StarPU-MPI

Task Programming over Clusters of Machines Enhanced with Accelerators

RUNTIME

Cédric Augonnet, Olivier Aumage, Nathalie Furmento, Raymond Namyst, Samuel Thibault

Task Scheduling on Accelerated Cluster Nodes

Clusters follow the general accelerator movement

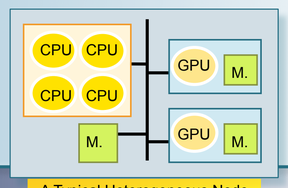
- Distributed Computing
- Node level heterogeneity
  - Multicore CPUs
  - Accelerator boards (GPU, etc)

Applications embrace the task-based approach

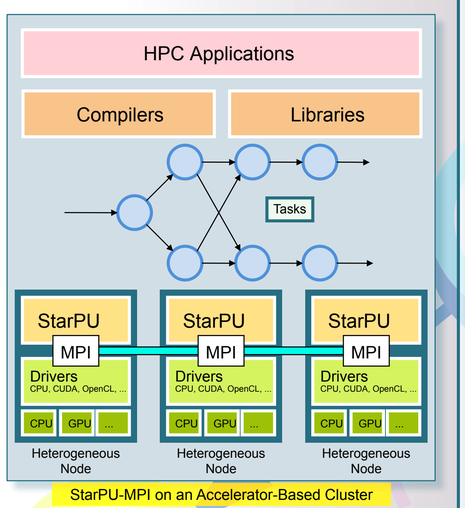
- Portability
- Flexibility
- Portability of Performances

Problem

- Design a task-based programming model for clusters



A Typical Heterogeneous Node



StarPU-MPI on an Accelerator-Based Cluster

The StarPU Paradigm

Process-level task scheduling

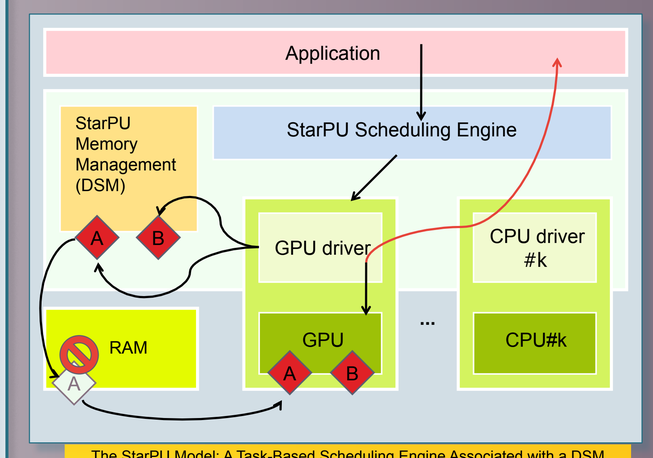
- Use every available computing units
  - CPU cores
  - Accelerated units (GPUs)

Data management using a DSM

- Data transfers to/from embedded memory
- Caching, replication
- Consistency management

Optimization

- Minimize a given cost function
  - Overall execution time
  - Power consumption
- Estimate the cost of upcoming tasks
  - History backlog
  - Parametric task-cost models
- Estimate the cost of upcoming transfers
  - Hardware buses calibration



The StarPU Model: A Task-Based Scheduling Engine Associated with a DSM

The StarPU-MPI Task Model

Goals

- Extend the StarPU model to clusters
- Avoid global cluster-wide scheduler
- Minimize changes to single node clusters

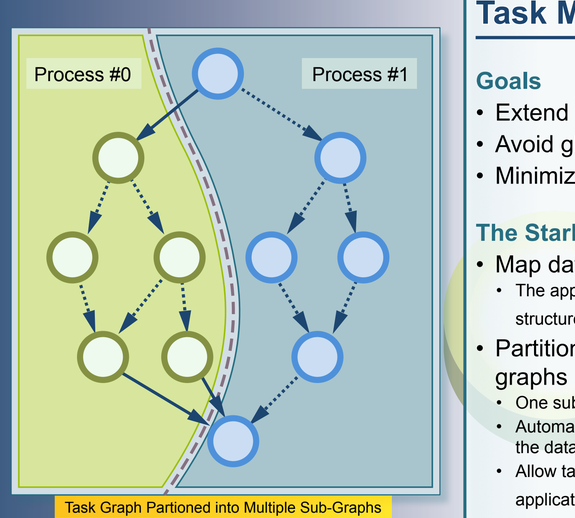
The StarPU-MPI Approach

- Map data onto cluster processes
  - The application controls the mapping of data structures
- Partition the graph into multiple sub-graphs
  - One sub-graph per StarPU instance
  - Automatically execute tasks on the process that own the data to be written to
  - Allow task execution location override by the application
- Handle inter-process dependencies
  - Automatically replace data dependencies across sub-graphs with MPI transfers
  - Implement an optional cache mechanism to eliminate redundant data transfers

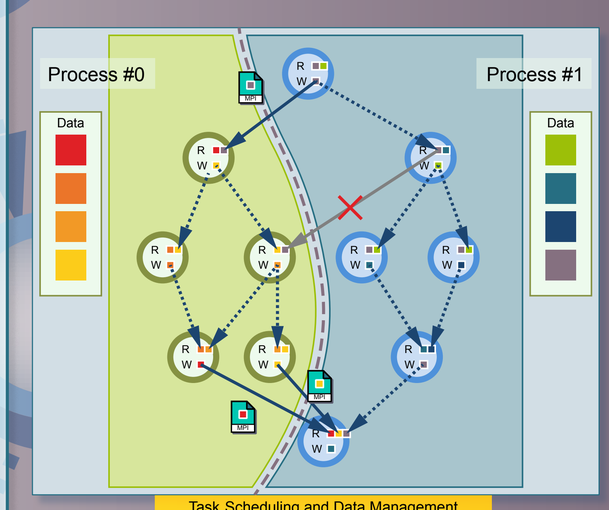
- Preserve the StarPU model within processes
  - StarPU-MPI inherits the scheduling capabilities of StarPU within a process
  - StarPU-MPI inherits the data management capabilities of StarPU within a process
  - StarPU Application codelets can seemlessly be reused for StarPU-MPI Applications
- Optimize data transfers
  - StarPU-MPI transparently overlaps MPI communication with computation

```
for(x = 0; x < X; x++)
    for(y = 0; y < Y; y++)
        starpu_matrix_data_register(&A[x][y], 0, &A_tile[x][y], ld, tile_s,tile_s);
        starpu_data_set_rank(A[x][y], (y%Y_BLK)*X_BLK + (x%X_BLK));

for (k = 0; k < Nt; k++)
    starpu_mpi_insert_task(MPI_COMM_WORLD, &potrf, RW, A[k][k], 0);
    for (m = k+1; m < Nt; m++)
        starpu_mpi_insert_task(MPI_COMM_WORLD, &trsm, R, A[k][k], RW, A[m][k], 0);

    for (m = k+1; m < Nt; m++)
        for (n = k+1; n < m; n++)
            starpu_mpi_insert_task(MPI_COMM_WORLD, &gemm, R, A[m][k], R, A[n][k], RW, A[m][n], 0);

        starpu_mpi_insert_task(MPI_COMM_WORLD, &syrk, R, A[m][k], RW, A[m][m], 0);

starpu_task_wait_for_all();
```

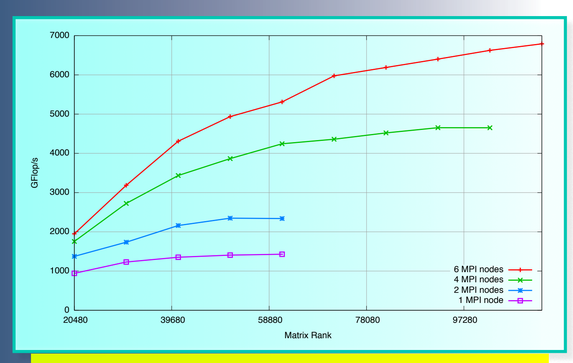
Example Code: Cholesky Decomposition using StarPU-MPI



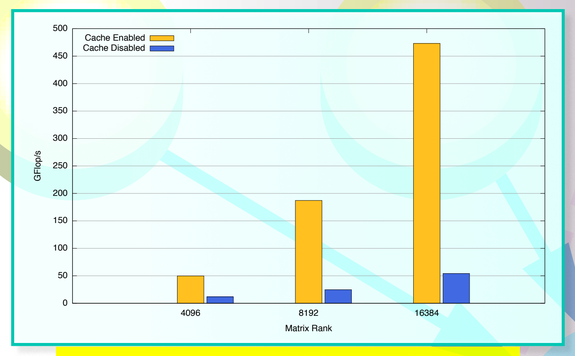
Task Graph Partioned into Multiple Sub-Graphs



Task Scheduling and Data Management



Strong scalability of a Cholesky decomposition over a cluster of machines accelerated with 3 GPUs and 12 CPU cores per MPI node



Effect of the StarPU-MPI cache mechanism on the efficiency of a Cholesky decomposition computation using two MPI nodes

Perspectives

Availability

- On-going work to include StarPU-MPI support in the MORSE software distribution

Scalability

- Optimize global graph traversal
- Prune submissions of non-local tasks that don't have cluster-wide impact

Granularity

- Automatically adapt task workload to available hardware
  - Aggregate tasks to improve the computation/overhead ratio
  - Split tasks to improve load balancing and accomodate many-core resources

Flexibility

- Adapt sessions to changing contexts
  - Changes in cores availability
  - Changes in nodes availability

Evaluation

Strong Scalability

The plot above 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.

Effectiveness of the Cache Mechanism

The plot above shows the effectiveness of the cache mechanism implemented by the data management layer of StarPU-MPI.

The data cache dramatically improves the computation efficiency by avoiding unnecessary MPI transfers.

Contact

Runtime Team / StarPU

- Inria – LaBRI, Bordeaux, France
- Web: `http://runtime.bordeaux.inria.fr/StarPU/`
- E-mail: `starpu-devel@lists.gforge.inria.fr`

- Grants
  - EU FP7 PEPPHER [contract num 248481]
  - ANR ProHMPT [ANR-08-COSI-013]
  - GENCI DARI [t2012066431]