The Jacket Platform:
Productivity platform for GPU Computing

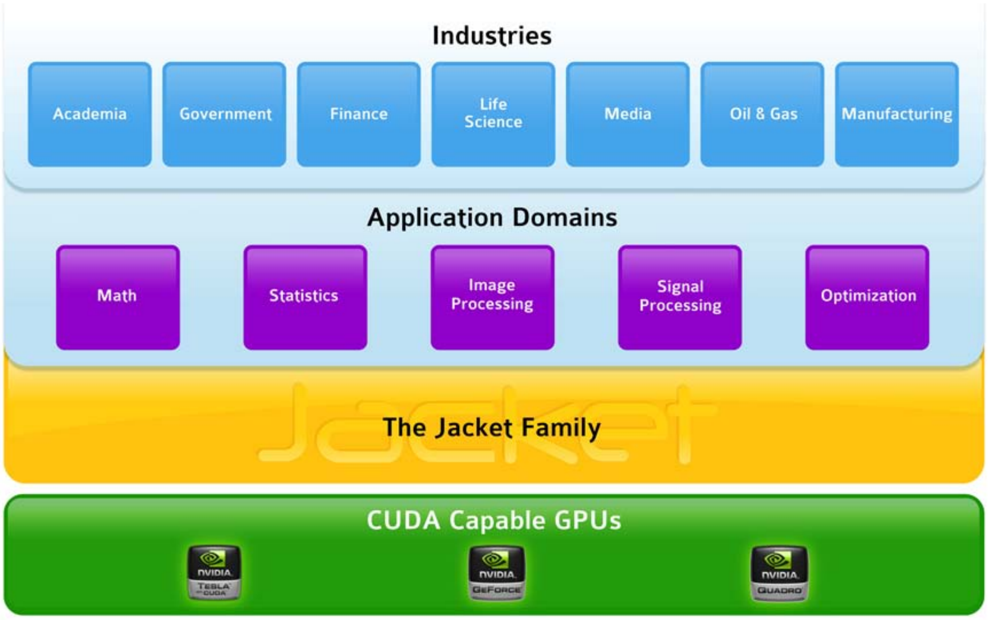
Summary

- Modeling and simulation is of growing importance to engineering, science and analytics. This trend has been the case for the last decade and will continue to be the case for decades to come. Technical computing is the primary vehicle for accomplishing results with increasingly complex data, time sensitivity and model sophistication of today's technical challenges.

- Various forms of technical computing systems and technology continue to be designed and delivered into the market but the tools used by Domain professionals to accomplish results remain the critical ingredient in new model and algorithm development;

- Developing algorithms for a changing technology landscape presents fundamental challenges to the productivity of the community that can leverage the new technology the most. This has been the case with the emergence of multi-core processors and large scale high performance computing clusters and is also the case with the emerging general purpose graphics processing unit (GPGPU) technologies. The algorithm development workflow introduces a level of complexity that requires a significantly new learning curve for the broad community and prevents interactive discovery that is most common in engineering, science and analytical problems;

- An emerging class of software – a platform for productivity with GPUs – is showing promise in enabling common languages used by domain professionals to operate interactively with GPUs;



- A platform for productivity with GPUs will help language vendors leverage GPUs without having to solve the significant challenges associated with low level programming and supporting multiple GPU architectures. Domain professionals will enjoy the performance gain of GPUs while minimizing the learning curve required to deliver improved results.

Growing Computational Challenges in Science and Engineering

In the mid-1990s, the desktop computer established itself as the primary science and engineering computing vehicle given the host of user friendly languages and tools that emerged. This was particularly the case during the early stages of new algorithm development, optimization challenges and various forms of simulation. The interactivity offered by the desktop computer and the tools available enabled an iterative and interactive process of research and discovery.

That said, over the last couple of decades there has been a growing class of problems that demand orders of magnitude more computational power than traditional desktop computers can deliver. Across a broad set of industry sectors, the problems scientists, engineers and analysts need to solve are growing in size and complexity. This growth is by far outpacing the hardware advancement simply delivered by Moore's Law. Many have turned to cluster computing and more recently, Cloud Computing.

Competitive pressures – whether in research, defense, or other industries – are demanding dramatic reductions in *time-to-solution* for critical problems. Models and algorithms of increasing complexity are needed to simulate whole products (rather than sub-systems), full scale scenarios, and multi-domain phenomena. Many of these problems require 10-100 times the computing power of a typical science/engineering desktop workstation, and thus GPU computing offers a computational alternative to traditional parallelism on larger scale clusters and clouds.

The Increasing Importance of GPU Computing

Graphics processing units (GPUs) were originally designed to perform the highly parallel computations required for graphics rendering. But over the last couple of years, they've proven to be powerful computing resources across more than just graphics applications. Designed with more resources devoted to data processing rather than flow control and data caching, GPUs can be leveraged to significantly accelerate portions of codes traditionally run on CPUs, ranging from science and engineering applications to financial modeling.

Although GPUs have largely been in the domain of gamers and moviemakers, recent GPU offerings are much closer to mainstream computing and are proving themselves to be capable workhorses across a broad set of application domains and computational challenges. The $4B technical workstation market, dominated by HP and Dell, are starting to offer GPUs in their product lines for general purpose computing and the $10B+ high performance computing market is starting to see large quantities of GPUs showing up in key computational resources such as the recent NSF Track 2 program lead by Oak Ridge National Laboratory and Georgia Tech.

Georgia Tech Team Secures NSF Track 2 Award to Develop Future Generation High Performance Computing System:

The initial system will pair hundreds of HP high-performance Intel processors with NVIDIA's new next-generation CUDA architecture, codenamed Fermi, designed specifically for high-performance computing. This project will be the first of the Track 2 awards to realize the vast potential of GPUs for HPC.

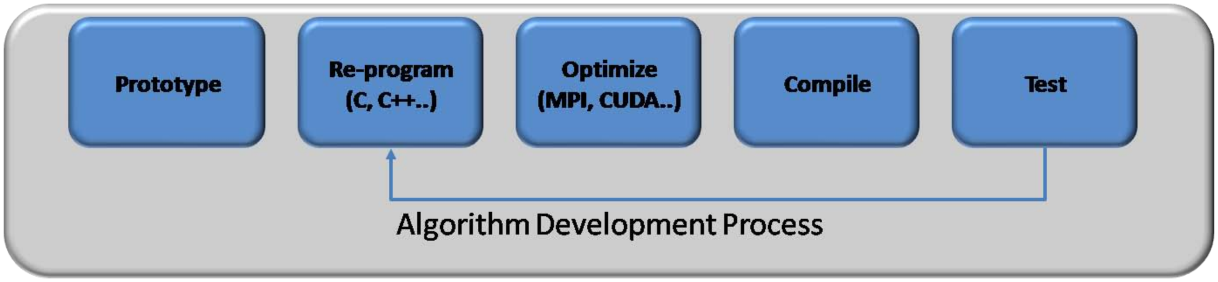
Although Nvidia has led the charge with general purpose GPU computing, Intel and AMD both are touting GPU-like general purpose computing functionality in current and future products. GPU computing is here to stay and will prove to be a major enhancement to productivity and performance both at the desktop and data center. Usability and programming capabilities will need to improve and those tools and platforms that are closest to what users are used to using today will turn out to be the well adopted technologies for this generation of computing technologies.

Last but probably not least, it should not go without saying that GPU technologies are far more energy efficient than traditional CPU architectures and less GPUs are potentially required to accomplish similar results than their CPU counterparts. If performance and productivity can be achieved, GPU computing will prove to be a very important advancement in technical computing for years to come.

Applications and Development for GPUs

Don't expect to log onto the "app store" to download your favorite app once you decide that GPU computing can help with solving technical challenges. Interactive desktop science & engineering tools are not broadly available for GPUs quite yet so lots of time and effort is spent moving problems (or subsets of problems) from CPU to GPU by a large set of developer communities. Although Nvidia's CUDA architecture and OpenCL are much needed software architectures to leverage GPU hardware, the **workflow to migrate algorithms and applications to GPUs takes time and sufficient knowledge of low level programming technologies and a few new extensions.**

Algorithm development for GPU computing is not unlike the development process required for parallel programming in multi-core or cluster computing – in other words, time, knowledge and energy is a pre-requisite. Scientists, engineers and analysts often start out prototyping algorithms with a familiar tool, such as MATLAB®, Python, or other higher level tools and languages. Once a prototype is complete there is typically a compilation step to get to lower level languages like C or C++ or computer scientists are engaged to migrate a prototype to a compiled language. Getting to parallel or GPUs requires even further time, knowledge and resource as MPI, OpenMP, CUDA, OpenCL and other programming extensions/architectures require tedious work to end with maximum performance on the target architecture. Upon completion of these migration steps, the domain professional is able to leverage the algorithm to solve the intended complex problem. This programming effort invariably takes away from model refinement and does take time.



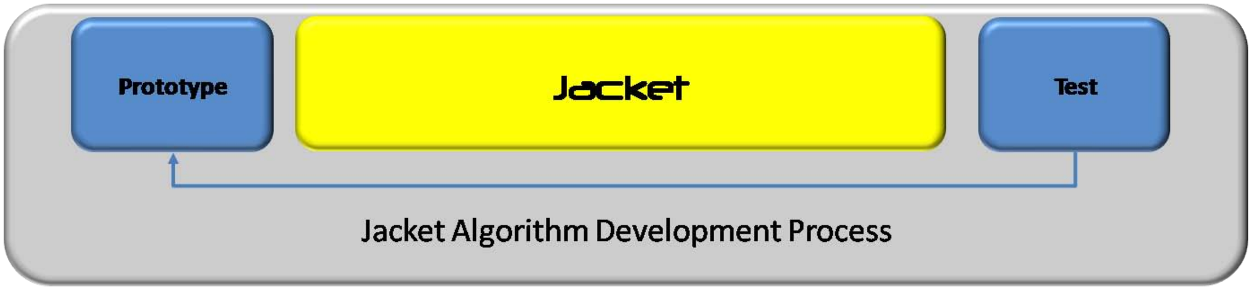
In many cases, the correct algorithm, approach, or key to the problem, may not be known up front, and may typically be discovered only by running the code on higher performance architectures, with the actual input data. Porting of models and algorithms to parallel and GPU architectures can take several months. Scientists and engineers are limited in the number of iterations to the algorithms and models they can make during the process. And this all happens before they ever get to actual usage of their models to solve the problems they have set out to solve. More than half of the "time to solution" is spent programming the models for use on parallel or GPU architectures, rather than developing and refining them up front, or using them in production to make decisions and discoveries.

The Need for a GPU Computing platform

When reducing *time-to-solution* is the goal, it is the engineers' and scientists' time that is typically the gating factor, not computing resources. GPU computing provides an additional resource to improve model development, performance and problem complexity given the magnitude of computational power available on the desktop with GPUs in addition to CPUs. By utilizing existing GPUs in desktop systems the domain professional is able to further refine models and algorithms to a different level before migrating to external parallel systems or GPU clusters. In addition to added computational horsepower in GPUs, the domain professional is able to remain in their favorite desktop tool of choice deeper into the development process – increasing productivity.

The emergence of a software platform for GPU computing – one that maintains productivity for domain professionals and maximizes GPU computational ability – is about to change the paradigm for algorithm development.

There is now an opportunity to bring the power of GPUs to today's and tomorrow's users, by eliminating the last hurdle: the need to manually reprogram models initially prototyped using desktop tools. With a GPU computing platform, the recoding is minimized, delayed or eliminated, so scientists and engineers can use the same desktop tools they know and love. They can prototype and scale in a tightly coupled process, in real time, with fine-grained control of both algorithms and data, transparently harnessing the GPUs computing resources.



The Opportunity for a GPU Computing Platform

A key benefit of GPUs is that many computations can take place simultaneously. These benefits come with a price – numerous technical challenges must be overcome in programming GPUs:

- Allocation of memory on the GPU device and moving data to/from that memory.
- A kernel—a function callable from the host and executed on the device by many threads in parallel.
- Calling the kernel from the host using an execution configuration—a means of specifying the number and grouping of parallel threads used to execute the kernel.
- Performing parallel computations using the built-in variables to subdivide the problem domain.

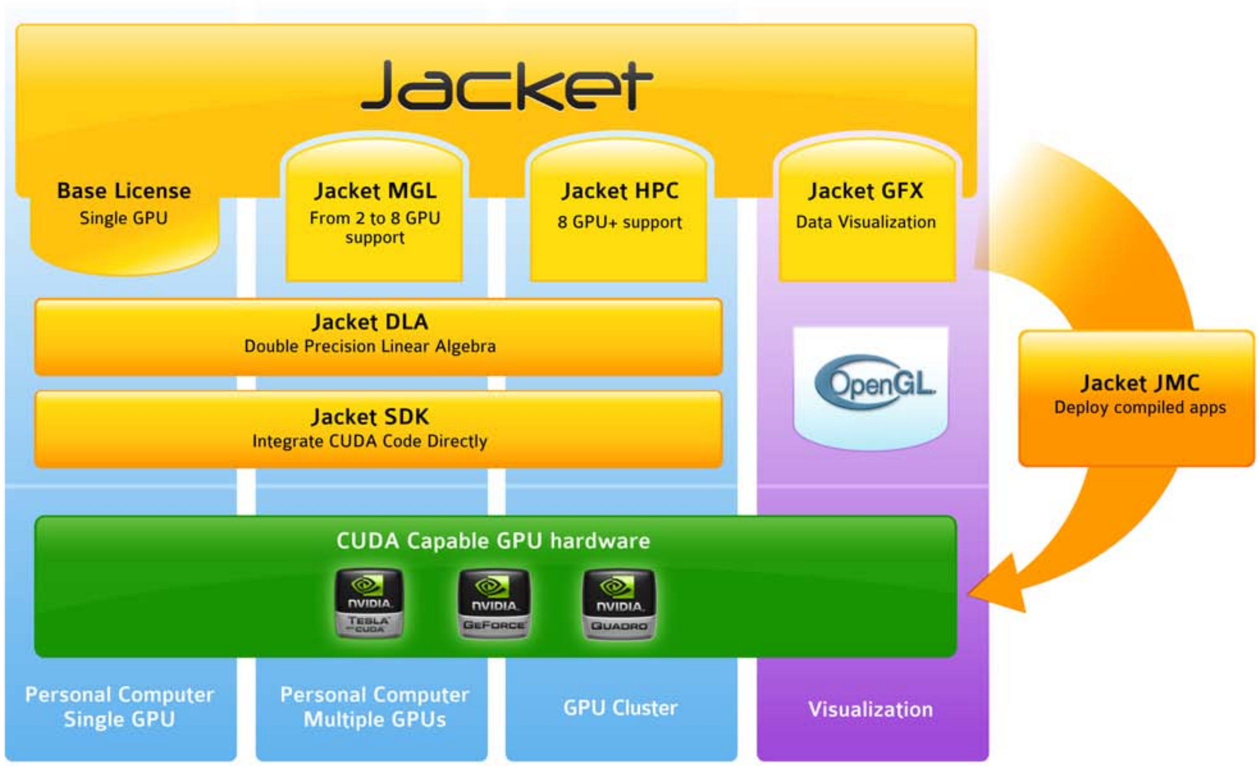
These programming challenges are a key bottleneck in both the *time-to-solution* for key problems, and the adoption of GPUs in science and engineering. Independent software vendors providing GPU based solutions are very limited today because:

1. The business model for GPU-specific application software has not been established;
2. ISVs typically cannot afford to spend the time and money that would be needed to optimize for specific architectures until a well established business model and market opportunity are defined; Nvidia has paved the way for a business model to develop very soon.

A productivity platform for GPUs can:

1. Reduce for the ISVs the required efforts to port their applications to GPUs;
2. Expand the number of software applications that run on GPUs;
3. Deliver the power of GPUs to popular desktop languages and tools usable by domain experts;

The Jacket software platform from AccelerEyes (Figure 5) does exactly that, links GPU performance and energy efficiency with “user friendly” numerical and scientific languages and tools. Jacket consists of a runtime and language processing system that automatically optimizes existing applications or new algorithms for GPU computing. Jacket currently supports the MATLAB language as a frontend to the platform. Jacket’s language processing system automatically translates MATLAB code to high performance primitives required for best utilization of GPUs. Working in concert with the translation system, Jacket’s runtime system optimizes memory transfers, compiles code on-the-fly for realtime tuned performance, and launches GPU kernels efficiently for maximal performance. All GPU-specific programming details are handled by Jacket, freeing the user to focus on science, engineering, and analytics.



A broad collection of libraries, solvers and kernels are available and will continue to be developed by a large community. These kernals can be integrated with Jacket via Jacket's software developer kit or SDK. With a few simple SDK functions, code from CUDA and future architectures can benefit from the Jacket platform.

Conclusion

The emergence of new software tools and cost-effective GPU-based computers are ushering in a new era for scientists and engineers working on large and complex problems. Familiar desktop tools can be used to develop new algorithms and applications and transparently tapping into the computing power of GPUs. By hiding programming difficulties and enabling a familiar environment, numeric computation can move to new levels on the various computing systems that house GPU technology. GPU computing is no longer an expensive time consuming niche in Technical Computing, GPUs are mainstream.

About AccelerEyes

AccelerEyes launched in 2007 to commercialize Jacket, the first software platform to deliver productivity in GPU computing. With advanced language processing and run time technology to transform CPU applications to high performance GPU codes, Jacket extends desktop workstation performance and fully leverage GPU clusters. Based in Atlanta, GA., the privately held company markets Jacket for a range of defense, intelligence, biomedical, financial, research and academic applications. Additional information is available at www.accelereyes.com.