GPU-Meta-Storms: Computing the similarities among massive microbial communities using GPU

Xiaoquan Su[$], Xuetao Wang[$], JianXu, Kang Ning*
Shandong Key Laboratory of Energy Genetics, CAS Key Laboratory of Biofuels and BioEnergy Genome Center, Qingdao Institute of Bioenergy and Bioprocess Technology, Chinese Academy of Sciences, Qingdao 266101, Shandong Province, People's Republic of China
*ningkang@qibebt.ac.cn
$ These authors contribute to this work equally.* To whom correspondence should be addressed.

***Abstract*—With the development of next-generation sequencing and metagenomic technologies, the number of metagenomic samples of microbial communities is increasing with exponential speed. The comparison among metagenomic samples could facilitate the data mining of the valuable yet hidden biological information held in the massive metagenomic data. However, current methods for metagenomic comparison are limited by their ability to process very large number of samples each with large data size.**

In this work, we have developed an optimized GPU-based metagenomic comparison algorithm, GPU-Meta-Storms, to evaluate the quantitative phylogenetic similarity among massive metagenomic samples, and implemented it using CUDA (Compute Unified Device Architecture) and C++ programming. The GPU-Meta-Storms program is optimized for CUDA with non-recursive transform, register recycle, memory alignment and so on. Our results have shown that with the optimization of the phylogenetic comparison algorithm, memory accessing strategy and parallelization mechanism on many-core hardware architecture, GPU-Meta-Storms could compute the pair-wise similarity matrix for 1920 metagenomic samples in 4 minutes, which gained a speed-up of more than 1000 times compared to CPU version Meta-Storms on single-core CPU, and more than 100 times on 16-core CPU. Therefore, the high-performance of GPU-Meta-Storms in comparison with massive metagenomic samples could thus enable in-depth data mining from massive metagenomic data, and make the real-time analysis and monitoring of constantly-changing metagenomic samples possible.

Keywords—Metagenome; Phylogenetic Comparison; GPU; High Performance Computing

I. INTRODUCTION

Most microbes live and reproduce together as "microbial communities" in nature, meanwhile, their activities and metabolisms also have profound effect to environment. The complete genomic information of an environmental microbial community is referred to as "metagenome". As most of the microbes are not isolatable and cultivatable[1], metagenomic technology has become one of the most important and efficient methods to analyse the structures and functions of microbial communities.

A. Large-scale datasets for microbial communities

Next-generation sequencing techniques[2] have enabled the fast profiling of a large number of metagenomic data. Thus, a rapidly increasing number of metagenomic profiles of microbial communities have been archived in public repositories and research labs around the world, such as MG-RAST[3], CAMERA2[4] and NCBI[5] contain thousands of metagenomic related works with more than 10,000 samples. Therefore, it is becoming more and more important to compare microbial communities in large scale for in-depth data mining for precious biological information held in the massive metagenomic data.

B. Comparison of microbial communities

A number of methods have been proposed for comparison of different metagenomic samples mainly adopting two different approaches: taxon-based (using overlap in lists of species, genera, OTUs, and so on) and phylogenetic-based (using overlaps on a phylogenetic tree).

For taxon-based methods, many recent pyro-sequencing studies have been developed to compare samples. MEGAN[6] is a metagenomic analysis tool with recent additions for phylogenetic comparisons [7] and statistical analyses[8], however, can only compare single pair of metagenomic samples based on taxonomy without quantitative measurement , as is also the case with STAMP[9], which introduces a concept of "biological relevance" in the form of confidence intervals. Other methods, such as MG-RAST[3], ShotgunFunctionalizeR[10], mothur[11], and METAREP[12] process metagenomic data using standard statistical tests (mainly t-tests with some modifications), yet would turn out to be insufficient in accuracy[13].

In phylogenetic-based approaches, such as UniFrac[14] and Fast UniFrac[15], utilize the similarities and differences among species[16] to make phylogenetic beta diversity measurement more effective at showing ecological patterns. Nevertheless the sample size and running speed restrictions limit the extension on the rapidly increasing scale of metagenomic experiments. Recently we have developed Meta-Storms [17], a metagenomic database engine for sample searching, which also supports quantitative phylogenetic comparison of metagenomic samples. However, for many

978-1-4799-1389-3/13/$31.00 ©2013 IEEE

metagenomic samples, the CPU based computation speed becomes the bottleneck again in massive data analysis: when the number of samples exceeds several hundred, the time-cost of comparison increases to be unacceptable.

II. METHODS

In this work we have developed GPU-based metagenome comparison method, GPU-Meta-Storms, to enable the high-speed comparison on massive microbial community samples. GPU-Meta-Storms implements the scoring function algorithm of Meta-Storms [17] on Compute Unified Device Architecture (also refer as "CUDA") [18] to evaluate the quantitative similarity between metagenomic samples with high acceleration rate. More importantly, it has designed with optimizations in memory accessing, threads invocation, register allocation for the many-core architecture of NVIDIA GPU.

A. Scoring function of Meta-Storms

The scoring function of Meta-Storms [17] compares two microbial communities' structure by calculating the maximum common component of their weighted co-phylogenetic tree considering the beta-diversity, phylogenetic distance and abundance of each species. In this algorithm, initially a common binary phylogenetic tree is built, in which leaf nodes (Fig. 1, eg. node X) represent species with abundance values in two samples (Fig. 1, eg. X.P1= 30% and X.P2 = 40%), as well as the branch length indicates the evolutional distance from one species to its ancestor. We define the MIN(X) as the similarity score for a single species X:

$$MIN(X)= (X.P1<=X.P2)\ ?X.P1 : X.P2; \quad (1)$$

Then the formula Reduce(X) parses the reminding component of a node to its ancestor (Fig. 1, eg. X.Anc = N) for comparison in higher phylogenetic level multiplied by the factor of 1-Dist (Dist is the phylogenetic distance between X and X.Anc).

```
Reduce(X){
        M = MIN(X);
        X.Anc.P1 += (X.P1- M) * (1 - Dist);    (2)
        X.Anc.P2+= (X.P2 - M) * (1 - Dist);
    }
```

We further define that for an internal node N, its two children nodes are N.Left (Fig. 1, eg. node X) and N.Right (Fig. 1, eg. node Y). Then the overall similarity score of one whole branch in the phylogenetic tree can be calculated recursively by this function:

```
GetSimilarity(X)=
    { MIN(X);
      Reduce(X);                          If X is a Leaf Node
                                                          (3)
      GetSimilarity(X.Left)
    + GetSimilarity(X.Right)
      + MIN(X);                           If X is an Internal Node
      Reduce(X);
```

Therefore the overall similarity between two metagenomic samples can be calculated by GetSimlairty(R), in which R represents the root node of the co-phylogenetic tree of two samples.

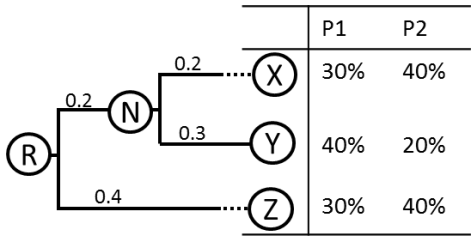


Fig. 1. An example for common binary phylogenetic tree with leaf node abundance and branch length of two metagenomic samples. P1 and P2 values are the abundance of two samples in each species.

B. Non-recursive Transformation

For the iteration depth and stack size limitation of recursive function in GPU and CUDA, formula (3) needs to be transformed into no-recursive format. Focusing on a basic binary branch with one ancestor node (Fig. 1, eg. node N) and its children (Fig. 1, eg. node X and Y), we found that the scoring function could be transformed into serial operations by poster-order traversal to the branch with following formula:

```
GetSimilaritySerial(X, Y, N) {
    MIN(X);
    Reduce(X);
    MIN(Y);                                  (4)
    Reduce(Y);
    MIN (N);
    Reduce(N);
}
```

which can be also extended to all nodes of the common phylogenetic tree by post-order traversal to all basic branches without recursive overlap to transform formula (3) into no-recursive format.

C. CUDA-based implementation

Based on the many-core architecture of GPU, formula (4) can be invocated in parallel by large number of threads of CUDA to process the calculation of similarity values among different metagenomic samples. For the synchronization of

978-1-4799-1389-3/13/$31.00 ©2013 IEEE

many-core programming, we map all samples to Greengenes coreset phylogenetic tree [19], in which inexistent species are marked by abundance of 0 . Therefore all threads can parse the sample phylogenetic tree using formula (4) for high parallelization efficiency.

To calculate the pair-wise similarity matrix of N samples, we launch N * N threads in GPU to make each similarity value in the matrix processed by one independent thread. Fig. 2 shows the process of GPU computing: abundance values of species and phylogenetic distances are loaded from the file system and initialized in RAM to build the common phylogenetic tree by CPU (Fig. 2, step 1), then sent to GPU on-board RAM for parallel computing (Fig. 2, step 2). After all threads of GPU kernel finish the tasks (Fig. 2, step 3, the key step), all elements of similarity matrix are sent back to RAM (Fig. 2, step 4), and stored into file system on hard disk (Fig. 2, step 5).

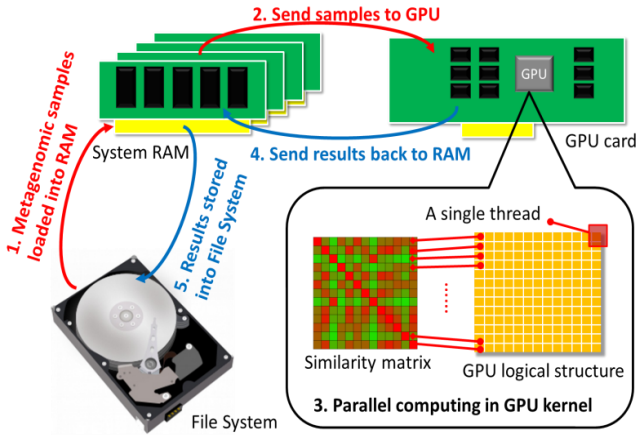


Fig. 2. Overview of the GPU based similarity matrix computing

D. CUDA-based optimization

Limited by the I/O bandwidth to the GPU on board RAM (also refer as "global memory" in CUDA), we have also designed the following optimizations to adapt the GPU architecture to improve the running speed.

1) Global memory Alignment

Since all threads calculate the same phylogenetic tree by formula (4) with same node order, their abundance values can be sorted in the same order as the leaf nodes (species) and aligned straight in the global memory (Fig. 3). Then abundance values of each species among different samples could be accessed from the same index in the global memory (Fig. 3) to accelerate both the transmission from RAM to GPU on board memory and the memory access by GPU.

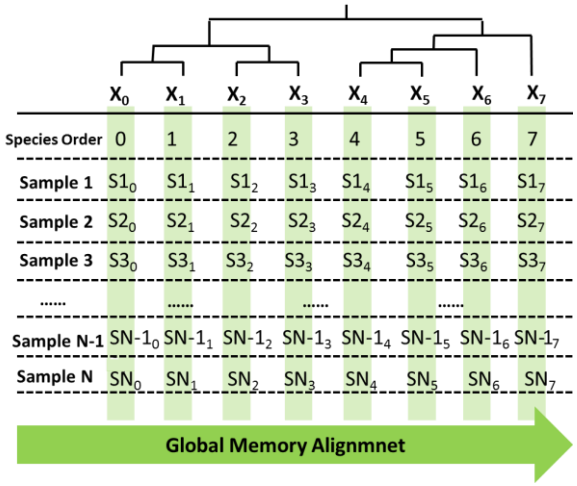


Fig. 3. Global memory alignment of abundance values

2) Register recycle allocation

In formula (4), results of each Reduce function (formula (2)) need to be added into the ancestor node (refer to formula (3) for details). To eliminate the I/O frequency to global memory, all internal nodes of the phylogenetic tree are kept into registers, of which the I/O speed is about 100 times faster than global memory. Theoretically each internal node should be assigned to a unique register, which requires a space complexity of O(N) (here N is the number of internal nodes). However, the available registers limitation of each thread (eg. ~ 128 registers for each thread in one block of Fermi GPU) is smaller than the total amount of internal nodes (eg. in Greengene coreset[19] there are >4900 internal nodes). We also found that an internal node that has been reduced by formula (3) would not be used again, and then its registers could be released to reduce the space complexity to be constant. In such, we developed a register recycle method to cut the total register number to be only 10 for the phylogenetic tree of Greengenes coreset[19].

3) Application of shared memory

For all threads calculate on the same phylogenetic tree, distance values are stored into shared memory, which can be accessed by all threads with low I/O latency, and also reduce the I/O access time to global memory in total.

III. Results& Discussions

In this work, we have used 7datasets of human habitat microbial community samples from the project "Moving pictures of human microbiome"[20] to evaluate the performance of GPU-Meta-Storms of metagenomic comparison among different amount of samples in 4 aspects: (1) GPU granularity, (2) efficiency of modules (refer to section II.C), (3) acceleration rate compared to CPU and (4) results consistency compared to CPU. All experiments in this work were finished on a rack server with dual Intel Xeon E5-2650 CPU (16 cores in total, 2.0GHz), 64GB DDR3 ECC RAM, NVIDIA M2075 GPU (448 stream processors and 6GB GDDR5 on board RAM) and 1TB hard drive in RAID 1.

978-1-4799-1389-3/13/$31.00 ©2013 IEEE

TABLE 1. THE SAMPLE AMOUNT OF EACH DATASET

Dataset	Sample amount	Total size (M) Byte)
Dataset 1	8	53
Dataset 2	64	337
Dataset 3	128	637
Dataset 4	256	1331
Dataset 5	512	2663
Dataset 6	1024	5222
Dataset 7	1920	9216

A. Granularity analysis for block size configuration

In CUDA, threads were grouped into blocks, in which computing resources such as registers, shared memory were shared among threads; therefore the computing capability of a block correlated with the thread number in it. We selected 4 largest datasets from Table 1 (Dataset 4, 5, 6 and 7) to test the running time in different block configuration with (4*4), (8*8), (16*16) and (32*32) threads. In this test only the GPU running time was recorded excluding the data transfer time from file system.

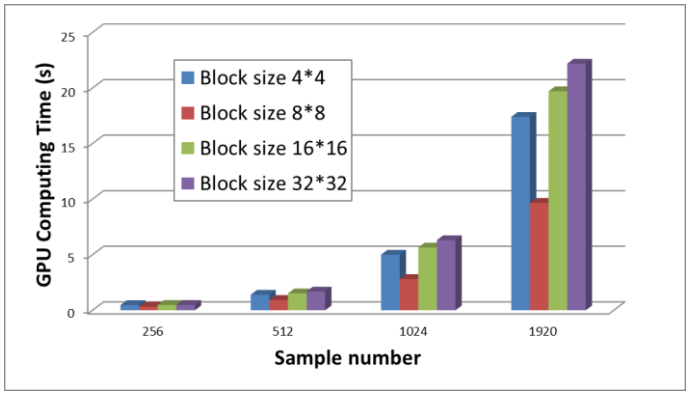


Fig. 4.Similarity matrix computing time of GPU with different block size

Results had shown that for various amounts of input samples, block size of (8*8) cost the least time. The reason was that although each block supported maximum thread number of 1024 in CUDA, registers' total size was restricted to be 32768 Byte for all threads. Since for the calculation of formula (4), a single thread needed 21 variables of 152 Byte space (Table 2), block size of (8*8) could efficiently use the computing resources, while smaller size blocks (4*4) wasted the computing capacity and larger blocks (16*16 and 32*32) rotated the threads due to the insufficiency of register spaces (Table 3).

TABLE 2. THE REGISTER USAGE OF A SINGLE GPU THREAD FOR CALCULATION OF FORMULA (4)

Variable purpose	Type	Number	Size (Byte)
Phylogenetic tree Parsing	Double	10	80
Abundance value accessing	Double	4	32
Abundance value reduce	Double	2	16
Loop control	Integer	1	4
Thread ID	Integer	3	12
Result	Double	1	8
Total		21	152

TABLE 3. THE REGISTER SIZE FOR EACH THREAD OF DIFFERENT BLOCK SIZE

Block size	Thread amount	Register size (Byte)
4*4	16	2048
8*8	64	512
16*16	256	128
32*32	1024	32

B. Efficiency of each module

Then we focused on the time cost of 5 modules of GPU-Meta-Storms illustrated in section II.C and Fig. 2 including input loading of metagenomic samples, RAM to GPU data transfer, GPU kernel computing, GPU to RAM data transfer and results save to file system time, which could also detect the bottle-neck of the system. We computed the similarity matrix of all 7 datasets in Table 1 with block size of (8*8) that is considered as the most optimized configuration in section III.A, and timed all 5 modules of GPU-Meta-Storms.

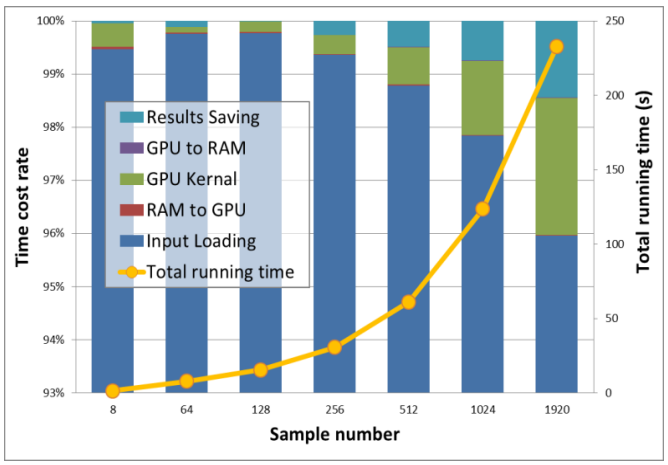


Fig. 5. Time cost rate of each module of GPU-Meta-Storms

In Fig. 5 bar-charts represented the time cost rate of modules of GPU-Meta-Storms for processing each dataset. In GPU-Meta-Storms, 5 modules were affected by 3 factors: a. File system bandwidth (for input loading and results saving); b. PCI-E bandwidth (for data transfer between GPU and RAM) and c. GPU kernel computing ability. Obviously in Fig. 5 that input loading and results saving took more than 98% time due

978-1-4799-1389-3/13/$31.00 ©2013 IEEE

to the low average bandwidth of the file system (file system bandwidth was calculated by dataset size / loading time), which was only 43.32 MB/s. We also checked the time cost of data transmission between RAM and GPU via the PCI-E bus, and calculated the average PCI-E bus bandwidth was 2.25 GB/s, which was significantly higher than the file system. As the GPU kernel computing cost only less than 2% time on average during the test, the file system bandwidth can be considered as the bottle-neck in the system. In addition, since our file system was based on HDD (Hard Disk Drive), and the bottle-neck condition could be improved by replacing HDD by SSD (Solid State Disk) which provides much higher bandwidth.

C. Running time comparison with CPU-based computation

In this work, we also used GPU and CPU to compute the similarity matrix of all datasets in Table 1 to show the acceleration rate of GPU-Meta-Storms. For CPU computing, we tried both the single core and multiple cores (with 16 threads) and for GPU we used the block size of (8*8). Additionally we recorded the time cost of entire program including all components described in section III.B.

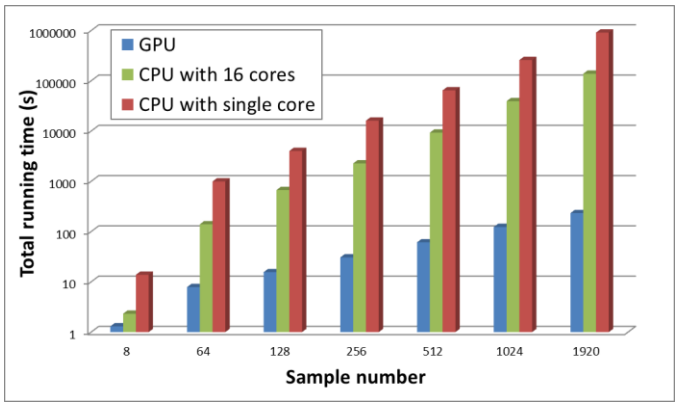
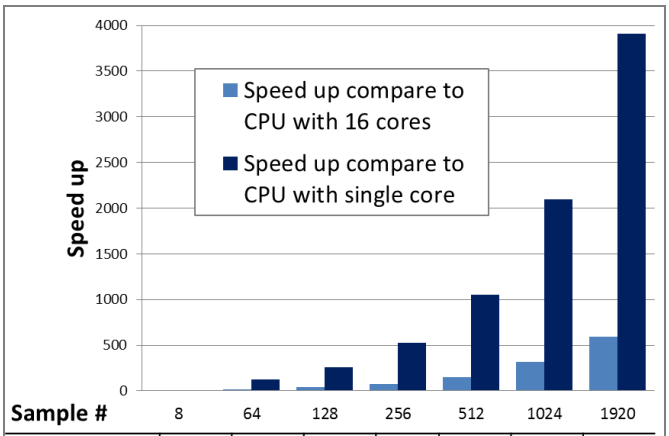


Fig. 6. Overall running time of similarity matrix computing by CPU and GPU.



Sample #	8	64	128	256	512	1024	1920
Speed up to 16 cores CPU	1.7	18	43	74	152	318	593
Speedup to 1 core CPU	11	127	257	525	1056	2093	3905

Fig. 7. Speed up of GPU compared to CPU

Form the results of Fig. 6 and Fig. 7 we can observe that to build out the same similarity matrix GPU had a maximum speed up of 3905 times compared to single core CPU and 593 times to 16-core CPU, which made the similarity matrix of 1920 metagenomic samples can be constructed within 4 minutes by GPU-Meta-Storms on Tesla M2075. Since the computing was in parallel, the running time did not linearly increase with the sample number as CPU, but depended on the throughput of GPU thread scheduling and memory access. Therefore dense computing with large number of samples approached the efficient usage of resource, which made the higher speed up rate of large number of samples. Furthermore, the high acceleration could not only largely eliminate the running time in computing the similarity of metagenomic data, but also enabled in-depth data mining among massive microbial communities.

D. Consistency of results between GPU and CPU

Considering the similarity values computed by CPU as the standard, we compared the results of all 7 datasets in Table 1between GPU and CPU based methods to evaluate the accuracy of GPU-Meta-Storms. For each dataset, the error ranges with average differences (absolute values) of each dataset were illustrated in Fig. 8.

978-1-4799-1389-3/13/$31.00 ©2013 IEEE

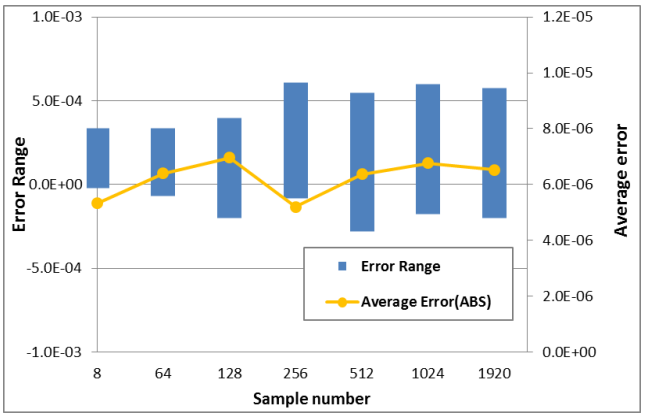


Fig. 8. Inconsistency and average error between CPU and GPU computing

Although hardware difference for float number computing (eg. Float Point Unit, FPU) caused the inaccuracy between CPU and GPU, supports of double float computing in GPU CUDA minimized their inconsistency. From the result of Fig 8 we can observe that the inconsistency (error range) between CPU and GPU computing could be controlled into a restricted range and gotten an average error of 6.22E-06, which elucidated the reliability of GPU-Meta-Storms.

IV. Conclusions

With the number of metagenomic samples increased rapidly, analyzing the large volume of these data quickly faces the bottleneck of computation efficiency. In this work we proposed an optimized metagenomic comparison algorithm based on GPU and CUDA to calculate the similarities among a large of microbial community samples with very high speed. The GPU-Meta-Storms program is optimized for CUDA with non-recursive transform, register recycle, memory alignment.

A large number of human microbial community samples have been collected and compared, and our tests have shown that this GPU algorithm reduces the computing time largely with a speed of more than 500 times compared to the 16 cores CPU parallel program for 1920 samples. Specifically, the pair-wise similarity matrix of 1920 metagenomic samples to be completed within 10 minutes, so that clustering based on such a large set of samples could be made. Therefore, acceleration techniques based GPU make it possible to perform in-depth data mining in massive metagenomic samples, as well as make the real-time analysis and monitoring of constantly-changing metagenomic samples possible.

V. Material Availability

Source code of GPU-Meta-Storms and CPU version Meta-Storms with complete manual are available at http://www.computationalbioenergy.org/meta-storms.html.

All 7 datasets of Human habitat microbial community samples for test in this work can be accessed from ftp://www.computationalbioenergy.org/Meta-Storms/GPU.

References

[1] A. Jurkowski, *et al.*, "Metagenomics: a call for bringing a new science into the classroom (while it's still new)," *CBE Life Sci Educ*, vol. 6, pp. 260-5, Winter 2007.
[2] E. R. Mardis, "The impact of next-generation sequencing technology on genetics," *Trends Genet*, vol. 24, pp. 133-41, Mar 2008.
[3] F. Meyer, *et al.*, "The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes," *BMC Bioinformatics*, vol. 9, p. 386, 2008.
[4] R. Seshadri, *et al.*, "CAMERA: a community resource for metagenomics," *PLoS Biol*, vol. 5, p. e75, Mar 2007.
[5] http://www.ncbi.nlm.nih.gov/
[6] D. H. Huson, *et al.*, "MEGAN analysis of metagenomic data," *Genome Res*, vol. 17, pp. 377-86, Mar 2007.
[7] S. Mitra, *et al.*, "Comparison of multiple metagenomes using phylogenetic networks based on ecological indices," *ISME J*, vol. 4, pp. 1236-42, Oct 2010.
[8] S. Mitra, *et al.*, "Visual and statistical comparison of metagenomes," *Bioinformatics*, vol. 25, pp. 1849-55, Aug 1 2009.
[9] D. H. Parks and R. G. Beiko, "Identifying biologically relevant differences between metagenomic communities," *Bioinformatics*, vol. 26, pp. 715-21, Mar 15 2010.
[10] E. Kristiansson, *et al.*, "ShotgunFunctionalizeR: an R-package for functional comparison of metagenomes," *Bioinformatics*, vol. 25, pp. 2737-8, Oct 15 2009.
[11] P. D. Schloss, *et al.*, "Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities," *Appl Environ Microbiol*, vol. 75, pp. 7537-41, Dec 2009.
[12] J. Goll, *et al.*, "METAREP: JCVI metagenomics reports--an open source tool for high-performance comparative metagenomics," *Bioinformatics*, vol. 26, pp. 2631-2, Oct 15 2010.
[13] M. Hamady and R. Knight, "Microbial community profiling for human microbiome projects: Tools, techniques, and challenges," *Genome Res*, vol. 19, pp. 1141-52, Jul 2009.
[14] C. Lozupone and R. Knight, "UniFrac: a new phylogenetic method for comparing microbial communities," *Appl Environ Microbiol*, vol. 71, pp. 8228-35, Dec 2005.
[15] M. Hamady, *et al.*, "Fast UniFrac: facilitating high-throughput phylogenetic analyses of microbial communities including analysis of pyrosequencing and PhyloChip data," *ISME J*, vol. 4, pp. 17-27, Jan 2010.
[16] C. H. Graham and P. V. Fine, "Phylogenetic beta diversity: linking ecological and evolutionary processes across space in time," *Ecol Lett*, vol. 11, pp. 1265-77, Dec 2008.
[17] X. Su, *et al.*, "Meta-Storms: Efficient Search for Similar Microbial Communities Based on a Novel Indexing Scheme and Similarity Score for Metagenomic Data," *Bioinformatics*, Jul 26 2012.
[18] http://www.nvidia.ca/object/cuda_home_new.html
[19] T. Z. DeSantis, *et al.*, "Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB," *Appl Environ Microbiol*, vol. 72, pp. 5069-72, Jul 2006.
[20] J. G. Caporaso, *et al.*, "Moving pictures of the human microbiome," *Genome Biol*, vol. 12, p. R50, 2011.

978-1-4799-1389-3/13/$31.00 ©2013 IEEE