\theta_\phi, \sigma_\phi^2) \\
\theta_\phi &\sim \text{N}(0, \gamma_\phi^2) \\
\sigma_\phi &\sim \text{U}(0, \sigma_{\phi 0})
\end{aligned}
$$

- Using parallel random walk Metropolis steps, sample the ϕ_g's from their full conditional distributions,

$$
\begin{aligned}
p(\phi_g \mid \cdots) \propto \exp \Bigg(\sum_{n=1}^{N} \big[& y_{g,n} \cdot \mu(n, \phi_g, \alpha_g, \delta_g) \\
& - \exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g)) \big] - \frac{(\phi_g - \theta_\phi)^2}{2\sigma_\phi^2} \Bigg)
\end{aligned}
$$

Tons of opportunity for GPU parallelism across genes!

$$y_{g,n} \overset{\text{ind}}{\sim} \text{Poisson}(\exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g)))$$

- Use **parallel reductions** to calculate sufficient statistics for:
    - c_n's
    - τ, d
    - θ_ϕ, θ_α, θ_δ
    - σ_ϕ, σ_α, σ_δ, σ_c
    - π_α, π_δ

Example: τ^2

$$
\begin{aligned}
y_{g,n} &\overset{\text{ind}}{\sim} \text{Poisson}(\exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g))) \\
\varepsilon_{g,n} &\overset{\text{ind}}{\sim} \text{N}(0, \eta_g^2) \\
\eta_g^2 &\overset{\text{ind}}{\sim} \text{Inv-Gamma}\left(\text{shape} = \frac{d}{2}\ ,\ \text{rate} = \frac{d \cdot \tau^2}{2}\right) \\
d &\sim \text{U}(0, d_0) \\
\tau^2 &\sim \text{Gamma}(\text{shape} = a_\tau, \text{rate} = b_\tau)
\end{aligned}
$$

$$
p(\tau^2 \mid \cdots) = \text{Gamma}\left(\text{shape} = a_\tau + \frac{Gd}{2}\ ,\ \text{rate} = b_\tau + \frac{d}{2}\sum_{g=1}^{G} \frac{1}{\eta_g^2}\right)
$$

- Using a parallel reduction (NVIDIA's CUDA C/C++ `Thrust` library), calculate the sufficient statistic:

$$
\sum_{g=1}^{G} \frac{1}{\eta_g^2}
$$

- Use an efficient rejection sampler to sample τ^2.

Example: d

$$
\begin{aligned}
y_{g,n} &\overset{\text{ind}}{\sim} \text{Poisson}(\exp(c_n + \varepsilon_{g,n} + \mu(n, \phi_g, \alpha_g, \delta_g))) \\
\varepsilon_{g,n} &\overset{\text{ind}}{\sim} \text{N}(0, \eta_g^2) \\
\eta_g^2 &\overset{\text{ind}}{\sim} \text{Inv-Gamma}\left(\text{shape} = \frac{d}{2}, \text{ rate} = \frac{d \cdot \tau^2}{2}\right) \\
d &\sim \text{U}(0, d_0) \\
\tau^2 &\sim \text{Gamma}(\text{shape} = a_\tau, \text{rate} = b_\tau)
\end{aligned}
$$

$$
p(d \mid \cdots) \propto \Gamma\left(d/2\right)^{-G} \left(\frac{d \cdot \tau^2}{2}\right)^{Gd/2} \left(\prod_{g=1}^{G} \eta_g^2\right)^{-(d/2+1)} \exp\left(-\frac{d \cdot \tau^2}{2} \sum_{g=1}^{G} \frac{1}{\eta_g^2}\right) I(0 < d < d_0)
$$

- Using parallel reductions (NVIDIA's CUDA C/C++ `Thrust` library), calculate the sufficient statistics:

$$
\prod_{g=1}^{G} \eta_g^2 \qquad \sum_{g=1}^{G} \frac{1}{\eta_g^2}
$$

- Use a random-walk metropolis step to sample d.

Outline

The software

- In progress...

Sources

1. A. Gelman, J. B. Carlin, H. S. Stern, and D. S. Rubin. Bayesian Data Analysis. Chapman & Hall/CRC, 2 edition, 2004.
2. Prof. Jarad Niemi's STAT 544 lecture notes.
3. J. Sanders and E. Kandrot. *CUDA by Example.* Addison-Wesley, 2010.
4. http://www.astrochem.org/sci/Nucleobases.php
5. http://www.biologycorner.com/bio1/DNA.html
6. http://www.qualitysilks.com/images/products/artificial-corn-stalk.jpg
7. http://en.wikipedia.org/wiki/dna
8. http://en.wikipedia.org/wiki/rna
9. http://en.wikipedia.org/wiki/HSP60