DLB: Dynamic Load Balancing

Load imbalance is a source of performance loss in HPC systems and applications. We have seen that the problem, far from being solved, worsens with the growth of the systems. Our proposal is a dynamic solution to adapt the execution at runtime.

Summary

The load balancing problem concerns both system administrators and application developers. Because it affects both the efficiency of the system and the performance of applications. Specially in HPC the amount of resources that are wasted with a bad load distribution can be extremely large.

Fig 1: Typical application life cycle

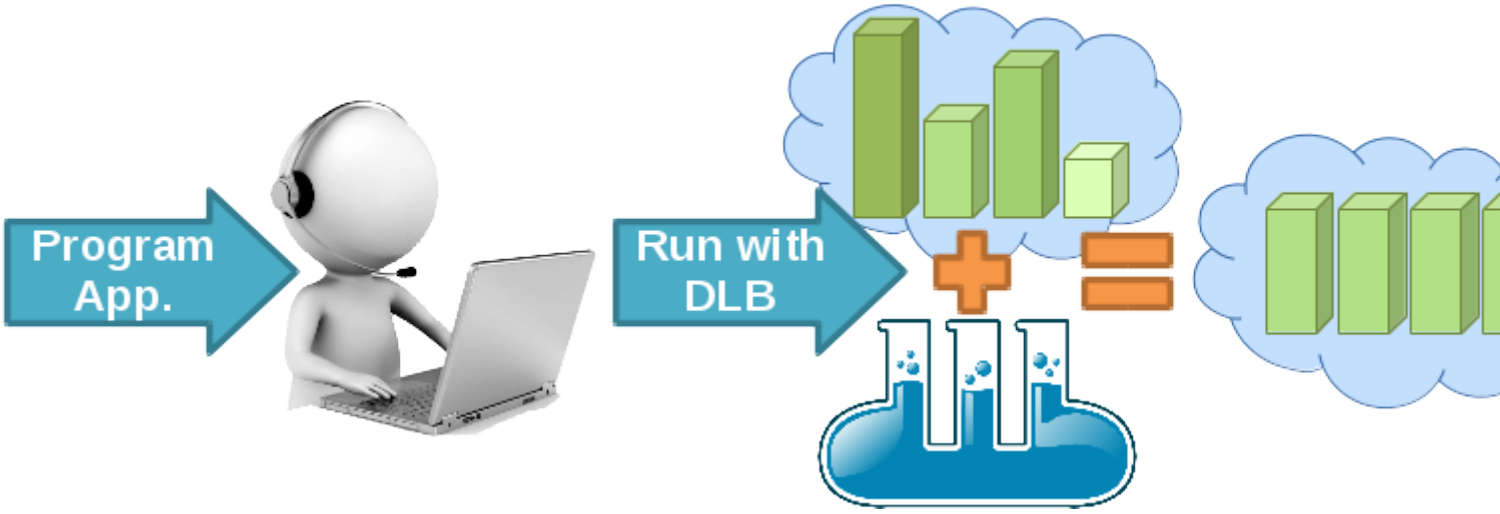

Fig 2: Application life cycle with DLB

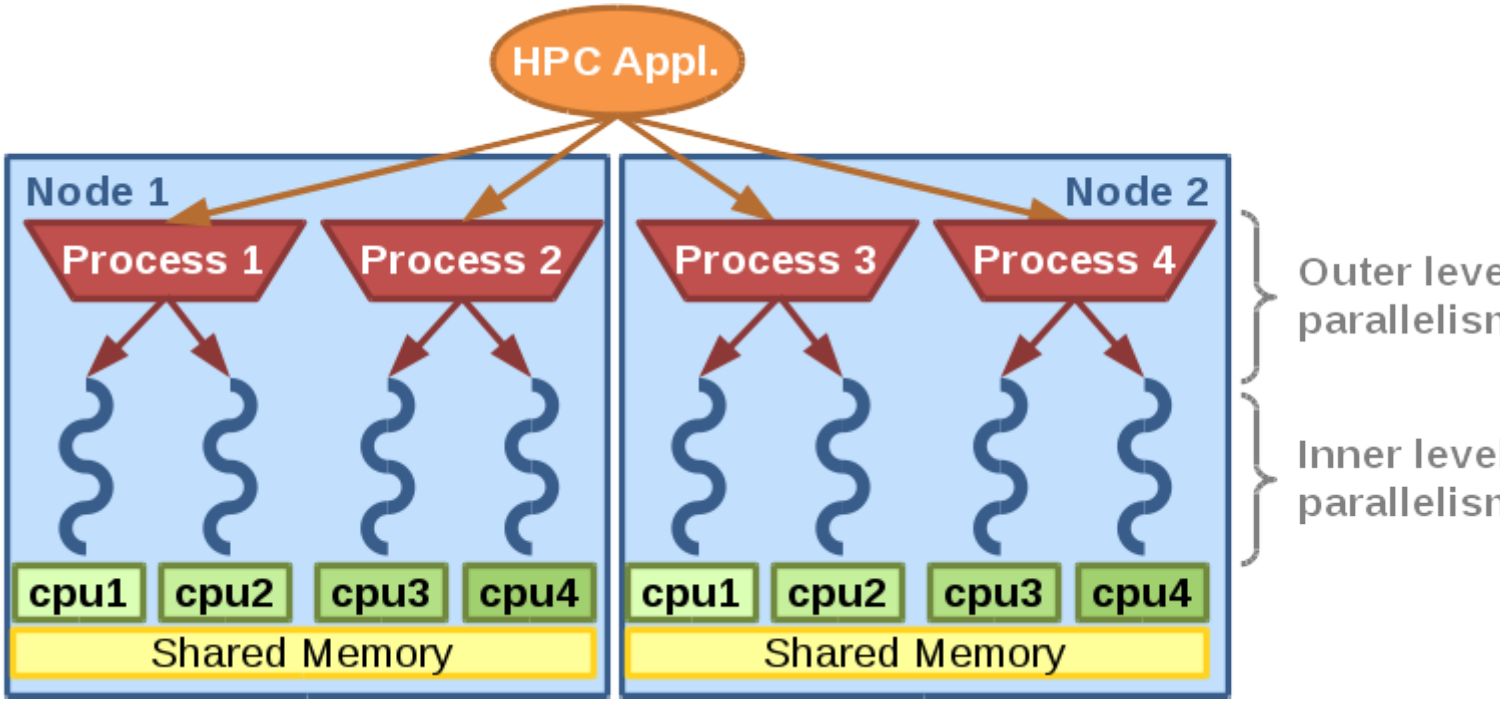

Fig 3: Hybrid application with nested parallelism

HPC developers usually try to solve the load imbalance problem in their own applications. In fig. 1 we can see a typical life cycle when developing an HPC application. This process is time-consuming for the programmer/expert and spends computational resources. Moreover, when talking about highly tuned applications, this cycle must be done for each different architecture the application will run and each input set used.

The objective of DLB is to avoid this cycle and offer a tool that can be used without a previous performance analysis of the application (see fig. 2). Therefore, we will reduce significantly the time spent by the programmers modifying the application, debugging and analyzing the sources of performance loss and also minimize the amount of computational resources necessary to run the performance tests.

DLB is a library devoted to speedup hybrid parallel applications (see fig. 3). And at the same time, DLB improves the efficient use of the computational resources inside a computing node.

To improve the performance of applications and maximize the use of resources inside a computational node DLB will attack the load imbalance at different levels. Instead of fighting the load imbalance DLB adapts the execution at runtime using the malleability of the inner level of parallelism (i.e. changing the number of threads in OpenMP).

This dynamism allows DLB to react to different sources of imbalance: Algorithm, data, hardware architecture and resource availability among others. The DLB approach to redistribute the computational resources, at runtime, depending on the instantaneous demand, can improve the performance in different situations:

- Hybrid applications with an imbalance problem at the outer level of parallelism.
- Hybrid applications with an imbalance problem at the inner level of parallelism.
- Hybrid applications with serialized parts of the code.
- Multiple applications with different parallelism patterns.

Who can use DLB?

Any application written in C, C++ or Fortran in any of the supported parallel programming models. The current supported parallel programming models are the following:

- MPI+OpenMP
- MPI+OmpSs
- OmpSs (Multiple Applications)

We are open to adding support for more programming models in both inner and outer level of parallelism.

How does DLB work?

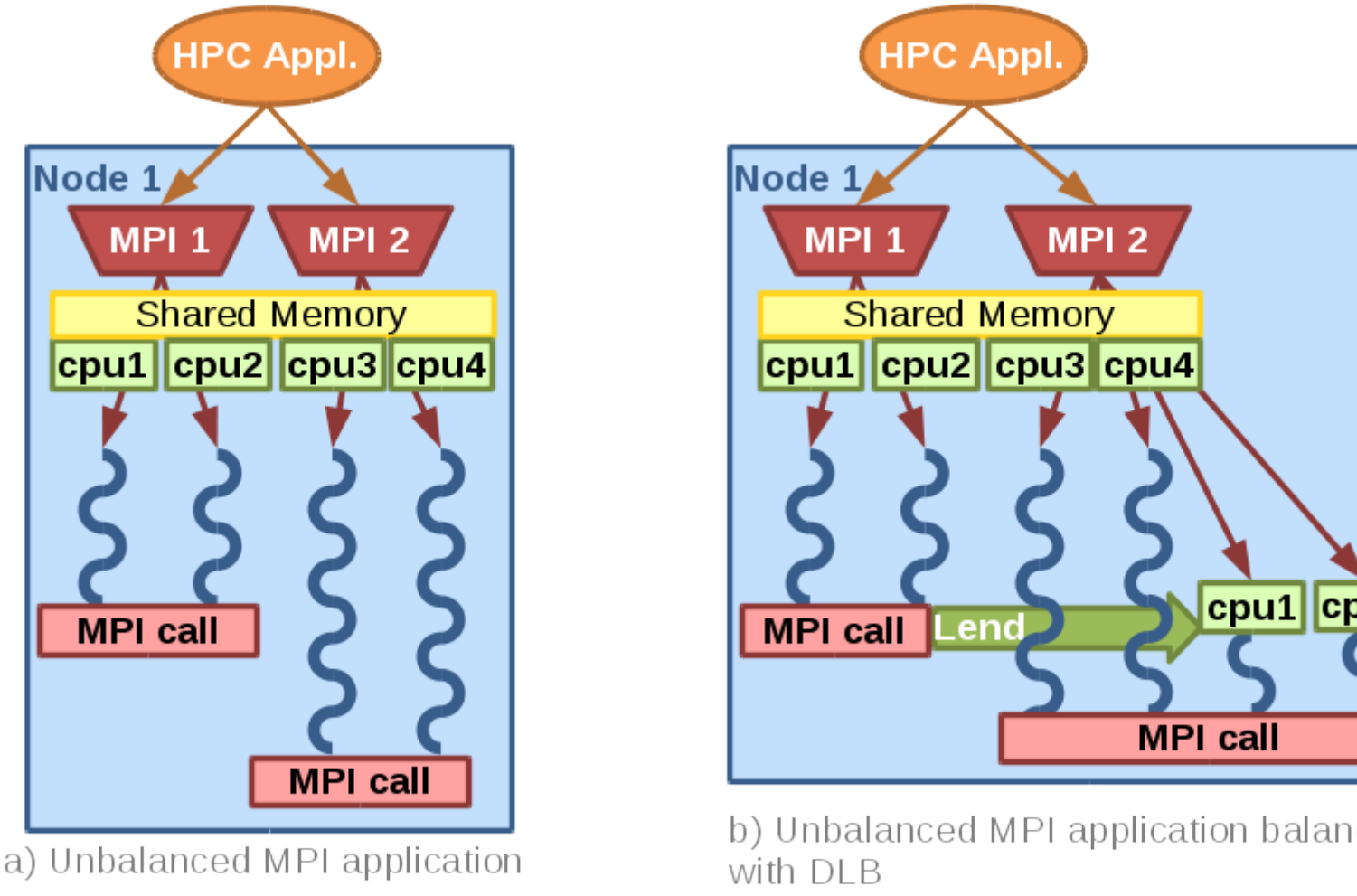

Fig 4: Example of DLB behavior

DLB will use the malleability of the inner level of parallelism to change the number of threads of the different processes running on the same node. There are different load balancing algorithms implemented within DLB. They all rely on this main idea but they target different types of applications or situations.

In fig. 4 we can see an example of a DLB load balancing algorithm. In this case, the application is running two MPI processes on a computing node, with two OpenMP threads each one. When MPI process 1 reaches a blocking MPI call it will lend its assigned CPUs (number 1 and 2) to the second MPI process running in

the same node. This will allow MPI process 2 to finish its computation faster.

What does DLB need?

DLB needs more than one process running in the same computing node (with shared memory). These processes can be MPI processes of the same application or processes of different applications.

We need the processes to use a shared memory parallel programming model (current version supports, OpenMP and OmpSs). The shared memory programming model must be malleable, both from the point of view of the programming model (OpenMP and OmpSs are highly malleable) and from the point of view of the application (do not rely on the number of threads).

Is DLB available?

Yes! DLB is ready to be used. Download the latest version.

Where I can find more information about DLB?

You can find more technical information about DLB in the group web page.

The user guide of DLB can be found here.

Objectives

The aim of DLB is to achieve the most efficient use of the resources in a computational node. Moreover, DLB tries to accomplish this in the most transparent way to the user or application developer. Also to be able to react to any kind of load imbalance, DLB will take all the decisions at runtime.

DLB will also offer tools to other components of the typical software stack (e.g. job scheduler). This tools will have two main functions. On one hand to obtain information about the performance of the application at runtime. On the other hand to offer mechanisms to adapt dynamically the application to the resources.

Barcelona Supercomputing Center - Centro Nacional de Supercomputación

---