Advanced Cluster Systems

SET™: A Software Framework for Parallel Programming*

The Fastest Way to Parallel Programming for Multicore, Clusters, Supercomputers and the Cloud.

ABSTRACT

Programming for Multicore, Clusters, or Supercomputers ("MCS") boils down to parallel programming. To take advantage of MCS hardware today, you must have parallel programs that properly utilize that hardware. Parallel programming has been used in HPC (High Performance Computing) for decades, but multicore places new demands on writers of mainstream applications for parallel programming. There is no way to avoid it. Any HPC expert programmer will tell you that finding the parallel pattern to use in a program is not the hard part; it is dealing with the nitty-gritty details of implementing it. Such parallelization details are very complex, to a point that few programmers are capable enough to deal with them (Krste Asanovic et al. - "A View of the Parallel Computing Landscape" article, UC Berkeley ParLab, October 2009). For mainstream programmers, practicing sequential programming for decades, learning concurrency is complex enough, let alone the daunting details involved in implementing it. Computer scientists of academia do not make it easier for programmers either, experimenting with a plethora of "Parallel Languages" and "Parallel Compilers" solutions, none fulfilling expectations. These solutions are low-level in nature, adding additional layer of complexity to an already a complex problem.

The software industry needs to stop for a minute and think of an "Outside-the-Box" solution that will make parallel programming virtually as easy as sequential programming.

SET™ (Supercomputing Engine Technology™) is such an "Outside-the-Box" solution. SET is a software framework that significantly simplifies parallel programming. Implementing parallelization with SET is almost as easy as sequential programming. In this white paper we discuss the ways SET simplifies parallel programming for all.

PARALLEL WITHOUT THE PAIN

The SET™ software framework provides an MPI-based library, accesible by means of API, that significantly simplifies parallel programming and enables a much wider range of software writers to scale their codes efficiently on hundreds and thousands of cores. Many scientific algorithms are data parallel, and in most cases, these are amenable to parallel formulation. Conventional parallel programming, however, suffers numerous pitfalls. When writing MPI or multithreading code, too many programmers are stuck with code that either performs badly or simply does not perform at all.

Programmability is where SET makes a difference. SET API raises the abstraction level, making parallel programming easier, faster, and steers its users to good parallel code, all while executing efficiently on current hardware, from embedded systems and multi-

core computers through clusters and supercomputers. With SET, programmers can now continue developing the same sequential codes they are used to, identify the parallel pattern required to increase performance via multicore, and apply it painlessly to their sequential codes with the SET API. No complex, special-purpose parallel languages are needed. No complex parallel compilers are needed. SET supports C, C++, Fortran and Java. A standard gcc 4.0 (or later) or equivalent is all that is required for a compiler.

Why SET is MPI-based and not OpenMP-based

MPI (Message Passing Interface) is the dominant programming model for high-performance computing. OpenMP is not. MPI is ubiquitous; any parallel algorithm can be expressed in the MPI paradigm. It runs everywhere, from embedded systems to supercomputers. MPI can run on shared or distributed memory architectures, while OpenMP is suitable for shared memory only, which limits the size of problems it can be used on (Large shared memory machines are extremely expensive). MPI development is supported by the US government, and it is a common standard for hardware manufacturers.

SET supports both shared-memory and distributed-memory hardware. It presumes a rudimentary MPI (only a handful of MPI calls) is available, which is easy to support on virtually any parallel system if not already present. We bundle one (SlimMPI) with SET.

SET Serves High-Level Message Patterns on a Silicon Platter

To overcome programmers' fear of MPI, SET handles low-level message-passing internally. When writing parallel applications with MPI, programmers still have to develop a significant amount of software to manage some of the low-level tasks MPI requires. It is a time-consuming and complex effort that requires high level of expertise. Only supercomputing experts are able to provide highly-scalable robust codes, but these experts are few and far between. Limited supply of and growing demand for this expertise means utilizing them for mainstream programming is cost prohibitive. SET, as an MPI-based library, handles management of MPI low-level tasks under the hood like supercomputing experts, practically eliminating this chore for mainstream programmers. Thus, mainstream programmers can concentrate on what parts of SET should be applied to their code and quickly enable a full-fledged parallelized application, rather than being forced to wrestle with time-consuming, low-level parallelization tasks.

SET Encourages MVC (Model-View-Controller) Software Design Pattern

Modern applications are modular, and most of these follow a widely-used paradigm called "Model-View-Controller" (MVC). In many cases, even if the software writer of an application was not aware of the terms Model, View, or Controller, that writer can recognize, in hindsight, this paradigm applied in that application's code. The View is

Advanced Cluster Systems

the part of the code that shapes the overall execution of the code and expresses that execution to the user, as input and/or output. The Model is the code or modules or functions that actually manipulate the data at a low level. The Controller connects or bridges the Model and View. This is important because parallelization with SET is accomplished in the Controller realm. In virtually all practical cases, the Model and View portions of the code are unchanged, and the code to add to the Controller to work with SET is a small fraction of the overall coding project. Figure 1 below shows how SET integrates into the MVC software design pattern.

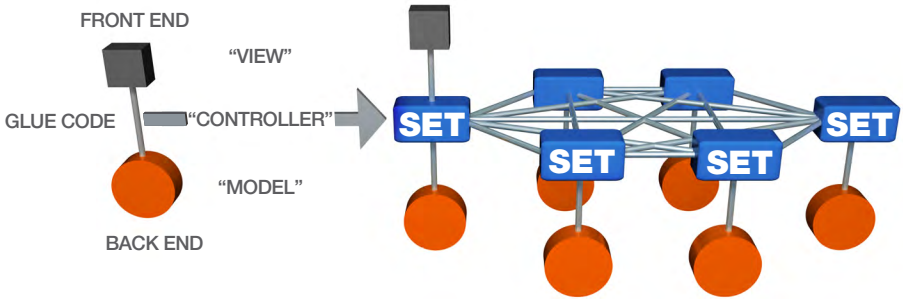

Figure 1. MVC (Left) and how SET is integrated into MVC (Right)

SET Avoids "Breaking" of Sequential Codes for MPI Parallelization

The standard workflow for developing high-performance software code is to design an efficient MVC sequential version that works well on a single CPU, then "break" it for parallel programming. However, as mentioned before, application development with MPI is difficult. Re-fitting sequential code using MPI is often a major undertaking, requiring extensive restructuring of the sequential code, and often taking more time than the development of the original sequential code.

Here is usually where programmers get in trouble, as it is quite easy to create codes that do not perform as expected. Moreover, ongoing maintenance of these codes can become a nightmare because all updates and upgrades of the parallel version easily break the parallel version.

A major advantage of SET (which overcomes a major disadvantage of MPI) is that there is no need to "break" the most important sequential code modules in order to parallelize them with MPI, so the parallel version of the code uses the same modules of the sequential code. This feature is not only important for new software designs, but for parallelizing legacy sequential codes as well. As for updates - every update or upgrade can be accomplished quickly and tested on the sequential version modules. The updated modules will work with SET just like the old ones, immediately deploying the updates or upgrades in the parallel version. Performance-wise, sequential code parallelized with

Advanced Cluster Systems

SET will approach or match performance achieved by native MPI parallelization.

SET Parallelization Approach Explained - Recognizing and Utilizing Modularity

Virtually all well-structured, non-trivial application code happen to follow the model-view-controller paradigm, even if its writer might not know those terms when writing the code. As described on the left side of Figure 2 below, every code has a main() or equivalent, as shown at the top. Every code has subroutines further down that actually do the hard work of the code, shown here as the deposit(), push(), and updateField() subroutines. Often these codes have a subroutine that call the low-level subroutines repeatedly, in a loop, such as the doLoop() here, and that loop is called by main().

Programmers recognize that the code itself expresses a hierarchy and connection be-

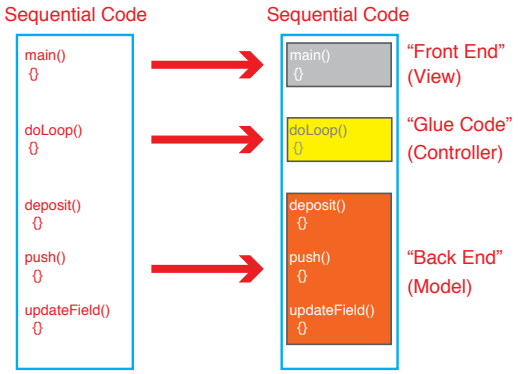

Figure 2. Identifying Modularity

tween its components, starting with main() and so on down the calling chain. As shown in Figure 1 on the right, programmers recognize that these low-level routines, like deposit(), push(), and updateField(), are the "Back End" or Model of the application, while the main() is the "Front End" or View of the application. The code interfacing in between the Front End and Back End is the glue code or Controller. Knowing this, programmers can then apply SET API and parallelize this application by placing corresponding pieces of the application above and below the SET infrastructure. The Front End code goes at the top, while the Back End code is encapsulated at the bottom. The Glue Code is split and changed with SET API, as needed, to interface to SET's parallelization layer. The size of the changes to the Glue Code is typically a very small fraction of the overall code and is usually below 1% for real-world applications.

As shown in Figure 3, all the changes in the code example are in the Glue Code. For the typical application, the Front End and Back End application codes are nearly unchanged. This fact translates into an enormous advantage in time savings for the ap-

Advanced Cluster Systems

plication writer. Not only is the parallelization process simplified and accelerated, but the resulting code is both much easier to maintain and much easier to upgrade because the Back End and Front End codes are isolated from the message-passing functionality and other details of the parallelization. The ongoing life of this application as a parallel application is, compared to other approaches, much more viable with SET.

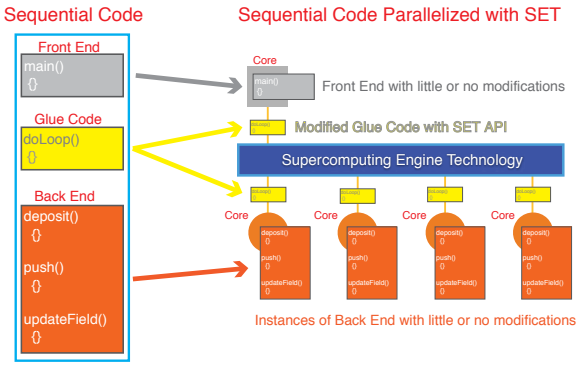

Figure 3. Utilizing Modularity for Quick Parallelization with SET

SET Runtime

SET runs on any Linux or Mac OS X hardware. It can run in embedded multicore systems, on a single multicore computer, a group of multicore computers networked together, clusters, and supercomputers.

A single multicore computer or a group of computers networked together in an office environment probably does not have clustering software and MPI installed. SET takes care of this by providing clustering management software and SlimMPI library (ACS's version of MPI as needed for SET) so that full functionality of SET can be developed and run in this environment. On a supercomputer, for example, SET uses the clustering and MPI resources available on that supercomputer.

SET Software Framework - The Perfect Companion to GPUs

As shown in Figure 4, SET is the perfect complementary technology to GPGPUs. The most successful approach to harnessing multiple GPGPUs is to designate one MPI process to manage each GPGPU. That creates a new problem: one needs to combine two very different and esoteric skills, MPI programming and GPGPU programming, rarely found in one person or one team and therefore expensive to corral into one project. SET can instead provide a far easier, yet still supercomputer-like, infrastructure to each node and across the cluster. SET compute cores on each node have access to a GPGPU pool

Advanced Cluster Systems

to expedite functions. Each CPU-based process of SET would leverage a GPGPU, and GPGPU functions would nest inside SET-supported computations. This isolation of GPGPUs from each other therefore fits perfectly into the distributed-memory paradigm of SET, while SET would provide the communications infrastructure between GPGPU-enhanced modules.

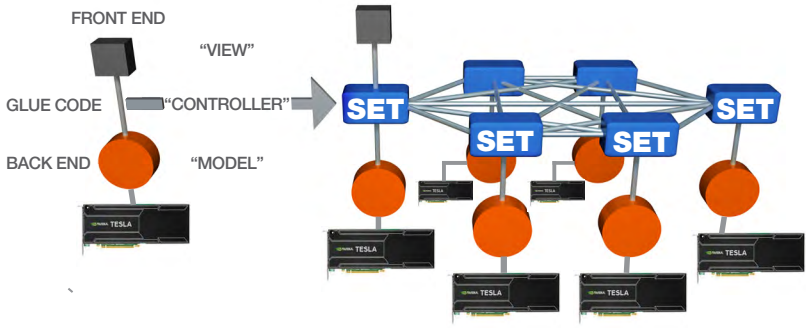

Figure 4. SET is a Perfect Companion to GPGPUs

SET Software Framework - Easy Access to Cloud Computing and HPCaaS

SET enables widespread and HPC-style exploitation of cloud computing's available parallelism. Cloud infrastructures, particularly HPCaaS, provide both greater processing power and network throughput, providing advantages to not only desktop computers, but especially to mobile devices like smartphones and tablets.

As shown in Figure 5, a simple variation of SET provides a way to bridge these two worlds. Essentially, the network connection to the SET and Application Front End is extended by thousands of miles. The SET Back Ends and Application Back Ends reside

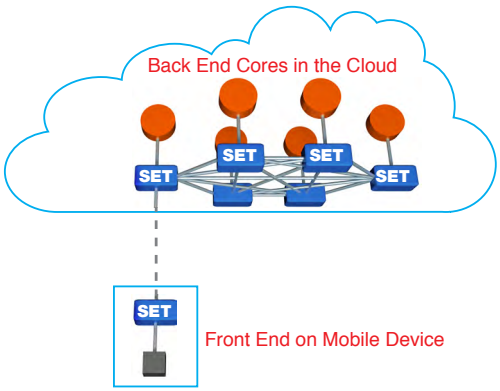

Figure 5. SET in the Cloud

Advanced Cluster Systems

in the cloud, just as for clusters and supercomputers, but Front End components, instead of residing on hardware in the same room as the Back Ends, reside on geographically-distant hardware of any kind. Naturally, the limitations in bandwidth, latency, and security of a network connection over that distance would apply, so some adjustments to the application may be needed to accommodate such issues, but the behavior and structure of SET remains.

This solution is useful because it addresses the understandably limited processing capabilities of any one tablet or smartphone, which are less than the processing capabilities of modern PCs, and are tiny by comparison to the computational capabilities of the cloud. Via SET, these mobile devices can fully direct and harness, above and beyond the usual cloud services, the parallelization power of cloud computing, creating a potent alliance advantageous for a wide range of applications.

Developing SET-based programs

SET SDK contains all software required to develop and run SET-based programs, including clustering software and SlimMPI. In addition, SET SDK contains programming examples described in the SET Reference Manual. Any standard compiler can be used to compile SET-based code.

SET Requirements

Operating System: Linux 64-bit or Mac OS X 10.4.11 and up

Compiler: gcc 4.0 or later or equivalent

Supported Languages: C, C++, Fortran (Compatibility Layer)

More Information

For more information, contact us at info@advclustersys.com or visit us at http://www.AdvancedClusterSystems.com.

* U.S. Patents 8082289, 8140612, 8402083, 8676877, Japan patent 4995902 and Patents Pending.