The Biggest Need: A New Model of Computation

Thomas Sterling
Louisiana State University
March 30, 2009

HPC is experiencing a phase change driven by technology advancements and constraints. For the first time in more than a decade conventional software practices for programming and managing system resources are no longer sufficient to address the challenges for achieving the high end scalability for a wide range of applications. Further, past strategies for system hardware architecture no longer utilize the emerging technologies to their fullest. Both are reflected by the emergence of heterogeneous multicore components and the systems that use them. Power is restricting clock rate, design complexity has been exhausted as a path of future performance gains, and within a decade parallelism on the order of billion-way concurrency will be required. Historically, major disparities between enabling technologies and the methods of their use have driven computing through an evolutionary event of punctuated equilibrium requiring simultaneous changes in architecture, programming models, and system software to achieve a new balance for efficiency and continued progress in performance gain. Sequential, vector, SIMD-array, systolic, dataflow, multithreaded, and most recently communicating sequential processes represent distinct phases in HPC, each a different model of computation. Currently, a new such model is required to redress the challenges imposed by the need for multicore.

A model of computation is not a programming model, architecture, operating system, or some form of virtual machine. Instead it is a strategy or discipline that specifies referents, their interrelationships, and the actions that can be performed on them. In so doing, a model of computation governs the semantics of state objects, function, parallel flow control, and distributed interactions. While it provides an image of an entire parallel computer, not just any single core, it leaves unbound policies of implementation technology, structure, and mechanisms. Yet, it influences the decisions for co-design of programming languages, compilers, runtime software, operating system, and even hardware architecture.

The goal of a new model of parallel computation for future Exascale computing is to serve as a discipline to govern future scalable system architecture, programming methods, and runtime techniques as semiconductor technology proceeds to nanoscale feature size. Such a new model has to innately hide latency both system wide and to main memory. It has to exploit parallelism in a diversity of forms and granularity. To this end it has to provide a framework for efficient fine grain synchronization and scheduling, enabling optimized runtime adaptive resource management and task scheduling for dynamic load balancing. Perhaps for the first time, the model of computation must extend farther to support full virtualization for fault tolerance and power management.

Then what would a new model of computation look like, even as it replaces the venerable message-passing model? While no definitive specification can be given without substantial

research in collaboration with the international community, there are a number of attributes that may prove imperative if it is to serve computing down to the nanoscale and up to the Exascale within the next decade. Perhaps most fundamental is to replace static processes assigned one on one to fixed processor cores with a new relationship between tasks and computing resources. One possibility is the basic work-queue model where each physical resource acts as a server to process a stream of task specifiers that work on relatively local program state. Instead of waiting for some remote access, the resource terminates the current task and begins a new one. Thus, the work-queue model decouples virtual tasks from physical processing resources to significantly increase resource utilization, at least in cases with sufficient parallelism. Complementing the work-queue model is the need to adopt a message-driven model, replacing conventional message passing. Message-driven computation allows work to be moved to the data when this is optimal, rather than always requiring that data be constantly moved to a fixed location of work. This is particularly well suited to dynamic graph problems such as adaptive mesh refinement and informatics. Such algorithms are heavily reliant on meta-data to describe the data structures. Extremes in parallelism will be required with future systems and meta-data used with message-driven computing may expose a diversity of parallelism forms and sizes, at least in comparison with conventional global barrier based techniques. Instead asynchronous methods need to be incorporated in the global flow control for adaptive management of resources. The elimination of global barriers allowing more flexible flow control such as data flow methods can be achieved with the powerful futures construct. To harness hundreds of millions of cores in to a single system requires a model capable of unifying all components and this requires a single system-wide name space. PGAS has been pursued and serves well for some problems. But when dynamic migration of first class objects is required; something more is needed so that data objects can move in physical space without changing their virtual names. Such an active global address space still excludes full cache consistency but enables lightweight access to remote data without overly constraining the distribution of that data. The parallel flow control state at the global level must be more flexible than simply the state of fixed allocation SPMD processes. A more powerful means of parallel control state based on the distributions of “continuations” is needed to decouple flow control from fixed physical resources allowing migration of control such as when traversing a dynamic directed graph. A new model will support a self-aware property that adjust system configuration and application rollback for fault tolerance and active power management.

The development of a new model of computation will free future application programming from the deadly embrace of MPI + Clusters/MPPs where no progress can be made because each is required to serve the other. It will permit a new co-design cycle of all levels of the system software and hardware delivering new programming models that will greatly simplify the programmer responsibility, dramatically improve efficiency, and exploit orders of magnitude more parallelism intrinsic to at least some algorithms.