Designing Advanced Software Architectures for High Throughput Instruments

A conversation with Andrew Dallas, CTO, Full Spectrum Software

Throughput

In the gene sequencing market, there seems to be enormous competition to achieve the fastest possible throughput. Can you talk about that and the implications for other scientific instruments that are also attempting to achieve the highest possible throughput?

Andrew

Well, as everyone knows, we have the X Prize which will be awarded to the first organization to sequence the human genome for $100 or less. A lot of what's driving the price right now is the volume of information and in some respects the physical processing of genetic material. So the high volume of information and the technologies that are in use today are depending on speed to get that price down. The price is coming down quickly, but we're not quite there yet. Fundamentally the problem of processing this amount of information is ubiquitous in many domains including gene sequencing, image processing and others within the scientific community.

There are a number of approaches that are being taken to achieve a higher throughput for information processing, data crunching and I'll talk more about that a little later. Going back 15 years ago when we were working on electrophoresis gel imaging, we were looking for very specific markers, but the genetic material, which was on large plates had to move a long distance through a gel in an electric field and it took a very long time. That technology is still in use today, albeit on a much smaller scale, so that movement of material is reduced to a very short distance and the amount of material is dramatically reduced.

Today you have nano-fluidics and nano-chips to dramatically reduce the size and volume which allows you to do single molecule sequencing. Helicos is using a technology where they build strands within a plate, which is not electrophoresis, it's a different process. Applied Biosystems has the SOLiD system, where they are using bead technology.

So there are a number of technologies out there to deal with physical processing. However, even when you get past the physical aspects of the systems, you are still left with an enormous amount of data in this instance in the form of base pairs in a long sequence.

Historically, there was a very linear approach to putting that information together. There have been a number of ways of applying very clever algorithms, using bio-statistical matching techniques to take large blocks of information and effectively find matches and speed up the process using these large blocks as opposed to the individual components. So, what Full Spectrum Software has done in gene sequencing and in other domains is really look at pipelining technology and parallelism. Pipelining and parallelism are methods, architectures and design patterns to perform processing in a parallel manner, executing many tasks in the same period of time.

This has really become effective since the advent of multi-core systems where you can have 8 cores or more. That has given us a very real capability for parallel processing. Prior to that there was pre-emptive multi-tasking where you would create two parallel processes but only one would be processing at a time. So it would appear as though both processes were happening at the same time, but in fact, they were not.

Preemptive multi-tasking was effective in achieving a smooth functional user interface while you were monitoring hardware in the background to see if you were on to the next step of a process or needed to display some sort of state change. Now we're doing true parallel processing on the front end and the back end and throughput is accelerating accordingly. One does have to be careful about contention between processes. There are techniques for avoiding collisions and deadlocks where you're sharing resources between multiple cores that are efficient and effective. Those can of course present points of friction and it's something that we're very careful about in our designs.

So preemptive multi-tasking was the first step in moving things forward in terms of being able to do multiple things at the same time but that didn't pay off much in terms of throughput. Multi-core parallel processing did. In parallel processing you're using multiple CPUs. Systems can have multiple chips which potentially can have multiple cores in them allowing for large scale parallel processing and the Cloud is beyond that still.

Graphics Processing Units

So you've covered the evolution of CPUs, what about the increasing popularity of utilizing GPUs for increased throughput?

Andrew

We're seeing a lot of interest now in using GPUs; it's really gaining a lot of traction. GPU processing is achieved by using SDKs from the GPU manufacturer. Perhaps the most popular is CUDA which is produced by Nvidia. It allows the engineer to write blocks of code that are installed on GPUs and can be triggered by data that's fed into the GPU at any time. Since the processing is occurring on a separate processor, it takes effectively zero time, from the CPUs. The fundamental difference with GPUs is that they can perform hundreds of parallel processing tasks simultaneously. By utilizing GPUs, we're seeing performance increases of tenfold to one hundred fold, depending of course on what we're doing. Another nice thing about GPUs is that they are relatively inexpensive

and you can install multiple GPU's on one system. So now we can have hundreds, if not thousands, of parallel processing procedures running simultaneously.

So pipelining is a method of "feeding" these available parallel processors, whether they are multi-core, GPUs or a combination thereof to optimally push information through the system, to find the path of least resistance for processing to come out with the fastest results. I believe that the company that implements the optimal pipelining for their system, leveraging GPU technology, if not Cloud and GPU technology is going to be the company that wins the X Prize.

When we design these types of systems, the biggest engineering challenge and the area we see other companies make mistakes is in deadlocking, resource management and effectively reducing friction as much as possible. There are tools and techniques that we use at Full Spectrum Software to identify those points of friction and to identify processes that are optimal or as close to optimal as possible for building that pipeline and for managing those resources. We are bringing those to bear in multiple domains, gene sequencing is certainly one of them, but image processing, signal processing, spectral analysis or any lab processing where you have a high volume of information that needs to be processed in a complex or high volume way.

Techniques

From a software engineering standpoint, can you describe the kinds of techniques that Full Spectrum utilizes to optimize your processes?

Andrew

The way I conceptualize friction in software engineering is when you have two parallel processes that need to use the same resource, either handing one resource off from one process to another process, or periodically needing to access that same resource simultaneously. Since they are likely to want that resource at the same time, you want to have access to that resource isolated so that only one process can access that resource at one time. Further, you don't want to hang on to that resource for a very long period of time. So you write your code in such a way, you design your system in such a way that these shared resources are correctly isolated, or mutexed properly and so the mutex is held for a very short period of time. If for some reason there is an exception that occurs during that processing or error or that core goes down, that mutex must be released instantly and in every case in order to prevent deadlocks and reduce friction or even system failures.

Best Practices

Are there any other best practices that you can describe?

Andrew

The other side of this is in resource management. This is more of a pipelining issue, but it applies in parallelism as well. When you have resources at your disposal, whether that's network bandwidth, memory, CPUs, GPUs or individual cores you must have a mechanism, a design pattern within your system that is going to allow the greatest parallelism and the greatest leveraging of those resources. Balancing workload between those cores or GPUs or threads can be a very challenging issue. If you can derive a unit of measure, complexity for example, for a particular type of task and distribute those tasks equally among multiple GPUs you'll end up with a greater throughout than just having all those tasks as being weighted equally. As an example, you might have one GPU assigned 4 tasks, another has 3 tasks and another has 4 tasks.

Well that sounds reasonably well balanced. However, what if the tasks on that first GPU are very lengthy, heavy weight algorithmic processing and the other GPUs have light weight processing, the second two GPU's will be finished with their tasks and will be sitting idle, while the first GPU may be just finishing the first of 4 tasks. When I talk about resource management and pipelining that's the type of issue I'm referring to.

Weighting Tasks

Is there a methodology to divide and weight those tasks?

Andrew

The methodology is based upon on the complexity of the tasks themselves and what we've found works well is that rather than just analyzing code through visual inspection we do a calculation of cyclomatic complexity using our internal tools as well as performance analysis using commercial tools to allow us to understand the actual weight of algorithms. In that way we can provide a heuristic for dividing tasks among multiple processors. Using those techniques together is very effective.

When we're considering the design of new, highly parallel systems, a platform to consider is Apple with its new systems that feature multi-cores and its new optimized SDK. The new OS X 10.6 Snow Leopard actually has a programming model and system services called Grand Central Dispatch (GCD) which helps manage the reduction of friction by building parallel processing tools and services directly into the OS. Apple is also taking advantage of the power of GPU's in Snow Leopard with a technology called OpenCL, which is a new API, language and runtime. I believe those systems may be targeted directly at the scientific community because of their sheer processing power along with Apple's well deserved reputation for very high quality and its ease of use.

Optimization

Is optimization in parallel processing done in an iterative fashion?

Andrew

It is. For example, when you are attempting to re-optimize an algorithm through re-structuring or re-coding, you have to "re-weight" your algorithms, you have to re-analyze them for performance. This generally happens when someone is trying to optimize a kernel of code to begin with. So the next logical step, of course, would be to re-weight the processes. However, it is something than can be, and often is, overlooked.

CUDA

Can you talk a little more about CUDA specifically?

Andrew

CUDA has been out for a couple of years. The SDK has matured and improved. There are many more developers that now have expertise with CUDA. In addition, Nvidia continues to make hardware advances with the board. It's a public SDK and there are a number of good guides and coding examples that are available. Unlike OpenCL, CUDA only runs on Nvidia chips. Now your mileage may vary in terms of performance improvement, of course, but you should see something on the order of a ten-fold improvement for core processes. The features of CUDA used to be more specialized towards matrix translation oriented tasks, but now they are much more general purpose. The source of CUDA comes from the incredible power of these graphics processing chips, which were originally designed for games which are very processor intensive. Today, nearly all systems have powerful display adapters, GPUs, which are basically sitting idle. So you may have an instrument that's displaying graphs or images, but it's not even close to stressing that chipset. Seeing this, Nvidia created an SDK which really increased their value in the marketplace. So there are many software development groups now working with this and here at Full Spectrum Software, we have had great success leveraging this resource.

For example, Nvidia recently released a new version of their developer toolkit. This new toolkit lets developers take advantage of innovations in the Fermi architecture. For example, it includes performance acceleration in ray tracing, physics, finite element analysis, high-precision scientific computing, sparse linear algebra, sorting, and search algorithms and that makes these new GPUs extremely well suited for scientific applications. This next generation CUDA architecture is the most advanced GPU computing architecture ever built. Fermi basically delivers supercomputing features and performance at about 1/10th the cost of traditional CPU-only servers.

Leverage Existing Code

If I'm an engineering manager in the life sciences industry, how can I take advantage of these advances in hardware and software methodologies with my own existing code bases?

Andrew

Well, if I can re-phrase that question a bit, the issue is really how to speed up the performance of my instrument if my competitor has come out with a faster, or perhaps more accurately, a significantly higher throughput device. There's really no single answer. There are a range of options and I'll give you some examples. The simple answer is that you put it on much faster hardware and that's effectively free from a software engineering standpoint. Perhaps you have to do some recompilation if it's an embedded system but depending on the chipset, that's the simplest solution.

The next step really has to be an analysis of the code that is executing. We recently received code from a client who wanted to achieve much higher throughput in an area of code that was very complex because it was performing very complex mathematical transforms. However, it was code that had been written in the nineties, the original developers were long gone, but there are new mathematical processing techniques, new processing techniques, that are simply faster, better and more accurate. Once we analyzed this code it became obvious that this code could simply be re-structured. In this example, we were able to rebuild the application to achieve much, much higher performance. We didn't need to utilize parallelization, it wasn't necessary to achieve the required performance.

The next step, if that were not the case, is to execute a structured walk through of the source code looking for things like Knuthian hotspots and see if you can unwind those and that's a technique for optimization. If you start to get into parallelization that's whole different ball game. If your product is really a sequential type of product without isolated work functions, you're probably looking at re-writing that code. If the product is written in a language that is not directly supported by CUDA, there's a strong likelihood you're looking at a re-write. If you're changing platforms, then of course you're going to be looking at a re-write. However, there are ways of hooking into older systems, for example through a TCP/IP conduit where you could have a product running in something like Perl or Python and communicate through a communication pipe to a service that is actually parallelizing the workload. That way you might not have to re-write your whole system. You are just taking advantage of a new service that is going to do the heavy lifting.

However, the algorithms that have been developed by your organization that are your intellectual property, the real pride of the company so to speak, of course, you certainly would want to consider rewriting those. If that is in fact the case, you really may want to consider a re-write anyway. If you're writing in an older Windows technology such as MFC/C++ you can take advantage directly of CUDA or multi-threading or parallel processing without major overhead, without having to put the effort into a huge re-write.

Again this assumes that the architecture of your system is such that the heavy lifting is isolated and there is the capability of pipelining within your system.

So as you can see there are many options to achieve the results, but it all starts with an analysis of the system.

Implications for Software Quality Assurance

If a scientific instrumentation company does a complete re-write, taking advantage of all the issues you just talked about, parallelism, GPU's and so forth, what are the implications for QA? In other words, does one test these highly optimized systems in a different fashion?

Andrew

Interestingly, it can reduce the cost of QA because the cost of processing time is reduced. It can also reduce the cost of hardware because rather than having to run a dozen machines in the QA lab, many more processes can now be run on single platforms in the same period of time. So it can actually reduce QA overhead. If you have done a complete re-write one of the issues to consider is whether you're looking for identity, whether you are expecting exactly the same output from the new system as the old system. In those cases QA can develop tests that can compare the output of the two systems and if the output is identity, then you know the systems are working properly and QA's workload is reduced. If the results are somewhat fuzzy because of mathematical rounding then you may have to come up with heuristics to determine if the results are "right enough" or not quite "right enough" based on the original system. You have to find out where there are differences, which system is better and whether changes have to be made to make them more similar or whether the differences are actually an improvement in the system which can be quite likely. In terms of QA tools, there's very little difference.

Today's automation tools are very good, they will simply work faster and often you can use your same automated QA tools. A lot of what we're doing these days at Full Spectrum is building unit tests into the software development process to run those processes during the course of the development of the product and that reduces the overhead on SQA.

Generally speaking it's an opportunity to reduce the QA overhead, the hardware overhead and the capital investment if you can really take advantage of the power of these inexpensive GPUs.

Learning Curve

What is the learning curve associated with becoming proficient at parallel processing development methodologies?

Andrew

Becoming efficient at parallel processing is a challenge for many developers. It is a different way of thinking. For someone who is proficient at multi-threading and is familiar with the concept of mutexing and deadlock those developers will be able to pick up on the new systems and adapt rather easily. The biggest challenge in parallel processing is de-bugging the systems because the predictability of the systems is significantly reduced. You don't know precisely when something is going to happen. You also don't have the same methods to monitor and measure the system in a sequential system. So one of the techniques we use is parallel logging to observe what happened to the system post facto. So we don't, by probing the system, change the system's behavior.

I think that for the life sciences industry, parallel processing and harnessing the power of increasingly powerful, scientific specific GPU's is simply the next logical step in developing software for ultra high throughput instruments. As a life science software engineering firm, we have an obligation to our clients to stay on top of the very latest technology and assist our clients in deciding what makes the most sense for them, balancing both cost and technology.

Conclusion

We hope you found this discussion beneficial to your organization.

About Full Spectrum Software

Full Spectrum Software is an ISO 13485 certified, 15 year old consulting firm specializing in the development of embedded and applications software for the medical, life sciences and scientific industries. The company has delivered over 100 commercial products to market.

About the Author

Andrew Dallas, the firm's CTO, is widely considered one of the leading authorities on best practices in FDA and ISO 13485 controlled software projects Andrew serves on the Editorial Advisory Board of Medical Device and Diagnostic Industry magazine and he has published extensively in major trade and peer reviewed technical publications.

Contact Ken Carson, ClientServices@FullSpectrumSoftware.com
(508) 620-6400 • 225 Turnpike Road • Southborough, MA 01772 •
www.FullSpectrumSoftware.com