PROCEEDINGS OF SCIENCE

Towards a New Execution Model for HPC Clouds

Thomas Sterling*
Center for Research in Extreme Scale Technologies,
Indiana University, USA
E-mail: tron@indiana.edu

Matthew Anderson
Center for Research in Extreme Scale Technologies,
Indiana University, USA
E-mail: andersmw@indiana.edu

The application of emergent Clouds to the domain of high performance computing is considered by examining the various operational modalities comprising the field of supercomputing and by analyzing their suitability to Clouds based on underlying factors of performance degradation. It is found that while throughput computing may be readily supported for such HPC workflows as parameter sweeps, capability computing and even weak scaled "cooperative" computing may not be well served using conventional practices. But the possible advance of revolutionary methods to manage asynchrony, exploit message-driven computing techniques and declarative synchronization semantic constructs such as found in the experimental ParalleX execution model may provide an alternative paradigm for bringing Clouds more closely aligned to Science, Technology, Engineering, and Mathematics (STEM) applications. Experimental results capturing an Adaptive Mesh Refinement (AMR) application in numerical relativity using the ParalleX-based HPX-3 runtime system demonstrates many of the required properties for HPC Clouds.

The International Symposium on Grids and Clouds (ISGC) 2012,
Febraury 26 - March 2, 2012
Academia Sinica, Taipei, Taiwan

*Speaker.

PoS(ISGC 2012)038

1. Introduction

Even as HPC is transiting the pan-Petaflops performance regime towards the sought goal by the international community of a sustained Exaflops before the end of this decade, a second revolution is in process with the emergence of distributed Cloud computing. Its potential impact across a broad spectrum of computational needs in commerce and society ensures it will be a major transformational factor in the coming decade. But what of its impact on one of the most rarefied of computational fields, that of high performance computing? Can the Cloud serve the ever increasingly costly domain of HPC and broaden accessibility to HPC capability as commodity clusters did almost two decades ago? As is implied by its designation, HPC is about performance in the quest for pushing the frontiers of science, technology, engineering, and mathematics for knowledge, industry, commerce, and national security throughout the international community.

Like so many disciplines, supercomputing is multi-faceted with a diversity of operational modalities, exhibited properties, and disparity of requirements. The question of the role of Clouds in HPC is really a question of which aspects of HPC can Clouds serve, for which ones can it not fit and why, and how might these limitations of HPC Clouds be mitigated for broader impact? In the following, an examination is presented of the distinct modalities comprising the field of HPC and an analysis is conducted to consider each of these in terms of the capabilities and operational attributes of Clouds. It is recognized for example that the broad array of throughput computing workloads are well suited to Clouds assuming that costs of usage can be diminished to level appropriate to academic and labs computing cultures. At the same time, capability computing is cited as an important class of computing that cannot be provided by the Cloud today along with certain forms of data intensive computing.

The limitations on Clouds for HPC using conventional practices may yield to innovations in execution models to exploit dynamic adaptive methods at runtime. Management of asynchrony, message-driven computing, and constraint-based synchronization are proposed as possible means for alleviating some of the underlying problems with current Cloud environments. Thus, a new execution model for HPC Clouds may greatly extend their utility for Science, Technology, Engineering, and Mathematics (STEM) applications. One experimental execution model, ParalleX [1 – 3], is briefly described and experimental results are discussed that suggest its potential as a basis for a revolutionary new Cloud methodology.

The next section discusses the role that paradigm shifts have had throughout the history of supercomputing to continue the greater than two order of magnitude performance gain each decade for a total gain in one human lifetime of greater than a factor of a trillion suggesting a possible need for another phase change with the advent of multi-core and GPU component elements. Section 3 provides a qualitative analysis of the different HPC modalities and their underlying support requirements. From this, Section 4 establishes the roles that the Cloud can effectively contribute to HPC using conventional practices while Section 5 looks at those requirements for HPC that currently do not fit the Cloud and why. Section 6 then expands the potential of Clouds by considering a new execution model, ParalleX, and its possible adoption to HPC Clouds to address some of the limitations identified. Section 7 briefly presents empirical evidence that supports the merits of this approach. Finally, Section 8 finishes with a set of conclusions that propose the possible future of and HPC Cloud execution model.

2. HPC in Phase Change, again

The field of high performance computing has experienced a rate of growth unmatched in the history of human technology. In one human lifetime digital electronic stored program computing has exhibited change of a factor of a trillion and this is a conservative estimate. The lead author was born at a time when extant machines (there were only a couple) performed at approximately 1 KIPS and these were integer operations. This compared to the current Petaflops era demonstrates this amazing growth in performance. Further, it was not that the technology was wrong and then suddenly the right answer was discovered. Rather, the delivered performance improved by approximately a factor of 200X every decade over more than six decades. But this apparently smooth history of performance gain was not simply a consequence of continuous improvement but rather punctuated by dramatic change at critical moments driven by the introduction of new enabling technology opportunities. These abrupt transitions were nothing less than paradigm shifts reflected by the derivation and adoption of new execution models.

One possible breakdown of this history suggests five phases of HPC. The first phase, the von Neumann model, driven by vacuum tube logic and early memory technologies eventually including magnetic cores. This employed the classic fetch-compute-write cycle. The second phase driven by the emergence of transistor based enabling technologies exploits the higher clock rates possible and higher packaging density to employing pipelining execution units still with sequential instruction issue but with overlapping of operation execution exploiting one form of fine grain parallelism.

The third phase in the 1970s driven by the emergence of small and medium scale integration (SSI & MSI) demanded a new paradigm, the vector execution model, again using pipeline structures but this time for register access, floating point arithmetic units, and data access. The Cray 1 is the archetype of this class of vector computers although earlier systems employing some of these techniques had been tried. But by the 1980's the fourth phase of HPC was driven by large scale integration (LSI) that provided much more capability in a small space demanding yet another new strategy, this the SIMD-Array processing model in which the same operation would be performed by many simple processing units but on separate DRAM memory banks.

Finally, by the 1990s very large integration (VLSI) imposed a new opportunity for low cost system structures including for the first time the leveraging the mass market and resulting economy of scale. The fifth phase of HPC based on the communicating sequential process (CSP) was manifest in two forms: MPPs and commodity clusters. This last HPC phase has dominated supercomputing for the last two decades bringing us to the edge of the Petaflops era.

Two factors suggest that we are on the verge of a new HPC phase change. The first continues to be VLSI density increase that promises approximately 10 nanometer feature size by the end of this decade. The second is the availability of massive wide-area network bandwidth spanning continents. Together they are defining a new generation of computing and quite possibly a new phase in HPC.

The critical changes to computing technology have hit limitations that have resulted in new classes of structures and their use. Over the last two decades, microprocessor performance had continued to improve with Moore's Law through the exponential growth of clock rates and innovations in processor core architecture to improve instruction level parallelism (ILP). Due to power limitations, clock rates have flat-lined. Architecture opportunities with respect to ILP have been

exhausted. The result of these two trends is that microprocessor core performance is no longer increasing with Moore's Law. In response to this dramatic change in the path to system performance improvement, multicore sockets and GPU accelerators (both of which combine a plethora of cores although in very different organizations) are now creating a new dominant form of parallelism leading us across the Petaflops performance regime. HPC systems of the future will be very different from systems of the past. These changes are compelling HPC in to a new phase of how they are structured, programmed, and managed.

The second seminal change is the ubiquity of wide-area networks and availability of rapidly increasing long distance bandwidth. This is combined with the unification of cluster-based systems as the principal platform for the processing of both commercial and much of HPC scalable applications. Together, they enable the evolution of Cloud computing and its potential role in HPC.

3. HPC Modalities

To explore the potential role of Cloud computing to the challenges of high performance computing, the nature of HPC needs to be understood in terms of its distinct modalities. These define the different ways in which HPC may behave and the underlying requirements. Here four such modalities are discussed. Based on these modalities, the following two sections will discuss the conditions where Clouds can support HPC and where it would be inappropriate.

3.1 Capacity Computing

Capacity computing is also referred to as "throughput computing" and combines job stream parallelism with weak scaling. A key property of this modality is that there is no interrelationship among the independent and concurrent tasks (in this case jobs) either in the form of synchronization between them or in their exchange of data. Performance is measured by total floating point performance of the set of combined jobs. Weak scaling delivers greater performance on a larger system or ensemble of systems by increasing the job concurrency, i.e., the number of jobs that are available to be processed simultaneously. Because there are no operational dependencies, there is little overhead and no ordering constraints. This provides what is often referred to as "embarrassingly parallel" workflow. In addition, the coarse granularity exhibited at the job level amortizes and minimizes the overhead costs of the job scheduling middleware achieving high efficiency. Moab [4] and Condor [5] are just two examples of a long history of such middleware to support this kind of parallelism and HPC modality.

3.2 Capability Computing

At the other extreme of HPC is capability computing which employs strong scaling of a single fixed size job. All parallelism to be exploited is exposed and exploited from within the single job. Further, all scalability comes from this parallelism. As the scale (physical size) of the applied system is increased, the fixed sized job in this modality exploits the increased resources to proportionally reduce its execution time, that is its time to completion. This is the most difficult form of HPC as the parallelism exploited for this speed up is finer grained than that of the capacity computing approach and latencies, overheads, and waiting due to contention as well as the parallelism may all impose bounds on the degree of granularity that can be effectively employed. Performance

can be measured also in terms of floating point operations per second but scaling is measured in terms of response time and its reduction with respect to resources.

3.3 "Cooperative" Computing

An intermediate modality between capacity and capability computing is also used, although curiously not usually acknowledged by the conventional lexicon of HPC. For this purpose, the term "Cooperative Computing" is introduced for expository purposes to represent this third modality. Cooperative computing combines aspects of capacity computing with those of capability computing. Like capability computing, cooperative computing performs a single application or job with available parallelism extracted from within the job. Like capacity computing, cooperative computing uses weak scaling such that as the hardware resources applied to it are increased, so is increased the problem size usually in terms of the data set size being processed by the application. Usually the available parallelism of a problem is increased with increased data set size. The effect is to retain the same task granularity and therefore the same balance of computation to overhead and latency costs. But the response time of the problem does not decrease with increased system size. In fact, it usually increases somewhat due to secondary effects. Cooperative computing is one of the dominant ways in which HPC is employed, especially with such application programming interfaces as MPI [6].

3.4 Data Intensive Computing

Seymour Cray is purported to have defined a supercomputer as a machine that converts a compute bound problem into an I/O bound problem. Nothing could be more true for the last modality of HPC; that of data intensive computing. Data intensive computing changes the balance of resource requirements and in many cases brings in another resource, that of secondary or mass storage. While numeric processing in this modality is still important to the processing of the data, the combination of access from mass storage and the greatly increased proportion of data movement throughout the system alters, perhaps dramatically, the means of optimization. One critical aspect of this fourth class of computing is the impact of the many milliseconds-long latencies that are incurred by direct disk access. Not only the time but also the uncertainty of time of access greatly complicates specification of codes that will produce optimal performance. Finally, the shear scale of the data that must be moved between storage and the mainframe can be prohibitive except in a tightly integrated system with the highest possible internal bandwidths.

4. Where HPC Fits on Clouds

Among the greatest opportunities for HPC employing Cloud infrastructure is in the domain of capacity computing. Here job stream parallelism can be applied to an anonymous set of processing resources without concern for time or order of job completion as long as the scheduling of jobs is dynamic with respect to availability of resources. Much of science is performed through this basic modality. Although focus is often placed on very large, highly parallel applications, the majority of science is accomplished on smaller subsets of such systems or on single nodes alone.

One important version of this is for parameter sweeps. In this class of problems, all of the jobs are the same. But work in each instance is performed on a distinct set of input parameters.

Such parameter sweeps expose ranges of behavior either of simulated science phenomenology or for engineering design optimization studies. Although the results of the set of combined jobs contributes to a single outcome, the parameter sweep (as possibly presented in a graph form), there is no interrelationship during their respective operations either in sharing of partial results or in coordination. Only during the collection of the results after their separate computations is there any tangential association and this does not impose any overheads or latencies during the actual simulation runs.

The opportunity of job stream parallelism on Clouds is not restricted to ensembles of sequential tasks. If the separate jobs are scheduled by the Cloud on anonymous SMP (e.g., multi-core nodes supporting shared memory) nodes, the advantages of job stream parallelism are maintained while locally exploiting medium grained parallelism as well. Typically such jobs are cast using the OpenMP API [7] that allocates hardware threads to code loops and segments in addition to the master or control thread. Flexibility is allowed as to scale of SMP assuming sufficient memory capacity again due to the asynchrony of the separate jobs comprising the total ensemble.

Under some circumstances, individual clusters made available by a Cloud can be assigned to MPI jobs. In the simplest practice, the user specifies the number of cores required, the minimum memory capacity per core, and some measure of the network bandwidth assumed. It may be that assertion of latency bounds is also specified. In this approach only clusters that satisfy these minimums can be designated as appropriate and allocated to MPI jobs in the Cloud. But even in this limiting case, performance levels are not guaranteed. Differences in processor core speed, socket design and I/O pin contention, and sharing of the clusters with other jobs resulting in possible network contention can all change the effective throughput of the MPI application. The larger the system requirement (e.g., number of cores, nodes, etc.) the fewer are the possible sites of execution within the Cloud. Relaxing the constraints required, or at least some of them permit the tradeoff of execution time to time of initiation. Allowing the use of smaller systems sooner may provide results earlier even if the computation itself takes longer because of availability of lower scale systems or system partitions.

One last class of workloads is suitable, even required, to be processed by the Cloud. In some science domains employing large sensors such as telescopes (e.g., Keck, Arecibo) or synchrotrons (e.g., LHC) the data sets accumulated at a continuing rate are enormous and their movement across the Internet can be infeasible in totality. In lieu of moving these prohibitively large data sets to where the work is to be performed, the Cloud permits the work to be moved to the data set. Computing systems local to the large depositories of data avoid the need for long distance data transfer and instead can perform required jobs on the data at their location. When data is distributed among multiple sites, the job to be deployed may not even know where the data that is required is. Or the data may reside at multiple sites. The Cloud can identify the necessary resources by the identified data and manage the remote processing automatically, returning only the final results to the requesting site.

5. Where HPC Doesn't Fit on Clouds

To fully delineate the limitations of Clouds in supporting the breadth of HPC needs, several factors of performance degradation are discussed by which to consider the implications of Clouds.

These are Starvation, Latency, Overhead, and Waiting for contention or SLOW. Starvation is the insufficiency of work to keep all physical resources busy either due to either an inadequate totality of work for the entire machine or imbalance of distribution of work such that while some computing elements may have too much, others will have too little at the same time. Latency is the distance, often measured in cycles (the time normalized to the inverse of the processor core clock rate it takes to transit the distance), required to perform a remote access or to request a remote service. Overhead is the additional work required to manage the parallel physical resources and the abstract tasks that would not be required in a purely sequential execution context. Overhead is a source of inefficiency requiring time and energy that does not directly contribute to the actual work to be performed. But a second effect is that the amount of overhead imposes a lower bound on task granularity that can be effectively performed. This in turn negatively impacts the amount of parallelism that can be exploited and therefore the overall scalability of an application of a given size. Lastly, the delays due to contention for shared resources, either physical or logical, can impose unpredictable bottlenecks and skew in the order and lengths of execution. Together, these factors can result in severe degradation of efficiency and scalability leading to dramatic reduction in delivered performance.

Historically to address these factors, HPC applications developers have explicitly managed many aspects of program operation through direct and explicit tuning of codes to the underlying hardware. The codes were performance tuned for physical machines. But this required knowledge of machine structure and key properties for optimization. Within these constraints, resources constituting as many as hundreds of thousands of cores are exploited at one time for a single application with million-way parallelism required at levels of both coarse grained and fine grain parallelism. To achieve efficiencies such as high cache hit rates and large network message packets, strides through memory have to be controlled and fine tuned to cache sizes throughout their hierarchy. When achieved, this can minimize effective memory access latency. One major concern as described above is overheads that determine efficiency and granularity of parallelism imposing constraints on scalability. HPC application programmers employ methods that minimize such overheads. Network latency effects are generally mitigated by the use of coarse-grained parallelism but increasing communication latencies can offset this. The granularity of the parallelism is ultimately determined by the memory size of the node. So there is a maximum level of latency that can be tolerated. Another factor is the minimization of skew delays. It is essential that all separate concurrent subtasks of an application execute at about the same time, starting and completing within a narrow window of the other parallel tasks. When this does not happen, hardware resources are blocked and efficiency diminishes, possibly drastically. All of these properties of parallel codes and programming have enabled the success of MPP and cluster computing and challenge the broad use of Clouds in some HPC regimes.

There is an expectation among some that Clouds can replace HPC systems by aggregating multiple or even many separate and distinct systems together to achieve a high aggregate capability. Due to the challenges just described, this strategy is infeasible for other than throughput computing. Therefore, there are modalities of HPC in which Clouds do not fit. Among these is the execution of a single MPI job spanning multiple systems such as MPPs or commodity clusters. The reason for this reflects the factors of performance degradation, in particular: dramatic increases in latency and overhead. It is likely that the granularity of processes, although coarse, will be insuf-

ficient to mitigate the effects of the latencies between systems in the Cloud that would easily be 10 to 100 milliseconds, a thousand times or more that exhibited by commodity cluster integration networks. Because entire system boundaries are being crossed, the overheads are expected to be substantially greater than those internal to MPP systems. Equally damaging are the resulting skews from these effects and the additional uncertainties of different system schedulers of the separate systems working out of synchronization with each other. The last to return a result such as for a barrier or other global reduction operator will determine the rate of progress. Thus, the pace of aggregate execution is predicated on the slowest system. For the Cooperative Computing modality, Clouds will not replace large HPC systems.

The situation is worse for Capability applications across multiple systems; again, the objective of replacing HPC scalable systems with the essentially infinite resources of the Cloud. Even medium coupled commodity clusters struggle to serve the Capability class applications due to issues of overhead and latency. Overheads for clusters can be an order magnitude less than the general Cloud and latency as much as three orders of magnitude less; and the gap is possibly greater than that in both cases. Systems such as the Japanese Kei and the Cray XT family of supercomputers are designed to minimize these properties in support of Capability-class operation. Today's capability applications are usually fine tuned to the target machine platform and Clouds will generally provide only rough approximations of the resources required with asynchrony between them precluding effective optimizations. Overheads impose a lower bound on granularity and latencies absorb concurrency for hiding that might otherwise be used for response time reduction. Both therefore degrade scalability on the Cloud. Unpredictable variability of parameters destroys performance tuning. Of course it is possible that one might access a specific system or system class via the Cloud for Capability applications. But that rather defeats the purpose of the Cloud; exploiting economy of scale and amortizing costs of resources through anonymous sharing. Also, there are relatively few such machines available and these have complex time allocation policies. Indeed, time on many of these systems is often competed for through proposals. Again, the Cloud does not fit easily into this usage model.

Yet another modality, data intensive computing for science applications beyond a certain scale (data set size) becomes infeasible for Clouds. The problem is that all of the data that has to be processed must be moved over the Internet to the remote Cloud computing resources. For some applications, the data set is simply too large to move long distances due to the bandwidth capacity of the wide area networks employed. It is also quite possible that the majority of Cloud systems, most deployed for general needs, will not have sufficient memory capacity or secondary storage to stage the mammoth data sets being considered for these applications which could easily exceed many Petabytes or even Exabytes. For this class of problem, moving the data is not viable. This is one of those cases where the solution is to move the work to the data rather than the data to the work. Obviously this means populating the data archive with sufficient computing resources as well, thus eliminating the need or value for the Cloud in this form of HPC. One other special case involving data intensive computing emerges when the data is highly sensitive, either because it is proprietary and cannot risk falling in to the hands of potential competitors or because it is classified and related to national security and therefore has to be carefully protected. Protection of data is a major area of concern and research within the Cloud with expected advances making them increasingly trust worthy. However, it is unlikely that some data will ever be trusted to the Cloud

given the extremes to which security is applied today.

From this we see that there are important domains within the HPC arena that for various reasons are ill suited to the properties of the Cloud in spite of potential benefits.

6. Concepts for New Clouds Execution Model

Can the limitations of Clouds in serving HPC be eliminated or at least mitigated? The problems are clear at least to a great extent as discussed in detail above. Then, are there possible solutions? The answer lies in the way computation on the Cloud is structured and organized but this may require a radical, even revolutionary paradigm shift and change to a new execution model. As discussed in Section 2, such phase changes in HPC are not unprecedented with at least five identified over the history of supercomputing. The requirement is to address the fundamental problems of efficiency and scalability within the context of the clouds and this in turn requires new approaches to the challenges of starvation, latency, overhead, and waiting (for contention resolution). Fortunately, these same challenges, although at markedly lower levels, on MPPs and commodity clusters (e.g., Beowulf-class systems) have resulted in substantial research and experimental development relevant to the problem of HPC on Clouds. One strategy is embodied in the principles of the experimental ParalleX execution model and is proof-of-concept runtime system, HPX-3, developed at Louisiana State University. ParalleX is intended to exploit runtime information to achieve greater efficiency of computing resources (e.g., cores, nodes) and to extract dramatic increases in exploitable parallelism to significantly increase scalability for such problem classes as dynamic graph-based algorithms. Some of the key integral semantic components of ParalleX are:

- Establishes a global address space (AGAS) that is active in the sense that a virtually addressed object may migrate through physical space (e.g., across nodes) without having to change its address,
- Defines contexts of data and tasks, ParalleX Processes, that provide protected abstract domains across multiple system nodes using capabilities based addressing to present a hierarchical name space (on top of the AGAS). Such processes are first class objects and are both dynamic is distribution and ephemeral in time (they are born, compute, move, and die). The can share physical resources as necessary.
- Employs "computation complexes" (threads are a subset of these) that serve as local combinations of interrelated actions performed on the same execution unit and shared memory with bounded action times and guaranteed local compound atomic action sequences. Complexes are first class objects, ephemeral, and can be switched in and out of active execution units as well as suspended due to lack of immediate work (depleted).
- Communicates by Parcels, an advanced form of active messages, that moves both work and potentially control state (continuations) to the data when appropriate as well as conventional asynchronous gathers of data to the work.
- Supports Local Control Objects (LCO) for powerful but lightweight synchronization to eliminate global barriers and manage asynchronous compound actions (e.g., dataflow, futures) to reduce overhead and expose additional parallelism.

7. An Example of Advanced Execution Model Components

An example of the opportunity afforded by ParalleX-based runtime and programming has been conducted for a particularly difficult application, one of simulating colliding neutron stars to understand the underlying mechanisms of the physical phenomenology associated with gamma-ray bursts (GRB) and gravitational waves. An important class of dynamic graph algorithms, adaptive mesh refinement (AMR), is employed to greatly reduce the computational demands imposed by fixed uni-grids at comparable accuracy but imposing complex and time-varying data structure management to achieve this. Historically using conventional static programming methods (i.e., MPI), scalability has suffered with long execution times limited to a few hundreds or thousands of cores.

Experiments run with this code and previously documented show three interesting results. In Figure 1, scalability of the MPI version of the HAD AMR software is compared to that of the ParalleX version using the HPX-3 software library. It shows that as the number of levels of refinement are increased, also increasing available parallelism that the HPX-3 version of the code increases in scaling while the MPI version actually diminishes in scaling due to the added overheads of managing it. The comparison of the actual wall-clock times of the two systems running on the same hardware platform and processing the same data sets are shown in Figure 2. This is run on a single SMP with increasing number of cores. The results are stunning. In the medium range of system scale, the HPX version of the AMR algorithm delivers performance advantage by a factor of 2X to greater than 3.5X with respect to the execution times of the MPI version. Finally, the importance of managing asynchrony is exposed in Figure 3 where multiple phases of execution, usually separate in the MPI version, are overlapped in the HPX version hiding communication latency and exposing a new level of parallelism even as it self-adapts to runtime asynchrony. These same opportunities, if employed on Cloud-based systems will ameliorate the challenges this new class of distributed system imposes and may further expand their role in HPC in areas and modalities current outside their scope of operation.

8. Conclusions

This paper has briefly described the modalities of high performance computing and discussed which of these are suitable for Cloud support while presenting those that are not. Basic issues of efficiency and scalability have been explained that are responsible for the limitations of Clouds for HPC. But new approaches based on revolutionary execution models including ParalleX were proposed as possible basis for a new Cloud execution model to expand the application of Clouds to HPC. Three major elements of the new Cloud execution model emerge as critical. The first is to provide the means of managing asynchrony at runtime. This is essential to address the challenges associated with Clouds including:

- Non-uniform response time for separate executing blocks
- Unpredictable access to shared Cloud resources
- Varying capabilities of Cloud resources

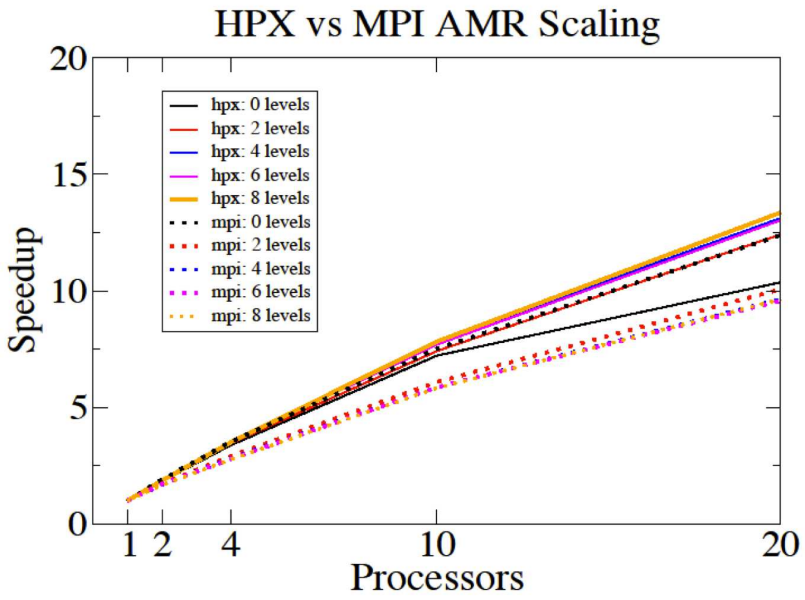

Figure 1: Scalability of the MPI version of the HAD AMR software compared to that of the ParalleX version using the HPX-3 software library. In the ParalleX version, as the number of levels of refinement is increased, the scalability improves. In the MPI version, the scalability diminishes as the number of levels of refinement is increased.

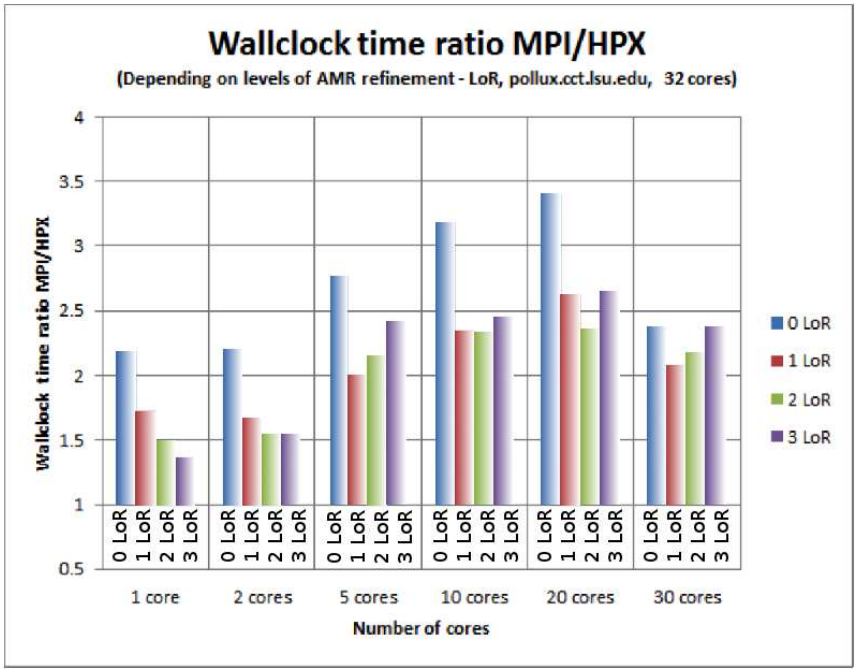

Figure 2: Comparison of actual wall-clock times between the MPI AMR software and the HPX AMR software running on the same hardware platform and processing the same data sets.

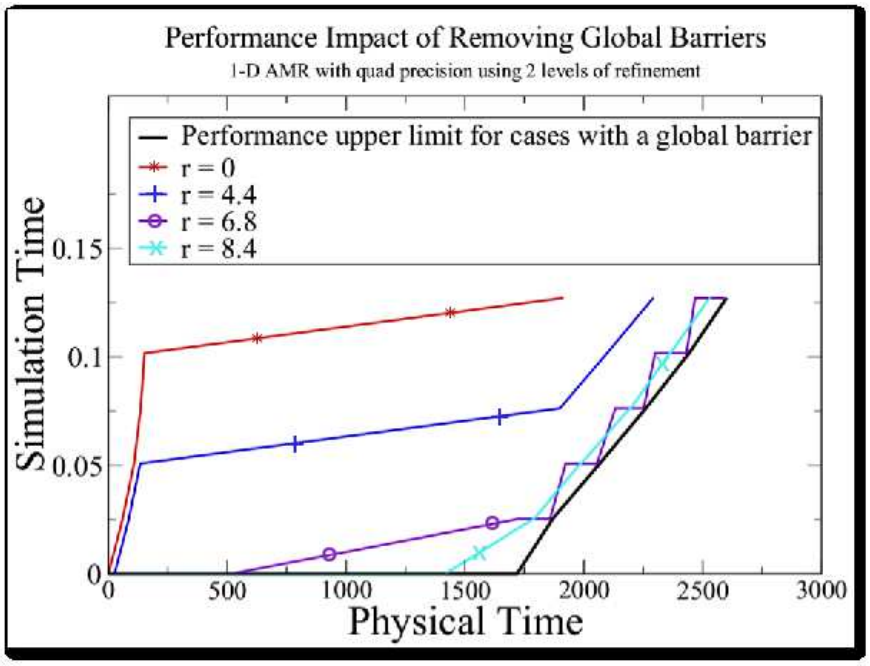

Figure 3: Measurements showing how multiple phases of execution are overlapped for individual domain grid points in the HPX AMR software.

- Wildly variant communication times due to varying latencies and contention

The second is the use of message-driven computation to move work to the data rather than always the data to the work and enable runtime adaptive latency hiding techniques. This is particularly important for certain classes of data intensive computing in science and societal applications where the data sets are too large to move over the Internet. The third element of a new Cloud execution model is event-driven constraint-based synchronization to provide dynamically adaptive methods of self-scheduling of concurrent tasks based on knowledge of runtime conditions. Such powerful synchronization semantics constructs establish the conditions for which new tasks may be instantiated rather than fixing the timing of the work through imperative commands.

But even with these advanced methodologies empowered by a revolutionary execution model like ParalleX, it is still anticipated that Clouds will exhibit limitations in their support for HPC constrained by the hard physical properties of the Cloud environment and hardware imposing boundary conditions. It is likely that Clouds will have relatively few bigger machines so scale will continue to be an issue. Cross machine scaling will always be vulnerable to access and synchronization patterns. Overheads of Cloud control will impose lower bounds on granularity of parallel tasks. The latency exhibited by wide-area networks also will impose lower bounds on response time of execution problems. And even if adopted to minimize the effects of these problems, new models will require new programming interfaces to make it work and the refactoring of legacy codes through new compilers and programmers in the loop. Nonetheless, the Cloud is the second and simultaneous revolution impacting the future of HPC along with the use of a new generation of runtime system software and programming models. Together, they are anticipated to inaugurate

the exciting frontiers of Exascale computing by the end of this decade for HPC and the end science it will deliver.

References

[1] G Gao, T Sterling, R Stevens, M Hereld, and W Zhu, *ParalleX: A study of a new parallel computation model*, Parallel and Distributed Processing Symposium. IPDPS 2007, 1–6, 2007.

[2] H Kaiser, M Brodowicz, and T Sterling, *ParalleX: An Advanced Parallel Execution Model for Scaling-Impaired Applications*, Parallel Processing Workshops. 394–401, 2009.

[3] A Tabbal, M Anderson, M Brodowicz, H Kaiser, and T Sterling, *Preliminary Design Examination of the ParalleX System from a Software and Hardware Perspective*, SIGMETRICS Performance Evaluation Review, **38**, 4, 2011.

[4] http://www.clusterresources.com/products.php

[5] D Thain, T. Tannenbaum, and M. Livny, *Distributed Computing in Practice: The Condor Experience*, Concurrency and Computation: Practice and Experience, **17**, no. 2-4, 323–356, Feb-Apr, 2005.

[6] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard, Version 2.2*, High Performance Computing Center Stuttgart, September 2009.

[7] L Dagum and R Menon, *OpenMP: An Industry-Standard API for Shared-Memory Programming*, IEEE Computational Science and Engineering, **5**, 46–55, 1998.

PoS(ISGC 2012)038