Speed Acceleration with Graphics Processing Unit in Protein Database Search Using Isotopic Envelope Fingerprinting

Your N. Here
Your Institution

Second Name
Second Institution

Abstract

Liquid chromatography mass spectrometry(LC-MS)-based proteomics is currently the major horse power for highly sensitive qualitative and quantitative characterisation of large proteome systems consisting of hundreds and thousands of proteins and proteoforms.

1 Introduction of GPU

The graphics processing unit (GPU) has become an integral part of today's mainstream computing systems. Over the past six years, there has been a marked increase in the performance and capabilities of GPUs.The modern GPU is not only a powerful graphics engine but also a highly parallel programmable processor featuring peak arithmetic and memory bandwidth that substantially outpaces its CPU counterpart. The GPU?s rapid increase in both programmability and capability has spawned a research community that has successfully mapped a broad range of computationally demanding, complex problems to the GPU.

The two major GPU vendors, NVIDIA and AMD, recently announced their new developing platforms, respectively CUDA [7] and CTM [8]. Unlike previous GPU program- ming models, these are proprietary approaches designed to allow a direct access to their specific graphics hardware. Therefore, there is no compatibility between the two plat- forms. CUDA is an extension of the C programming language; CTM is a virtual machine running proprietary assembler code. However, both platforms overcome some important restrictions on previous GPGPU approaches, in particular those set by the traditional graphics pipeline

In CUDA, the GPU is viewed as a compute device suitable for parallel data applications. It has its own device random access memory and may run a very high number of threads in parallel (Figure 2). Threads are grouped in blocks and many blocks may run in a grid of blocks. Such structured sets of threads may be launched on a kernel of code, processing the data stored in the device memory. Threads of the same block share data through fast shared on-chip memory and can be synchronised through synchronisation points. An important aspect of CUDA programming is the management of the memory spaces that have different characteristics and performances.

More fascinating text. Features[1] galore, plethora of promises.

2 Related Work

ProteinGoggle is a tool which described in Zhixin Tian's paper[x] which provide a new way to implement search engine for top-down intact protein database searching.

In his paper, Protein database searches with iMEF and ProteinGoggle utilize raw MS data, i.e., iEs of both precursor and product ions, for fingerprinting and database search, and the deisotoping step used in other mass fingerprinting algorithms is bypassed. With the development and wide availability of mass spectrometers of high mass resolution and mass measurement accuracy, we expect that iMEF and ProteinGoggle will find wide application. Based on Net Framework 4.0, the ProteinGoggle program has been written in C with Client/Server (C/S) infrastructure and MySQL Server 5.0 as the data system. The code uses the standard three-layer structure, i.e., User Interface/Business Logic Layer/Data Access Layer (UI/BLL/DAL). A standalone ProteinGoggle 1.0 software package for computers running on Windows operating systems is freely available to academic users upon request.

However, The speed of calculation protein's fingerprinting(iMEF) is a large calculation workload which always costs several days to calculate a 200-250 sequence protein.

describe the algorithm

Thus, ProteinGoggle's range of application is limited by its speed so that it is necessary to accelerate the calculation speed.

3 Work

Experiment have been performer to accelerate the speed of Protein Isotopic Envelope Fingerprinting database computing which used to compare the similarity level of mass spectrum. We compared the the implementation of ProteinGoggle in CUDA with:

* The result of former resort was implemented in single PC which is with corei7/8 core cpu.

We have implemented our solution on a workstation which has the xxxx GHz Intel processor, one NVIDIA GT760 graphic cards and 8Gb memory. The former programs are written in c which support easy-use GUI for most of users. In order to make the performance Optimization as simple as we can, we just change the core algorithm and strategy of calculation in our new program which based on GPU calculation in CUDA framework.

3.1 CUDAFY Framework

CUDAfy .NET allows easy development of high performance GPGPU applications completely from the Microsoft .NET framework. It's developed in C. The old software are based on C so that we have to call GPU calculation program in C runtime environment. CUDAFy is a framework which can support user program in c to call core program of calculation in GPU with CUDA framework. It makes CUDA programming even more convenient than with Nvidia's C-based runtime.

3.2 IPC Library

IPC is a library to calculate one Molecular's distribution probability based on each element's distribution probability in nature. OpenSource IPC Library is written by Java Programming language.

3.3 Old Strategy

In order to calculate a Molecular's distribution probability of m/z, IPC library will add probability of each elements one by one, which means it will get a large sequence of m/z and probability key value pair which's scale is defined by number of elements.

Program will get a list as result which contains a list of m/z and probability key value pair. Each calculation step is add m/z with probability to result get in former step, which means the scale of list will be very large if program have run large number steps. However, program will combine probability result if m/z are same of every two element in result list. Besides, in order to confine the computing scale, IPC set a threshold to limit size of result in each step which can effectively decrease computing complexity of original algorithm.

For example, if we calculate C1000H4000O300N400,the complexity is about 200*200*N, N is number of element. In this case N is 1000+4000+300+400.

There should be several molecular from a protein sequence shear. In general, a 200-250 length sequence will be shear to 800 900 molecular. There also are about 3000 protein sequence in a single file about 14mb. Above all, the complexity is very large.

3.4 New Strategy

Obviously, the old strategy has much redundant computing in calculating several molecular. The C1000's peaks result will be calculated twice if there have C1000 and C1001 molecular. This situation has high frequency in ProteinGoggle computing.

We put forward a new way to calculation large amount of molecular's m/z and probability key value results. If a molecular which is C1000H2000O300N300 should be calculated, we can calculate C1000, H2000, O300 and N300's result then save results in a data structure. Then, we calculate the matrix combinations of each results and get the top N results.

The advantage of this new strategy is we don't need to calculate redundant result but save it in database before. If we need it , we extract it from memory. We call it middle result.

4 Evaluation

We have run test with a 200-250 sequence size protein file with is as large as 14mb. In former strategy and workload, it will cost about 13 days in database calculation.

With our new strategy and algorithm, it only costs about 1.5 hours. This result are tested in our workstation which described before.

For more details, the diagram shows the performance tracks:

5 Conclusion

5.1 Future Work

Notes

[1]Remember to use endnotes, not footnotes!