StarPU-MPI: Task Programming over Clusters of Machines Enhanced with Accelerators

Cédric Augonnet[1], Olivier Aumage[4], Nathalie Furmento[2], Raymond Namyst[2], and Samuel Thibault[2]

[1] NVIDIA Corporation, Santa Clara, California, USA
[2] LaBRI, CNRS, University of Bordeaux, France
[3] Inria, Bordeaux, France

Abstract. GPUs clusters are becoming widespread HPC platforms. Exploiting them is however challenging, as this requires two separate paradigms (MPI and CUDA or OpenCL) and careful load balancing due to node heterogeneity. Current paradigms usually either limit themselves to offload part of the computation and leave CPUs idle, or require static CPU/GPU work partitioning. We thus have previously proposed StarPU, a runtime system able to dynamically scheduling tasks within a single heterogeneous node. We show how we extended the task paradigm of StarPU with MPI to easily map the task graph on MPI clusters and automatically benefit from optimized execution.
Keywords: Accelerators, GPUs, MPI, Task-based model

1 Adapting the StarPU paradigm to clusters of GPUs

A lot of research has been conducted to allow MPI applications to offload kernels on GPU devices. StarPU [1] is a runtime scheduler for heterogeneous architectures. A StarPU program is written as a graph of tasks, each task working on a set of data. The computation part of the task, the *codelet*, wraps different implementations of the task for each type of device (CPU core, GPU, etc.). StarPU uses a virtual shared memory for automated data transfers between all the heterogeneous processing units to enable scheduling tasks over all these units. By carefully combining StarPU with MPI, we now benefit from both paradigms: scheduling tasks over CPUs and GPUs, and using clusters equipped with GPUs.

The integration of StarPU and MPI uses two strategies, depending on whether we accelerate existing MPI codes, or we add distribution to existing single node applications for exploiting clusters. The first strategy uses a small library we presented in [2], to extend the StarPU's data management layer with MPI-like semantics. The second strategy builds on the task-oriented model of StarPU.

2 Mapping task graphs on clusters

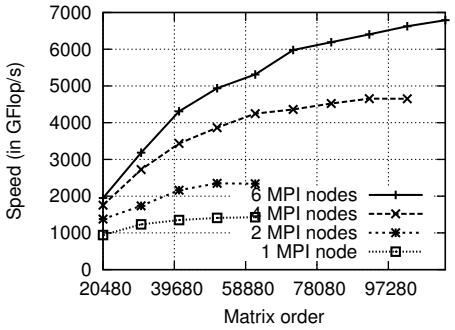
Task graphs are indeed a convenient and portable representation which is not only suited to hybrid accelerator-based machines, but also to clusters of nodes

enhanced with accelerators. The first step is to partition the graph into multiple sub-graphs of tasks that will be executed by the different instances of StarPU. Data dependencies that cross the boundary between the nodes are fulfilled by replacing the dependency with a MPI data transfer that is performed by the means of our MPI-like library. In other words, a node that generates a piece of data required by its neighbour(s) makes a *send* call. Similarly, a node that needs a piece of data that was generated on another node makes a *receive* call. Provided an initial partitioning of the DAG, this shows that our task-based paradigm is also suited for clusters of multicore nodes enhanced with accelerators.

The source code below shows the StarPU-MPI version of the Cholesky decomposition. The MAGMA reference linear algebra library is currently being extended to clusters by using this paradigm.

```
1  for(x = 0; x < X; x++) for(y = 0; y < Y; y++)
2    starpu_matrix_data_register(&A[x][y], 0, &A_tile[x][y], ld, tile_s,tile_s);
3    starpu_data_set_rank(A[x][y], (y%Y_BLK)*X_BLK + (x%X_BLK));
4  for (k = 0; k < Nt; k++)
5    starpu_mpi_insert_task(MPI_COMM_WORLD, &potrf, RW, A[k][k], 0);
6    for (m = k+1; m < Nt; m++)
7      starpu_mpi_insert_task(MPI_COMM_WORLD, &trsm, R, A[k][k], RW, A[m][k], 0);
8    for (m = k+1; m < Nt; m++)
9      for (n = k+1; n < m; n++)
10       starpu_mpi_insert_task(MPI_COMM_WORLD, &gemm,
11                              R, A[m][k], R, A[n][k], RW, A[m][n], 0);
12     starpu_mpi_insert_task(MPI_COMM_WORLD, &syrk, R, A[m][k], RW, A[m][m], 0);
13 starpu_task_wait_for_all();
```

The plot below shows the strong scalability obtained by the Cholesky decomposition on a cluster of machines enhanced with accelerators. Each machine has two Intel Nehalem X5650 sockets with 6 cores each, running at 2.67 GHz, as well as 3 NVIDIA Fermi M2070 GPUs each.



References

1. Augonnet, C., Thibault, S., Namyst, R., Wacrenier, P.A.: StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. In: Proceedings of the 15th International Euro-Par Conference. (2009)
2. Augonnet, C., Clet-Ortega, J., Thibault, S., Namyst, R.: Data-Aware Task Scheduling on Multi-Accelerator based Platforms. In: The 16th International Conference on Parallel and Distributed Systems (ICPADS). (2010)