SNP Genetic Data Analysis

Primary Goal

Differentiate Patient and Control Groups

- Identify the genetic origins of diseases
  - Cancer, auto-immune, bone marrow failure, etc.
- Prevent sickness and death by enabling avoidance of disease triggers
- Provide direction in searches for treatments and cures

•Rapidly analyzes huge SNP genetic data sets for high scores

•Developed for and with the Cleveland Clinic

•Extensible to similar applications

CUDA Optimizations

Memory

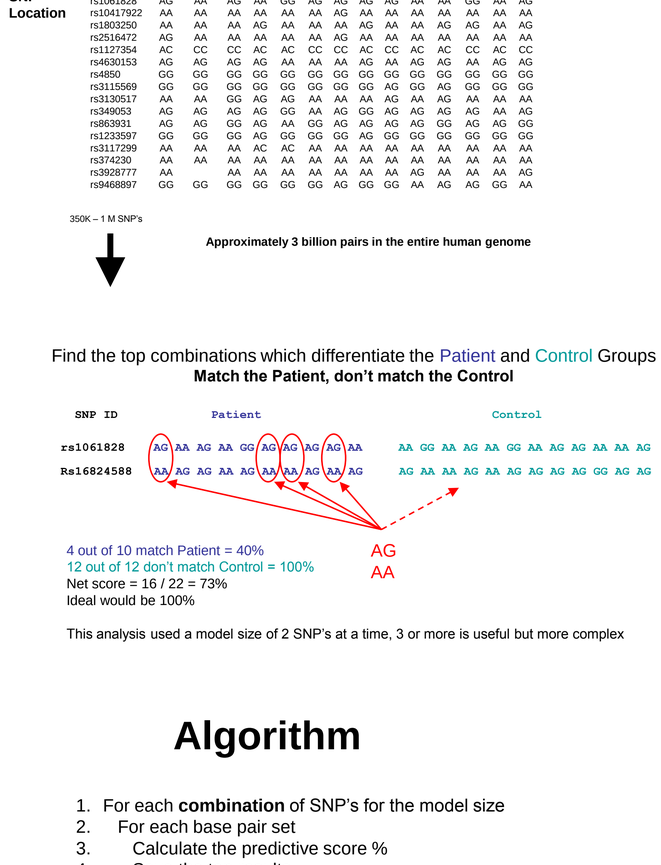
Improve Data Density

- 16 bits: “AA” is 2 chars,16 bits per data point.
- 8 bits: Map AA, AG, GG to 1,2,3 (0 = null), single char
- 2 bits: only 4 values, pack 16 values per 32 bit integer
- 4 bits: not as dense as 2 bits but may facilitate calculations

- 8X compression enables large data sets to fit in global memory
- Data transfer operations net 8X faster
- Bitwise operations do multiple compares in a single GPU instruction
  - Parallel inside parallel !

SNP : Single Nucleotide Polymorphism

Gene Base Pair at a specific location in the DNA



CUDA to the Rescue!

The problem is well suited to CUDA analysis

- **Each score is independently calculable**
  Asynchronous threading, high parallelism
- **Complete datasets fit into global memory**
  Low data transfer, high computation
- **Reuse of data**
  Leverage fast shared memory
- **Large scale important problem**
  Stop playing games, make a difference !

Threadblocks

Threadblocks are the primary organizational unit

- Leverage fast shared memory in the threadblock
  - Each TB does N values at a time
  - N^2 fast reads of the data per threadblock
- Size and number of threadblocks are runtime parameters
  - Easily scale to hardware capability

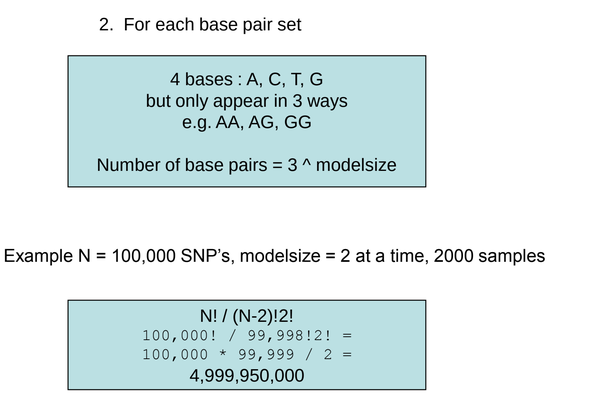
CPU / GPU Overlap

CPU manages results while the GPU is computing the next set

- No added data movement, just rearrange the same calls
- More speed, utilize fast CPU too

Find the top combinations which differentiate the Patient and Control Groups

Match the Patient, don’t match the Control

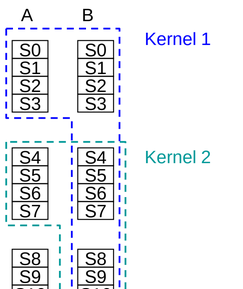


4 out of 10 match Patient = 40%
12 out of 12 don’t match Control = 100%
Net score = 16 / 22 = 73%
Ideal would be 100%

This analysis used a model size of 2 SNP’s at a time, 3 or more is useful but more complex

CPU

12 SNPs, modelsize=2



CPU Implementation

A is the 1st SNP to compare
B is the 2nd
Picture as 2 arrays, though actually
pointers into the single array

For A = 0 to 10
  For B = A+1 to 11
    For m = 0 to 3^modelsize
      Calculate score
      if score > min save result

Dynamic Results

Host maintains a low water mark (minimum score)
- Only threads which find a better score report back

- Uses Atomic Add to coordinate thread write locations

- Later kernels, very little or no data to transfer

Algorithm

1. For each **combination** of SNP’s for the model size
2. For each base pair set
3. Calculate the predictive score %
4. Save the top results

1. For each combination of SNP’s

Compare N SNP’s, modelsize at a time

Number of Combinations = N! / (N-modelsize)!modelsize!

(Grows very fast as modelsize increases)

2. For each base pair set

4 bases : A, C, T, G
but only appear in 3 ways
e.g. AA, AG, GG

Number of base pairs = 3 ^ modelsize

Example N = 100,000 SNP’s, modelsize = 2 at a time, 2000 samples

N! / (N-2)!2!
100,000! / 99,998!2! =
100,000 * 99,999 / 2 =
4,999,950,000

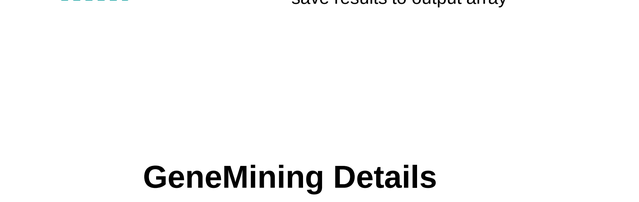
3^2 base pair sets = **9**
(AA,AA),(AA,AG),(AA,GG)
(AG,AA),(AG,AG),(AG,GG)
(GG,AA),(GG,AG),(GG,GG)

Total comparisons
SNPs x base pairs x samples

89,999,100,000,000 (= 90 trillion)

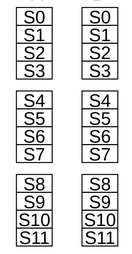
350K SNP datasets are here, 1 M on the way, larger modelsizes = Big problem . . . need a big hammer.

CUDA - Threadblocks



Group analysis into thread blocks
Use (reuse) fast shared memory
Facilitate scaling, launch blocks to optimize hardware utilization
Currently 5000 blocks / kernel
1D Grid

CUDA - Kernels



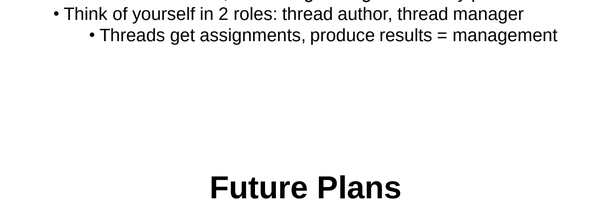
Launch a kernel per A set and maxThreadBlocks
Use CPU to check results, in parallel with GPU

```
{host}
blocksize=4
for A = 0 to SNPCount / blocksize by blocksize
  launchKernel[[[  ]]]
  for B = 0 to SNPCount by maxTblocks
    copyData()
    launchNextKernel[[[ ]]]
    analyzeCurrentKernel()

{thread}
Copy data to shared memory
if score(SNPA,SNPB,BasePair) > minpct
  save results to output array
```

SNP Comparisons per Second

Modelsize = 2



CUDA Hints and Tips

- “Steal” from the samples – that’s why they are there
- Debug data movement to and from the threads first
  - Verify each thread has its proper data and returns results
  - Keep the debug code commented out for later use
- Kernel calls are expensive – give each kernel lots to do
- Don’t be afraid to load up the threads
  - Sometimes better to have a thread do 2 things than 2 threads doing 1 each and coordinate them
- Understand the x, y and z thread scheme, it can be very helpful
- If the screen flickers, it’s not a good sign – memory pointer issue
- Think of yourself in 2 roles: thread author, thread manager
  - Threads get assignments, produce results = management

GeneMining Details

Current Settings

4 SNPs per block, 5000 Blocks per kernel
2 SNPs at a time, 350K SNP’s

Threadx = SNP A value
Thready = SNP B value
Threadz = base pair set in model

Threads per kernel = 4 threadx, 4 thready,
10 threadz, 5000 blocks = 800K threads

Notice how the x, y, and z threads correspond to the loops in the CPU code

- AMD Athlon64 X2 5000+ Black Edition
- EVGA 8600GT (32 cores, 540 MHz)
- •7X gain for CUDA
- •Newer GPU (240 cores, 1476 MHz) **140 X?**

Future Plans

Use 64 bit integers, upgrade GPU, larger modelsize
Test constant memory vs. shared memory, run Profiler
Review CUDA documentation for further potential improvements
Identify other applications which could benefit

CUDA Hints and Tips

Presenter: Rainer Richter, IBM, Cleveland Clinic

Acknowledgments :
Steve Kauffman, IBM Corporation
Dr. Jaroslaw P. Maciejewski M.D., Cleveland Clinic
Bartlomiej Przychodzen, Cleveland Clinic
Robert Wipfler
Aplastic Anemia and MDS International Foundation

(Copyright 2009)

350K SNPs printed =
4200 ft = .8 miles = 1.3 km