Partitioned Global Address Space (PGAS): Bridging The HPC-GRID Gap

A. Djaoui†, S. Scott §, S. Purdie §, D. McNab §, M. Kenyon §, T. Doyle §

† Rutherford Appleton Laboratory, UK

§ University of Glasgow, UK

Contact: Abdeslem.Djaoui@stfc.ac.uk

INTRODUCTION

Traditionally, High Performance Computing (HPC) was targeted at running tightly coupled parallel tasks on large scale supercomputers, while the GRID was limited to running more loosely coupled jobs on geographically dispersed resources. Typically, on HPC supercomputers and high end clusters, OpenMP and MPI are popular options for parallelism at the shared memory and distributed memory levels respectively, with inter-process communication and synchronisation left to MPI. In the GRID, workflows and scripting languages are used for orchestrating, at another distributed parallel level, loosely coupled jobs with communication enabled through file systems. This use of different programming paradigms for loosely and tightly coupled tasks, reflects the characteristics of the different hardware architectures involved in HPC and the GRID; shared and distributed memory machines with low-latency interconnects for HPC and geographically distributed machines with low bandwidth, potentially unreliable network connections for the GRID.

HIERARCHICAL HARDWARE IN HPC AND GRID

New petascale supercomputers such as IBM Blue Gene are build using a hierarchical design where many multicore nodes are combined onto cards, which slot onto server rack midplanes. Many servers racks are then connected together using high bandwidth interconnects. This design hierarchy is reminiscent of collections of clusters in a GRID environment except for the lower bandwidth communication networks between clusters in the GRID. Parallelism is exposed, in both, at more than one distributed level, depending on the intercluster communication speed, in addition to shared memory. Programming models and flexible runtimes that can deal with multi-level task granularity mapping on modern hierarchical hardware would allow porting applications between supercomputers and GRIDs more practical. This can be achieved with existing hybrid models and with the new High Productivity PGAS languages.

MAPPING TASK GRANULARITY ON COLLECTIONS OF CLUSTERS

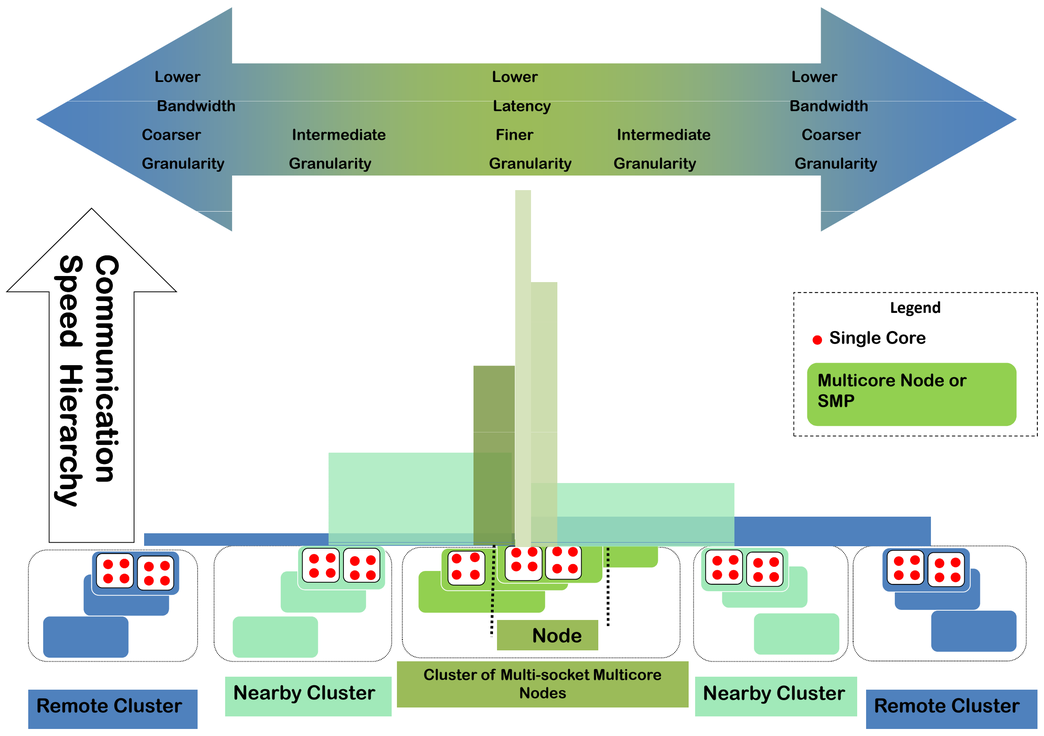

Figure 1: Schematic view of communication speed differences between a task and another, between cores (highest), in another node within the same tightly coupled cluster, in a nearby cluster and in a remote cluster (lowest). Better performance and resource utilisation can be achieved with mapping of application tasks single nodes, clusters, across nearby clusters or across remote clusters, according to their granularity.

Multi-level domain decomposition during application design provides a good abstraction of the new hierarchical hardware. At runtime, good mapping of tasks on hardware resources taking into account their granularity and coupling is needed in order to benefit from cheaper local communication in the presence of significant bandwidth differences between, on the chip, between cores, between nodes and between clusters in a supercomputer or on the GRID.

SHARED, DISTRIBUTED AND HYBRID MODELS

THREADS: Thread programming is a general purpose concurrency model, but can be complicated by the necessity to avoid conflicts between reads and writes to the same memory location by different threads. Native threads are usually not portable to other platforms.

OpenMP: For shared memory, OpenMP is a portable, simpler thread model for C/C++ and Fortran programmers. OpenMP however is limited to (virtual) shared memory and provides little control to users on thread scheduling. Scalability with OpenMP is limited by the available memory.

MPI: MPI addresses the problem of moving data from one processor to another on the same or different cluster. Unlike OpenMP, MPI runtime features allow mapping of processes onto specific nodes or even specific cores thus allowing communication overhead to be minimized.

HYBRIDS: Hybrid MPI-OpenMP and MPI-Threads is a popular model for clusters of multicore. Coupling OpenMP/Threads tasks on shared memory nodes using MPI for communications is the easiest approach, although more complex mixed code scenarios are possible.

MPI-2: Dynamic process creation and management and the ability to establish communication between independent MPI executables, allow composition of coarser grained tasks into larger applications on more than one cluster. Portability between HPC and GRID is much simplified when using similar runtimes.

FUTURE MPI-3: Fault-tolerance in MPI-2 is available in many implementations, but being non standard, results in reduced portability. This, and other issues of performance, scalability and cluster support are currently being addressed in on going effort at the MPI FORUM.

HIGH PRODUCTIVITY PGAS MODELS

PGAS provide a global shared address space on distributed resources and, as such, merges the shared and distributed memory models. By leveraging distributed hardware for application data structures, local memory limitations are overcome, leading to better shared memory scalability. Programming with the PGAS global view style, is much simpler compared to the fragmented view of MPI where the user, not the compiler, has to manage all the details of remote data access and synchronisation.

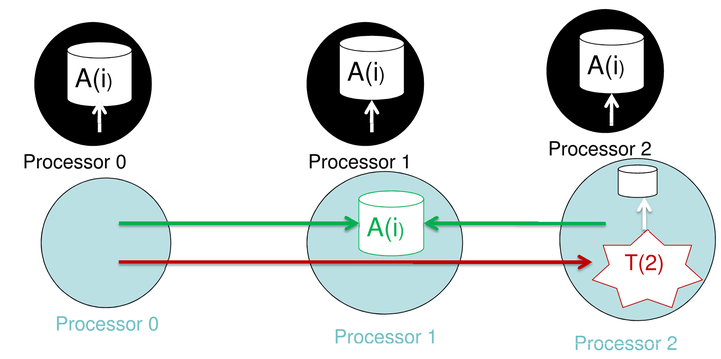

Figure 2: Matrix A is distributed among 3 processors. In the PGAS model index (i) is a global index referencing the same storage location from all 3 processors. In Asynchronous PGAS, processor 0 creates a task T for execution at processor 2, which can access data at processor 2 synchronously and must invoke asynchronous operations to access data at other processes.

High Productivity PGAS such as X10, allows a global view of computation on distributed hardware as well as data. They extend PGAS by allowing processes to invoke computation on other nodes using more robust asynchronous operations. Remote data can only be accessed by spawning asynchronous activities. All existing programming styles (Threads, shared and distributed memory) can be implemented in X10. When combined with a flexible runtime for scheduling computation, High Productivity PGAS is a suitable universal, multiparadigm model, for petascale hierarchical hardware, both in HPC and the GRID.

Science & Technology Facilities Council

ScotGrid Scottish Grid Service