Vrije Universiteit Brussel

Proposal bachelor thesis

Title: Massively Parallel Gene-Sequence Alignment

Promotor: Wolfgang De Meuter

Includes preparation course: Yes

Context

There is an undeniable push towards more parallel and distributed computing in many applications, especially within scientific applications. The need for larger-scale data analysis has exploded in recent years, making many traditional sequential programs obsolete. We know how to make regular and dense numerical computations run fast on multicore processors and computer clusters. However, large classes of applications exist that do not fit the mold of the traditional High Performance Computing or Big Data paradigms. In order to keep up with the exploding data and compute resource requirements, it is vital for these types of *irregular* parallel applications to explore alternative parallel computing paradigms.

Gene sequencing is the process of reconstructing the DNA sequence of an individual in order to map genetic differences. The applications of gene sequencing are legion and range from identifying genetic deceases to personalized medicine. A drop of blood is fed to a sequencing machine, which cuts up the DNA string in small parts of about a hundred base pairs (G, A, T or C) and reads these to produces a digital string. The output of a sequencing machine is thus gigabytes of short strings. The first step of any gene sequencing application is to reconstruct the original DNA. Reconstructing DNA with only the short DNA read fragments (de novo assembly) is very expensive, as it has a bad time complexity. More common is sequence alignment or mapping, where each short read is matched against a large reference genome. This type of algorithm performs a type of string matching and has a much lower time and space complexity. It is also massively parallel, because each read can be aligned separately and simultaneously.

In earlier work, we observed that this gene-sequencing program has bad performance characteristics on regular processor hardware. On the surface the problem is obviously massively parallel, a billion of short DNA strings can be matched simultaneously. You can thus throw more processors on the problem and expect a nice linear speedup. However, this is not the entire picture. Zooming in on the actual execution inside a single processor one can observe that the majority of its time is spent sitting idle, waiting for memory accesses. We measured that 74% of the memory access instructions go directly to the very slow DRAM, which is about 300 times slower than the processor cache memory.

The **Cray XMT** is unique computer architecture that takes multithreading seriously. Rather than relying on caches to speed up the average memory access, it relies on doing other work when an instruction needs data from main memory. In order for this to work, it needs enough of such 'other work' lying around. In other words, it relies on running a large number of virtual threads to hide memory latency. To illustrate, the XMT supports up to 8192 simultaneous execution threads. This way, it can deal with long memory latencies, without relying on data locality. As a result, the programmer does not have to cut up, distribute and keep data local over the many processors; the XMT's entire 2TB of memory is global and shared over all processors. The current killer application for the Cray XMT is graph analysis[1], another extremely parallel yet irregular parallel problem. We have good reasons to believe that gene sequencing can be the XMT's next big killer application.

Proposal bachelor thesis

For this bachelor thesis you will investigate the parallel execution of the gold standard gene-sequencing alignment algorithm on alternative parallel hardware: the *Cray XMT*. Concretely, this means you will create a port of the existing C source code for the XMT, which supports regular C, but includes additional operations specific to the XMT. The thesis work consists of three distinct stages: straight port, process-level parallelism and cluster-level parallelism.

In the first stage you will familiarize yourself with the XMT programming environment and produce a prototype XMT port of the BWA sequencing program. To reduce complexity at this stage, the scope here is limited to a single compute node. Possible issues with the BWA C-source code in the XMT programming environment will be ironed out, without producing XMT-specific code. The priority in this first stage is thus to produce a binary that runs, but which is not necessarily parallel or fast.

In the second stage, you focus on exploiting the single-node multithreading capabilities of the XMT's ThreadStorm processor. Here, XMT-specific operations and intrinsics are added to the source code, replacing the existing native pthreads-based BWA threading implementation. The goal of this stage is to improve performance, scalability and bandwidth benchmark results; directly comparing the single-processor ThreadStorm results to a single Intel Xeon multicore processor.

In the third and final stage, you will scale out and adapt the XMT-specific BWA program to run on the full Cray XMT setup, utilizing multiple compute nodes. This involves optimizing for the various memory, data movement and storage subsystems: memory mapped shared global memory, Cray LUC, fast asynchronous I/O, Lustre FS.

[1] http://yarcdata.com/blog/?p=182

Preparatory course bachelor thesis

As preparatory work you will familiarize yourself with two new computer science domains: bioinformatics and high-performance computing. For bioinformatics you will study a variety of sequence alignment algorithms, in order to have a good view on the structure, purpose and computational nature of the problem. You will also need to gain insight on modern computer architectures, which naturally will include an in-depth study of the XMT. Of equal importance is gaining a general understanding of parallel and concurrent programming concepts. The literature study for this preparatory work will be guided by the advisor, which will supply book chapters and necessary documentation.