

GPGPU to Many-Core Processing: Higher Performance for Mass Market Applications

Dinesh Manocha & Ming C. Lin

University of North Carolina at Chapel Hill

{dm,lin}@cs.unc.edu

Naga Govindaraju

Microsoft Corporation

nagag@microsoft.com

Introduction

A recent trend in improving the performance using parallelism is development of heterogeneous architectures that combine fine-grain parallelism and coarse-grain parallelism using tens or hundreds of processors. This is in the form of *many-core processors*, with the goal of achieving higher parallel code performance. This is in contrast with multi-core processors, which consist of best serial cores. As compared to current CPUs or multi-core processors, many-core processors can offer higher performance or power efficiency. The demand for many-core architectures is arising from consumer applications including computer gaming and multimedia. At the same time, they have tremendous potential for desktop computing, high-performance computing, robotics, and embedded applications.

One example of a commodity many-core processor is the current programmable graphics processing units (GPUs). For example, current top-of-the-line GPUs have tens of fragment processors and high memory bandwidth, i.e. 10x more than current CPUs. This processing power of GPUs has been successfully exploited for scientific, database, geometric and imaging applications (i.e. GPGPU or use of GPUs for general purpose applications). The significant increase in parallelism within a processor can also lead to other benefits including higher power-efficiency and better memory latency tolerance. In fact, the high number of fragment processors and high memory bandwidth of GPUs has been exploited for many scientific, database, imaging and geometric applications. In many cases, one order of magnitude performance was shown as compared to top of the line CPUs. For example, GPU TeraSort¹ used the GPU interface to drive the memory more efficiently and resulted in a 3X records/second/cpu improvement. Similarly, some of the fastest algorithms for many numerical computations including FFT,

¹ <http://research.microsoft.com/barc/SortBenchmark/>

dense matrix multiplications and linear solvers, and collision and proximity computations exploit the computational capabilities of GPUs².

The GPUs are primarily designed to achieve high parallel performance for rasterization and programmable shading. In order to exploit their computational capability and achieve portable performance, GPGPU designers typically reduce their problem into a series of rasterization problems. While this approach can work for some applications, it has some limitations for broad use:

1. The GPUs have poor programming support for general purpose applications. A programmer has to become an expert in graphics APIs like OpenGL and DirectX to exploit them well.
2. Current GPU runtime implementations do not support many other features including IEEE compliant 64-bit floating point support , scatter operations and the ability to execute multiple kernels simultaneously.

Despite of these limitations, the commodity GPUs have already demonstrated the potential of many-core computing for a broad set of applications.

Challenges of Many-Core Computing

There are many other fundamental challenges that arise in using these many-core processors. For example, many-core architectures provide us concurrency, i.e. the ability to execute the sub-tasks of a given task in parallel. However, in order to take advantage of the available hardware resources, we need to develop new parallel programming paradigms in software technologies and applications. The *new paradigm shift* brings us many challenges:

- **Software engineering and design:** Many of the current applications are not designed to take advantage of the newer architectures and many of these applications are designed using millions of lines of serial code. These applications may need to be redesigned — or even written from scratch. Tools are essential to help the transition of existing infrastructure onto many-core processors.
- **Handling heterogeneity:** Current scientific and multimedia libraries and applications are designed to take advantage of the homogeneous multi-core processors. New models and optimizations need to be designed to efficiently exploit the heterogeneous architectures.

² <http://gamma.cs.unc.edu/hardware>

- **Fault tolerance and reliability:** The increase in complexity of the hardware will also require better software fault tolerance and reliability techniques to handle failures.
- **Programming tools:** A key challenge is to design better programming models to take advantage of the many-core architectures. New tools with good debugging and profiling support are required to better understand the behavior of the applications.
- **Benchmarks:** As the hardware evolves, benchmarks that characterize the performance of applications need to be designed. These benchmarks can improve the hardware design and also validate the real application behavior.

These issues would drive a long-term research agenda with respect to many-core computing. Next, we describe a few applications that can directly benefit from many-core computing. These applications arise in computer gaming, 3D user interfaces, virtual worlds, robotics and scientific computation.

Applications

We give a short survey of some of the applications, which are immediate target of many-core computing.

Collision detection and response computation: The problem of proximity computation and response computations arises in computer gaming, animation, virtual worlds, robotics and CAD/CAM. In many cases, collision computations remain a major computational bottleneck for interactive simulation. Our research group has developed a number of algorithms and systems for collision detection and proximity queries over the last 15 years³. These algorithms and resulting software systems have been widely used in the research and industrial communities. While good solutions are available for rigid models, interactive collision detection among deformable models, including self-collisions remains a major challenge. As a result, there are relative fewer deformable models used in interactive applications, including computer gaming and surgical simulation. Our goal would be to exploit the multiple cores and high memory bandwidth of many-core architectures to generate bounding volume hierarchies and use them to perform intersection and culling tests in parallel. One challenge would be to develop scalable methods whose performance increases with the number of cores and memory bandwidth.

Modeling and control of soft articulated characters: Believable animation of articulated characters is essential to realistic virtual environments, computer games,

³ <http://gamma.cs.unc.edu/collision.html>

and other interactive applications. Often the movement of characters is driven by motion capture data. However, it is important to simulate the response of characters to collisions and secondary effects, such as skin wrinkles or surface deformations, thus enhancing the realism of the character animation. But, these effects are usually challenging to achieve in real time. It may be possible to use many-core architectures to efficiently simulate dynamic, colliding, detailed deformable characters with support for interactive manipulation of motion and deformation by an animator. These would include interactive algorithms that can enable physical simulation of deformable characters with high surface detail. The resulting system should be able to efficiently handle large areas of contact, automatically produce rich surface deformations, and effectively capture the skeletal response of the characters due to collisions.

Physically-based Simulation of Fluids and Solids: Fluid phenomena, either in liquid or gaseous form, play an important role in everyday life, including blowing wind, jet streams, chemical dispersion, granular flows, ocean waves, water current, etc. Although these phenomena may appear simple and ordinary, the mathematical modeling and computational simulation processes are actually very complex and challenging to perform in real time. The computational complexity of fluid behavior comes mainly from the complex interplay of various factors such as convection, diffusion, turbulence, and surface tension. Recent work has shown that a class of fluid solvers can potentially be mapped well to many-core architectures. Many-core architectures appear to be a good candidate to implement *Residual Distribution Schemes*, because of their natural affinity towards parallel architectures, their shock capturing character, and the ability to simulate mechanics of multiple phenomena (e.g. solid and fluids) on the same computational domain seamlessly.

Crowd and multi-agent simulation: Heterogeneous crowds consist of non-uniformly distributed groups of people with independent behaviors and different goals. Pedestrian dynamics of heterogeneous crowds deal with path planning and collision avoidance of each agent and it exhibits a rich variety of collective effects, such as lane formation, oscillation, chemotaxis and panic effects. Recently, we have proposed a novel multi-resolution representation called "Pedestrian Levels of Detail" for computing real-time pedestrian dynamics in highly populated urban terrains, for predicting civilian response to unexpected calamities and planning evacuation strategies⁴. Agents are grouped and crowds are split *on the fly* based on their dynamics states and behavior characteristics. Many-core processors can be used to adaptively compute global path planners and local dynamics models. Furthermore, they provide the capabilities to use empirically verified microscopic models for different agents in parallel.

⁴ <http://gamma.cs.unc.edu/Crowd>

Interactive ray tracing: Over the last decade, there has been renewed interest in ray tracing for interactive applications. This is due to exponential growth in processing power and development of ray-coherence techniques which can exploit the SIMD capabilities of current processors. However, current approaches (on a 8-core CPU) can mainly handle primary rays with simple reflections at interactive rates⁵. In order to generate global illumination effects, we need to trace a high number of secondary. Moreover, one main challenge for ray tracing is dynamic environments, that are widely used in computer gaming. The main goal is to develop ray tracing algorithms, which can easily scale with the number of cores in a many-core processor. A major challenge would be to develop appropriate hierarchies and representations that can fit into the local caches inside a many-core chip. Furthermore, as the number of secondary ray traversals increase, cache and memory coherence issues become significant.

Sound Synthesis and propagation: The believability of a computer simulated environment (e.g. game) often depends heavily on three main components: graphics, physics, and sound. Sound generation has not received as much attention, due to the extremely high computational cost for simulating realistic sounds. Physically based sound synthesis has clear advantages over recorded clips, as it can automatically capture the subtle shift of tone and timbre due to change in impact location, material property, object geometry, and other factors. Many-core processors can be used for physics-based sound generation techniques that will take the simulation results from physics engines to generate sounds automatically in real time for complex, dynamic environments undergoing various dynamic phenomena (such as fluid, explosion, etc.).

Recently, the PIs have developed a new technique called ray-frustum tracing for interactive sound rendering in complex and dynamic environments. Unlike prior geometric methods, that are either based on beam tracing or stochastic ray tracing, our approach can generate high fidelity sound and can handle complex geometric and dynamic scenes. Ray-frustum tracing involves adaptive sampling based on surface local geometry, and that makes it harder in terms of exploiting ray-coherence on SIMD architectures. It is also more challenging to parallelize such an adaptive technique on processors with hundreds of cores. Our goal would be to develop appropriate strategies to make our algorithm work well on many-core processors.

Scientific computations: One of the major goals is to exploit the computational power of many-core processors for scientific and numerical computations. Specifically, we will focus on sparse matrix computations, which are the building blocks of conjugate gradient methods and numerical solvers for differential equations. As compared to dense matrix solvers, sparse matrix solvers have a lower arithmetic intensity or relatively

⁵ <http://gamma.cs.unc.edu/RT>

less data reuse. This brings in new challenges in terms of how to utilize the multiple cores and high memory bandwidth of many-core processors for these computations. Ultimately, we will develop techniques for efficient sparse matrix solvers including linear system solvers, eigen-decomposition, singular value decomposition, and related computations.

Human-Computer Interfaces: We expect the human-computer interface in the near future to use more concepts that are based on the laws of physics to enhance the realism of interaction enabled by many-core architectures and many fundamental technologies described above. The proposed research will be central to the development of natural, intuitive physically-based simulation for enhanced multi-sensory human-computer interaction.

Potential Impact

It is expected that the game developers and simulation community would take advantage of many-core processors as well as transition years of high-performance computing research into commodity products to run on desktop and portable platforms.

The software generated as a result of many-core research can be useful to many applications such as computer games, surgical simulators, entertainment industry, robotics & automation, and be integrated with various libraries and systems (e.g. DirectX, Bungie) to run on different platforms (XBOX, PC, laptop, palm tops, cell phone or other portable or embedded devices). *Our long term vision is to incorporate physics-based computational approaches for specifying the behavior of virtual objects, as well as physics-based multi-sensory interaction for enhancing man-machine communication.* If successful, these can fundamentally change the way we interact with computer systems in general.