

UC Graduate BMI Presentation

Lee A. Carraher

School of Electronic and Computing Systems
University of Cincinnati

July 31, 2014

- ▶ From Cincinnati
- ▶ B.S. Computer Engineering (UC 2008)
- ▶ M.S. Computer Science (UC 2012)
- ▶ Ph.D Computer Science and Engineering (ongoing)

Some research interests:

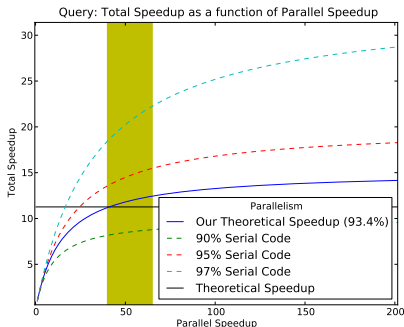
- ▶ Machine Learning (bioinformatics, filtering)
- ▶ Inverse Problems (min/max problems)
- ▶ Parallel Computing (CUDA, MPI)
- ▶ Distributed Computing (Mapreduce, Spark)
- ▶ Big Data

- ▶ Worked with Prof. Wilsey
- ▶ Researched Parallel Algorithms for Data Clustering
- ▶ Identified Issues in Clustering and Parallelism
- ▶ Formulated ways to combat these problems
- ▶ Considered Issues of data privacy
- ▶ Advised Jordan Ross in development of Bio Blocks
<https://docs.google.com/document/d/1kES5nI4EXUOtcj9j0D9joEkIpZTgW5hKUAHx0BmjQak/pub>

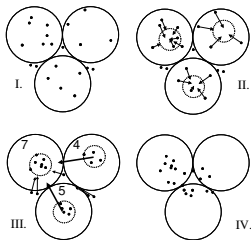
Parallel Clustering Algorithms

A common algorithmic theme of clustering is iterative updating.

- ▶ Format of Kmeans, EM/LDA, and Mean-Shift
- ▶ Presents a problem for parallelism (sequential bottleneck/Amdahl)
- ▶ Bad during communication



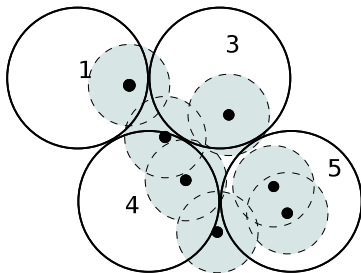
- ▶ Partition data across nodes
- ▶ Count partition sizes
- ▶ Move datapoints toward large partitions



Unsuccessful due to too much communication and the ever present iterative structure.

- ▶ Random Projection Hash Clustering
- ▶ Random Project vectors into a partitions of the Leech Lattice
- ▶ Count Lattice Partition Counts
- ▶ Accumulate partition counts across nodes($\Theta(\log\text{-reduce})$)

- ▶ Project randomly multiple times for each point
- ▶ Projected point has a distribution about the optimal projection (blurring)
- ▶ Counting and key-mapping fit well in the map reduce framework



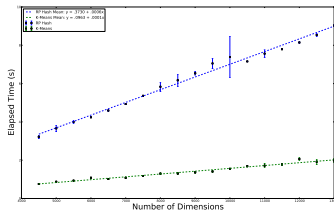
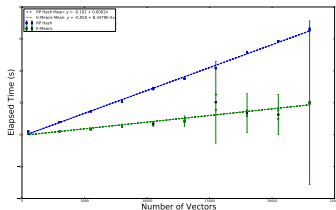


Figure : Computation Time for K-Means(green) and *RPHash*(blue) varying Vectors and Dimensions

- ▶ Worse than k-means!
- ▶ ok, since we are trading sequential complexity for parallel speedup

- ▶ RP is destructive mapping with possible application to deidentification attack prevention
- ▶ Important following infamous attacks
- ▶ Presidential Commison of WGS security[?]

$$u = \sqrt{\frac{n}{k}} R_{d \rightarrow s}^T v, v' = \sqrt{\frac{k}{n}} u^T R_{d \rightarrow s}^{-1},$$

$$s(v, v') = \|v, v'\|_2,$$

$$\forall \{v, v'\} \in V, \exists \hat{v} \in V : s(v, v') > s(\hat{v}, v) \text{ where } \hat{v} \neq v$$



- ▶ Ongoing research with NSF proposal being recommended for funding.
- ▶ Current subject of ongoing dissertation research

- ▶ Worked with Prof. Medvedovic
- ▶ Learned about Gene enrichment and Gaussian Infinite Mixture Model
- ▶ Expectation Maximization(EM) in java for MLE of parameters (java)
- ▶ Attempt to parallelize GIMM directly on GPU
- ▶ Theano: CPU and GPU compiler for math in Python
- ▶ Elastic Cloud GIMM server backend

UNIVERSITY OF
Cincinnati



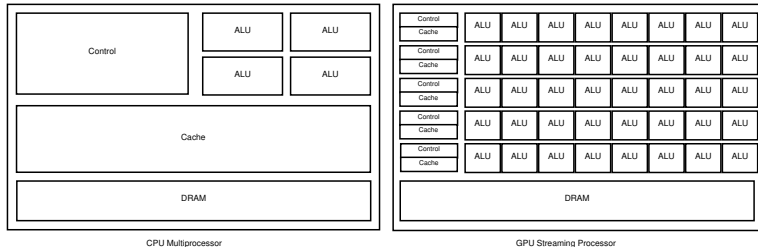


- ▶ Differential Expression is important, fisher's..
- ▶ MLE can approximate p-values via EM algorithm
- ▶ Developed a maximum-likelihood estimator in java, target hadoop
- ▶ Hadoop is a Map Reduce(MR) processing engine
- ▶ Mahout contains an MR optimized EM algorithm
- ▶ GIMM allows for mixture of prob. dist.

Gaussian Infinite Mixture Model (sometimes IGMM)

- ▶ Instead of EM, GIMM uses Bayesian Estimation
- ▶ Gibb's is a Markov Chain Monte Carlo Algorithm
- ▶ Gibb's Sampling uses conjugate priors (Dirichlet, Inverse Wishart)
- ▶ Main Process is Sampling the Prior Distribution

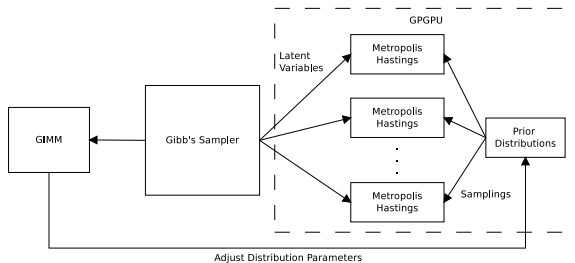
GPGPU's have lots of ALUs



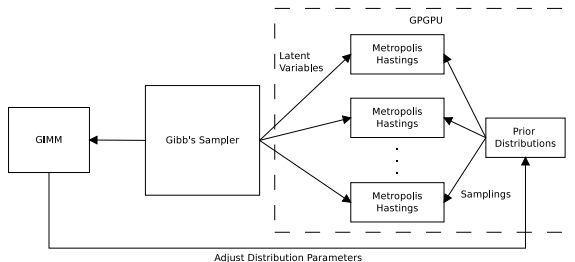
- ▶ More Arithmetic Logic Units Compared to Control Logic and Memory
- ▶ More common as more libraries leveraging GPU's are made available.
- ▶ NVIDIA implementation CUDA, also OpenCL for AMD

Before trying to implement directly, consider existing GPGPU acceleration in R and Matlab.

- ▶ Matlab has most Lin Alg implemented in CUDA
- ▶ Also has keywords for accessing gpu cores
- ▶ R has HiPlar - similar to Matlab (Lin Alg, Matrix)
- ▶ R can be C so also have direct cuda implementations of code
- ▶ gimmR is a compiled binary, so not very useful directly.



- ▶ Find parallelizable portions of GIMM (sampler)
- ▶ Most of the time spent generating samples
- ▶ Performance modeling this part in c and python



- ▶ Major Part is the Gibb's Sampler
- ▶ Can random distributions be sampled faster
- ▶ Fast gamma variate and Gaussian variates
- ▶ How do we overlap transfers to improve SPU occupancy and minimize sequential bottleneck

- ▶ From GPU Gems Nvidia
- ▶ Implement a wallace pool prng
- ▶ Based on Walsh-Hadamard Matrix
- ▶ GPU-based Wallace generator provides a speedup of 26 times

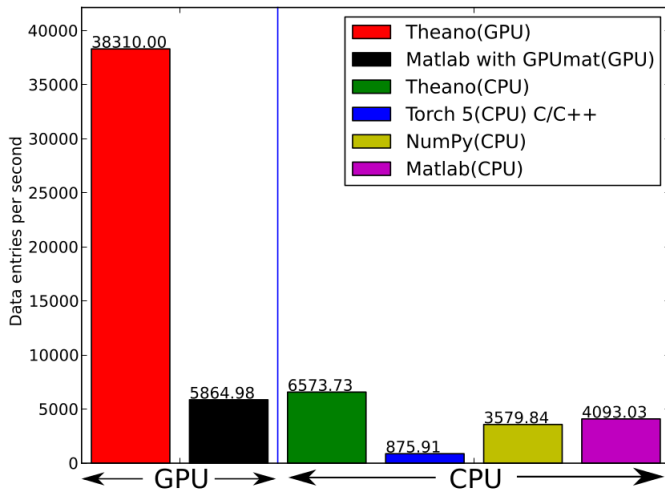
- ▶ CUDA installation had many issues, but eventually mostly worked (glu libs were broken)
- ▶ A direct CUDA implementation with occupancy tuning would likely be optimal
- ▶ Could not fit inverse-wishart code into a single thread's memory
- ▶ Global memory is very slow
- ▶ Implemented simpler Gaussian sampler
- ▶ Issues with data transfer bottleneck are apparent in naive code
- ▶ A CUDA implementation is realizable but would require tuning.

- ▶ Scripting Languages are slow, why is this here?
- ▶ Pylab/Numpy Give fast access to c functions, like R
- ▶ Implement a Gibb's Sampler in python not very fast even with pypy (4x slower than c)
- ▶ But there are more inventive ways to interpret Python

Theano is a mathematic expression compiler

- ▶ Mathematical expressions are defined in python
- ▶ Theano uses many parallel optimized functions from BLAS to turn operations into SIMD vector operations
- ▶ Theano supports transparent CUDA (GPGPU) conversion
- ▶ overlapped data transfer is automatically performed by Theano

Theano Performance Possibilities



conference.scipy.org/scipy2010/slides/james_bergstra_theano.pdf

- ▶ Functions as python module
- ▶ Add static decorators to datatypes
- ▶ Convert function into expression graphs
- ▶ Compile expression graphs into optimized c or gpu code

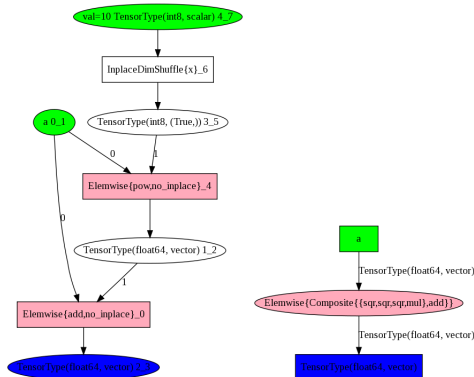
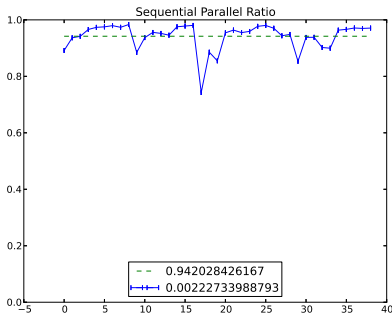


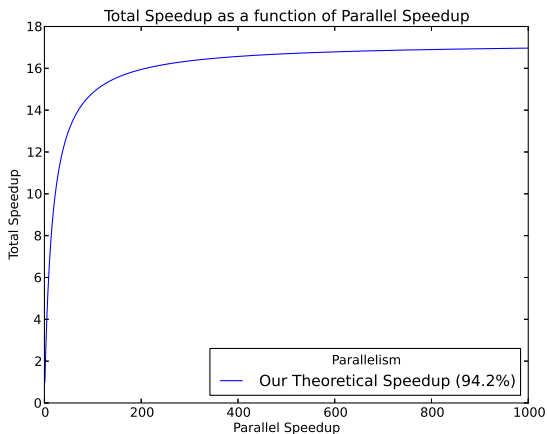
Figure :

(http://deeplearning.net/software/theano/tutorial/symbolic_graphs.html)



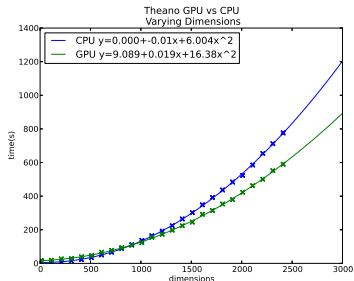
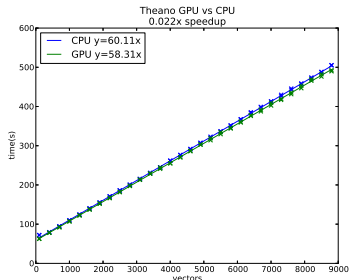
- ▶ Majority of processing occurs in the pdf generation
- ▶ 94.2% can be sped up by focusing here
- ▶ pdf generation is naively parallel

Applying Amdahl's Law



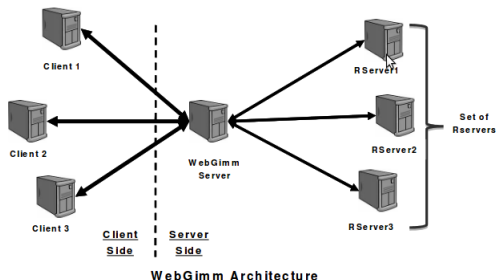
Applying Amdahl's Law on 94.2% sequential to parallel ratio.

Real World Speedup?



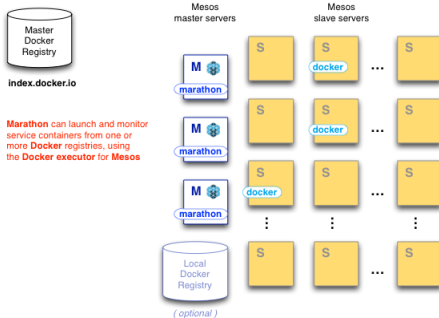
- ▶ Cost of JIT and cpu→gpu transfer times
- ▶ Dimension calls are vector ops in Theano,
 - ▶ More data per DMA transfer cycle

- ▶ Individual problems aren't really big data problems
- ▶ The issue is more of user load balancing and resource allocation (and unallocation)
- ▶ Load balancing independent jobs is parallel
- ▶ This suggested utilizing elastic computing services



eh3.uc.edu/gimm/webgimm/files/deployment.pdf

- ▶ A webserver and backend processing model
- ▶ GIMM server backends exist but are fixed to physical available machines
- ▶ Elastic cloud services can spin up and down lxc containers as needed



tctechcrunch2011.files.wordpress.com/2013/09/mesos-docker-1.png

- ▶ Created a linux container with the GIMM server modules for dynamic creation

- ▶ Attempted to setup internal cluster
- ▶ Attempts with Mesos, Cloudera and Pivotal phd could not be properly configured
- ▶ More custom method with just lxc and standard installations



- ▶ Setting up docker containers
- ▶ Automating creation and destruction of containers
- ▶ Updating WebGimm interface to control container creation
- ▶ Security of creation credentials



- ▶ Attempts to find more efficient parallel implementations of GIMM proved difficult
- ▶ Some methods are promising such as Theano
- ▶ The Container of the GIMM Server could very likely be launched in an elastic cloud for dynamic resource allocation, or internal to a University network