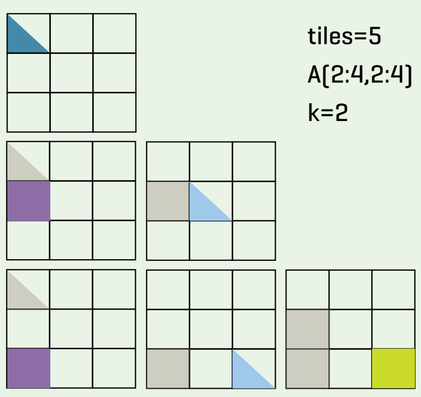
Scheduling Cholesky Factorization on Multicore Architectures with GPU Accelerators

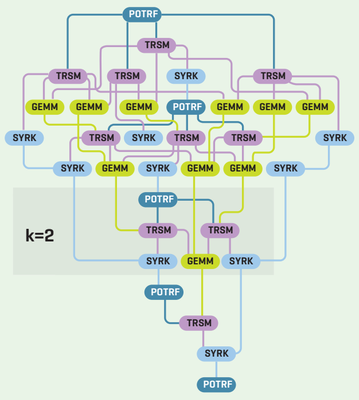
Emmanuel Agullo
Cédric Augonnet
Jack Dongarra
Hatem Ltaief
Raymond Namyst
Rajib Nath
Jean Roman
Samuel Thibault
Stanimire Tomov

Although the hardware has dramatically changed in the last few years, nodes of multicore chips augmented by Graphics Processing Units (GPUs) seem to be a trend of major importance. Previous approaches for scheduling dense linear operations on such a complex node led to high performance but at the double cost of not using the potential of all the cores and producing a static and non-generic code. We schedule dense linear algebra operations on multicore architectures with GPU accelerators thanks to a runtime system capable of using the full potential of the node and that handles the data coherency. High-level algorithms – such as the Tile Cholesky Factorization – are represented as collections of tasks/kernels with a data-driven execution order. The kernels are taken from the PLASMA and MAGMA libraries and their execution scheduled through the StarPU runtime.

Tile Cholesky Factorization

For k=0..tiles-1
  A[k][k]<- **POTRF** (A[k][k])
  For m=k+1..tiles-1
    A[m][k] **TRSM** (A[k][k],A[m,k])
  For n=k+1..tiles-1
    A[n][n]<- **SYRK** (A[n,k],A[n,n])
    For m=n+1..tiles=1
      A[m][n]<- **GEMM** (A[m,k],A[n][k],A[m][n])



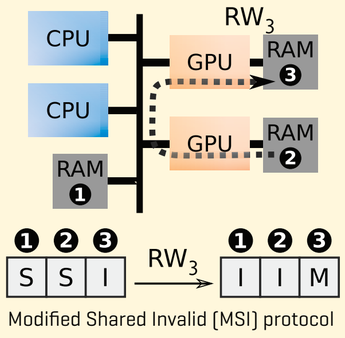


- Fine granularity
- Tile layout
- Asynchronism
- DAG to schedule

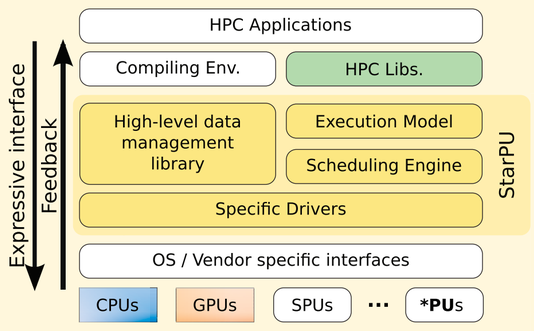
StarPU Runtime System

Download at http://runtime.bordeaux.inria.fr/StarPU/

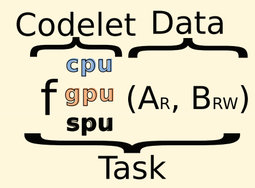
Transparent data management



A Unified Runtime System for Heterogeneous Architectures

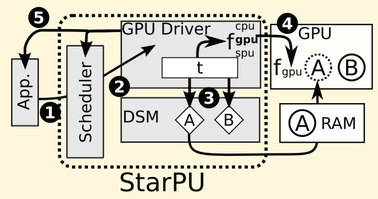


A Unified Task Abstraction



1. Asynchronously submit tasks
2. Schedule tasks onto processing units
3. Ensure data availability & coherency
4. Offload computation
5. Notify task termination

A Unified Execution Model



- Exploit all resouces
- Hide low-level complexity
- Performance portability

PLASMA CPU BLAS

Download at http://icl.cs.utk.edu/plasma/

Principle
- New BLAS (for QR and LU)
- Triangular on top of square

Cholesky Factorization
- Standard BLAS
- Wrapper on top of vendor

MAGMA GPU BLAS

Download at http://icl.cs.utk.edu/magma/

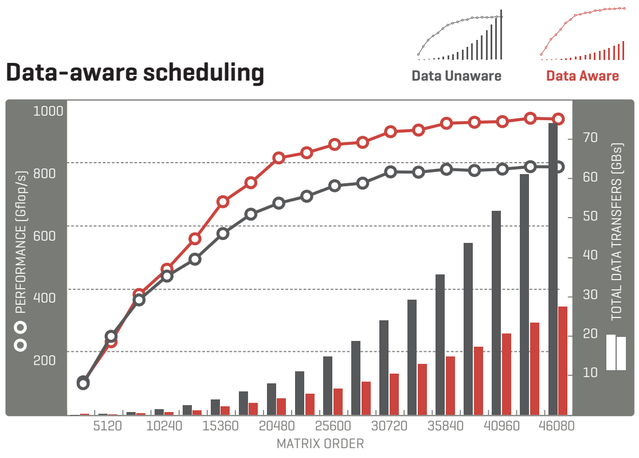
Principle
- CUDA BLAS
- Possibly hybrid CPU/GPU
- Auto-tuned

Cholesky Factorization
- Optimized SYRK and GEMM
- Special handling for TRSM
- Hybrid Kernels (magma_spotrf)

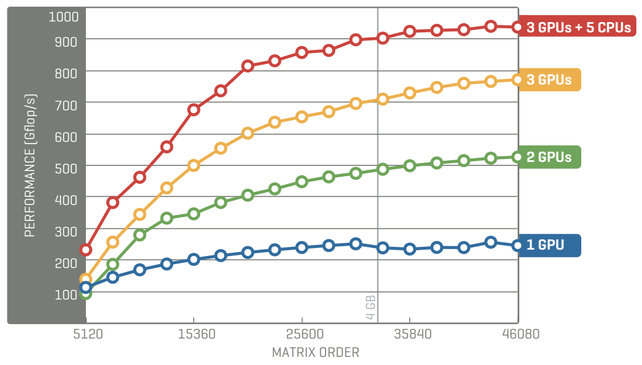
Performance Results

CPU Intel Nehalem X5550 @ 2.67 GHz, 8 cores - sgemm peak 20 GFlop/s per core
3 NVIDA FX5800 GPUs @ 1.30 GHz - sgemm peak 333 GFlop/s per CPU

Data-aware scheduling



Scalability



On-going Work

QR/LU Factorization
- Use of PLASMA CPU kernels
- Needs for new GPU kernels
- Coherency of Hybrid CPU/GPU kernels

Communication-Avoiding QR
- New high-level algorithm
- Needs for new GPU and CPU kernels

Distributed Memory
- Step1: One StarPU instance per node
- Step2: Distributed shared memory

SPONSORED BY