High Performance Gibbs Sampler for Mixed Density General Linear Systems

B. L. Golden[*], C. Lupo[°] and D. J. Garrick[◆]

[*]Theta Solutions, LLC, Atascadero CA; [°]Cal Poly, San Luis Obispo; [◆] Iowa State Univ., Ames

1.Abstract

The objective of the study was to determine if improvements in compute performance for a Gibbs Sampler (GS) applied to very large sparse or mixed-density general linear systems could be obtained using heterogeneous computing with CUDA. The GS is a Markov chain Monte Carlo method that has been used to approximate the joint distribution and marginal distribution of variables. It has been used in statistical problems, especially quantitative genetics/genomics, to obtain posterior distributions of perhaps millions of variables such as when making inferences about genomic or genetic effects in selection of livestock or plants. In conventional implementations GS involves sampling plausible values of the variables at single-sites (ssGS) or in blocks, after updating observations or right-hand sides for current values of all the other sampled effects. The process of sampling is repeated tens of thousands of times to generate a chain from which the posterior distributions are determined. For 10's of millions or more equations, using CPU based methods have been intractably slow. We propose an ssGS algorithm that partitions the linear system into an arbitrary number of blocks, then samples within the blocks after asynchronously adjusting the right-hand side for previous samples. The performance of the algorithm was evaluated for three sizes of linear systems taken from real problems in animal genetics.

2. Background

Methods for the solution of very large sparse linear systems have been well known and widely used in many fields. The linear system can be represented as,

$$\mathbf{Ax = b}$$

Where A is a square often symmetric coefficient or left-hand side matrix, b is a right hand side vector of known values and x is a vector of values for which inference is required. In statistical problems involving mixed linear models the linear system involves one or more residual variance parameters, and variance parameters of random effects in the model. The most popular method for solving this system is conjugant gradient and often incorporates some form of pre-conditioner (PCG). However, PCG and other non-sampling based methods for obtaining solutions cannot obtain the estimates of the sampling variances of the variables, which are necessary for statistical inference. GS can be used to estimate these variances.

A typical conventional implementation of the ssGS is,

```
FOR sample = 1 to Thousands
        FOR i = 1 to length of vector x
            $\hat{x}_i = (b_i - \sum_{j \neq i} a_{ij}\tilde{x}_j) / a_{ii}$
            $\tilde{x}_i = \hat{x}_i + RND(0,1) \times \sigma_e \times \sqrt{1/a_{ii}}$
        END FOR
END FOR
```

For many problems in genetics the majority of rows in A are very sparse and using a sparse dot product on GPU does not speed the computation because of the relatively small number of non-zero values in a row of A.

3. Methods

Partition,

Ax = b as,

$$\begin{bmatrix} \mathbf{A}_{11} & \cdots & \mathbf{A}_{1n} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{n1} & \cdots & \mathbf{A}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{bmatrix}$$

Sample,

```
Initialize on GPU $\mathbf{r} = \mathbf{b}$
FOR each sample chain of x
        FOR i is each row block of A
            Synchronize stream i
            FOR k is each row of $\mathbf{A}_{ii}$                                  [1]
                $\hat{x}_{ik} = (r_{ik} - \sum_{l \neq k} a_{kl}\tilde{x}_{il}) / a_{kk}$
                $\tilde{x}_{ik} = \hat{x}_{ik} + RND(0,1) \times \sigma_{ei} \times \sqrt{1/a_{kk}}$
            END FOR

            FOR $k \neq i$                                                   [2]
                Launch async on GPU $\mathbf{r}_k \mathrel{-}= \mathbf{A}_{ki}\tilde{\mathbf{x}}_i$
            END FOR
            Launch async on GPU $\mathbf{r}_i = \mathbf{b}_i$
        END FOR
END FOR
```

•The algorithm's performance was compared to an optimized conventional ssGS.

•The Mersenne twister in CURAND was used for both implementations.

•Three dataset sizes were drawn from the national cattle database administered by the American Simmental Association.

•The analytical model was a three trait multi-trait model with fixed effects and random additive direct and maternal genetic effects.

•The linear system was partitioned into 14 row blocks, 9 for fixed effects, 4 for the animal genetic effects and 1 for a permanent environmental effect due to the dam. Blocks of the 14x14 LHS varied in size and sparsity

•The computer used Ubuntu 12.04, an NVIDIA® Titan Z, and had a single 6 core I-4930K processor over clocked to 4.25GhZ. Only 1 GPU processor was used on the Titan Z.

•CUDA 6.5, Gnu C 4.8 using –mavx, and the cusparse and cublas libraries were used for the sparse Mv and dense Mv (e.g., covariates).

•Blocks of A were asynchronously copied to the GPU, allowing the CPU to perform sampling computations in [1] and the GPU to perform the Mv computations in [2] in parallel with the data copies.

•Separate streams were used to perform the asynchronous data transfers and Mv computations in [2] for each matrix copy/Mv.

•Blocks of A were stored as separate SCsr matrices and the x and b vectors were stored as dense floats.

4. Results

Table 1. Characteristics of three test linear systems taken from a typical very large problem in animal genetics

Problem Size	NNZ	# Equations	Density	# Animals Observed
Small	146,683,254	5,104,752	0.000563%	445,785
Medium	1,095,956,018	28,274,608	0.000137%	4,619,345
Large	2,071,863,441	51,448,023	0.000078%	10,412,175

Table 2. Time for a complete sample of the x vector for traditional and new algorithm

	Time for 1 Sample of x (s)		
Problem Size	Traditional Gibbs Sampler	New Algorithm	Performance Improvement
Small	1.482	0.407	3.65x
Medium	11.450	2.007	5.71x
Large	22.721	4.352	5.22x

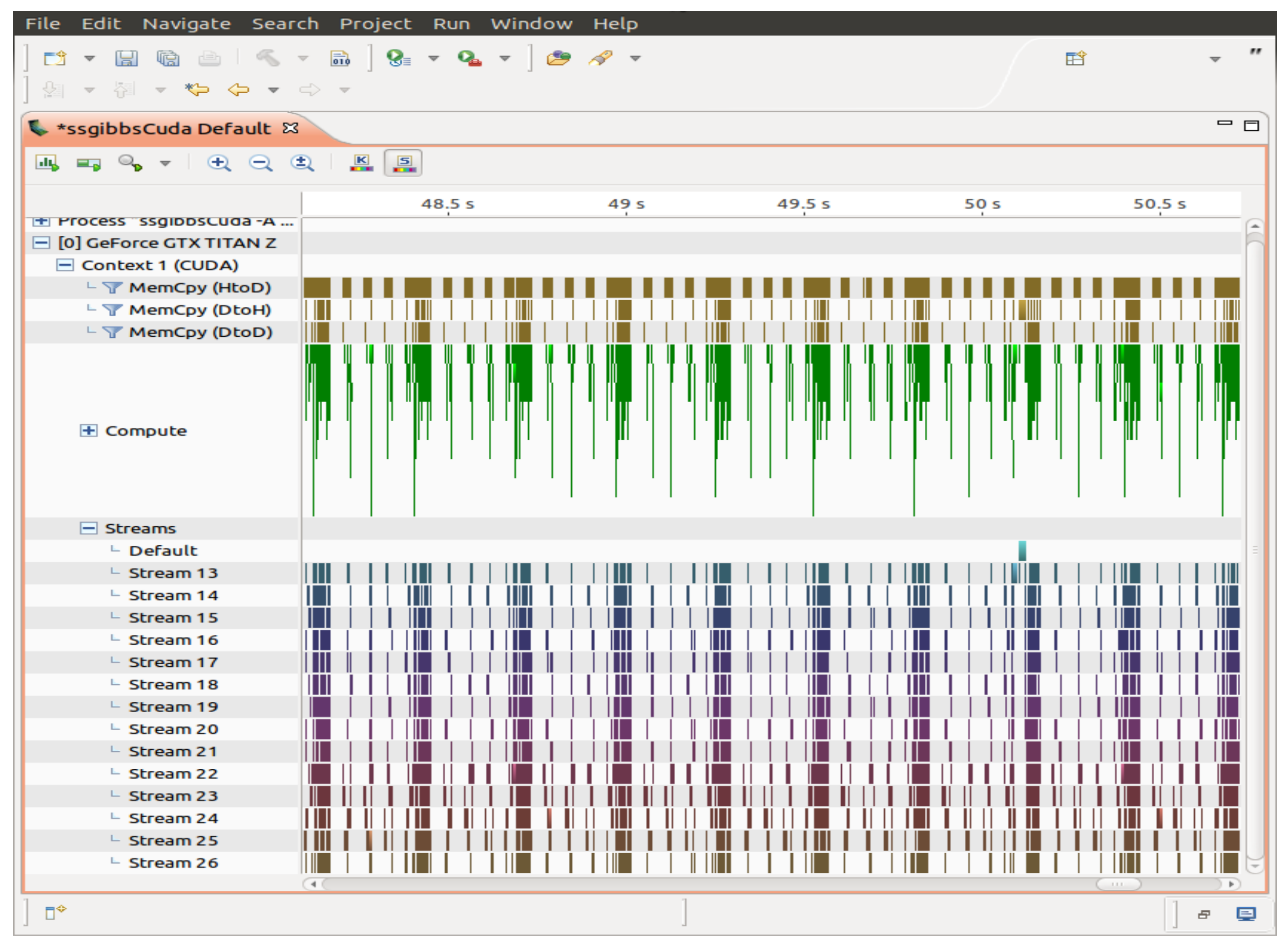

Figure 1. Output from the NVIDIA® Visual Profiler when sampling using the new algorithm and the "Small" problem size.

5. Conclusions

Substantial performance improvements were achieved (Table 2) for the heterogeneous GPU and CPU implementation compared to the CPU only based ssGS on large linear system problems (Table 1). These improvements were obtained through a new algorithm that could better leverage a combination of GPU based sparse matrix –vector multiplications that occurred asynchronously while sampling using smaller block diagonals of the LHS, and asynchronous data transfers of off diagonal blocks of the LHS to the GPU using CUDA streams. From Figure 1, asynchronous data transfer and computation on the GPU occurred for only about 40% of the time that the CPU was computing the samples. Future study should explore optimal block sizes and arrangement of equations that would move as many non-zero values out of the diagonal blocks as possible. Computing the dot products within the diagonal blocks while sampling remained the bottleneck.