Scheduling Strategies and Bounds for Cholesky Factorization on Heterogeneous Platforms

Student: Suraj Kumar
Advisors: Emmanuel Agullo, Olivier Beaumont and Lionel Eyraud-Dubois
INRIA Bordeaux, France

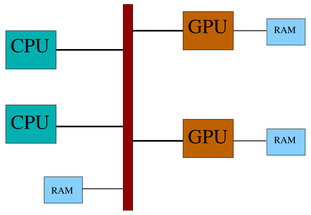
Introduction

Due to massive computational power of accelerators at limited cost, heterogeneous systems (especially CPUs with some GPUs) are very common in HPC field. Therefore, the HPC community is deeply interested in exploiting the full capability of heterogeneous systems for different Linear Algebra kernels.

The goal of this PhD is to understand deeply the behavior of task based scheduling algorithms and propose new scheduling strategies for heterogeneous systems. We present here scheduling analysis of one of these kernels, namely the Cholesky factorization.

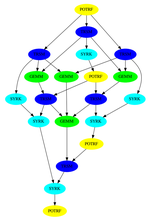
Context

Heterogeneous Platform



- Different types of PUs
- Affinity towards particular behavior
- Hard to program

Task Graph



- Direct acyclic graph
- Node represents task
- Edge represents dependency

- **Task**: A computational code block
- **Scheduling**: When and where to execute
- **Runtime**: A task programming library for scheduling (e.g. StarPU, PaRSEC, Quark)
- **Simulator**: A toolkit to perform simulation (e.g. SimGrid)
- **Linear Algebra Libraries**: MORSE, DPLASMA, MAGMA, LAPACK, etc.

Cholesky Factorization

Cholesky factorization is mainly used to solve linear system of equations $\mathbf{Ax} = \mathbf{b}$, where A is a positive definite matrix. It decomposes matrix A into a lower triangular matrix L, such that $A = LL^{\mathsf{T}}$.

Tile Algorithm

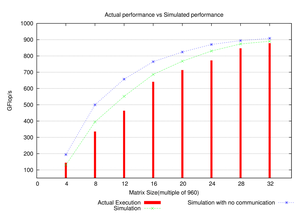
Tile size is tuned to fully exploit the accelerators (GPUs) computing capabilities.

Tile Cholesky factorization has four different types of kernels, whose acceleration factors on GPU are quite different. It has N POTRF, $\frac{N(N-1)}{2}$ TRSM, $\frac{N(N-1)}{2}$ SYRK and $\frac{N(N-1)(N-2)}{6}$ GEMM tasks for $N \times N$ tile matrix. Priorities of tasks can be computed offline based on their distance to exit(last) task.

Algorithm: Tile Cholesky Factorization

```
for k = 0 to N − 1 do
    A[k][k] ← POTRF(A[k][k]);
    for i = k + 1 to N − 1 do
        A[i][k] ← TRSM(A[k][k], A[i][k]);
    end for
    for j = k + 1 to N − 1 do
        A[j][j] ← SYRK(A[j][k], A[j][j]);
        for i = j + 1 to N − 1 do
            A[i][j] ← GEMM(A[i][k], A[j][k], A[i][j]);
        end for
    end for
end for
```

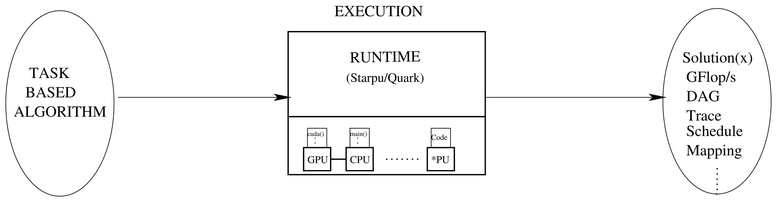
Cholesky Factorization: A compute-intensive kernel



- Single heterogeneous node performance
- Actual execution with StarPU scheduler
- Simulation with StarPU+SimGrid
- Moderate impact of communication on performance
- Cholesky: A compute-intensive kernel

Proposed Framework

Actual Execution

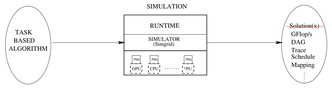


Our Approach

Simulation
- Use simulation for reproducibility
- Simplify the problem
  - Remove communication costs

Theoretical Analysis
- Assess the quality of schedule
- Bound on achievable solution

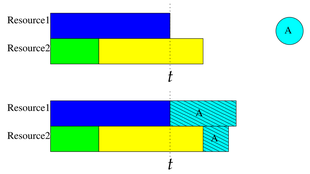
Simulation (No Comm)



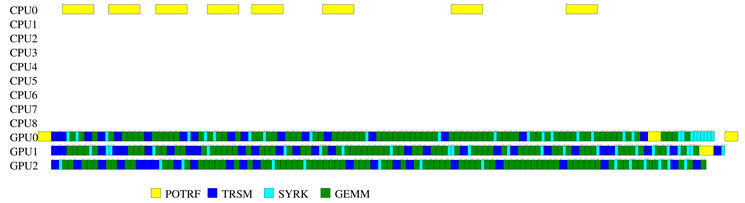
Theoretical Analysis

Task Centric Dynamic Scheduling

- heft scheduling
- Based on heterogeneous earliest finish time
- A is highest priority ready task at t
- Completion time on resource2 is minimum
- Task A schedules on resource2



Cholesky Factorization with *heft*



heft trace for 12 ×12 tile matrix

- Emphasis on critical task
- Poor CPU utilization
- Performance: 686 GFlop/s

Theoretical Lower Bound on Execution time

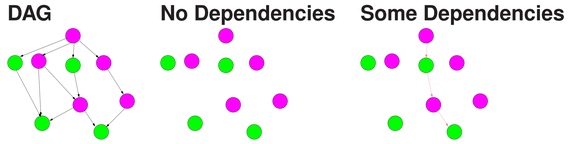
- Solve linear program to find makespan
- One variable for each task and resource pair

minimize l such that

$\forall i$, only one type of resource executes task i : OnlyOne$(b_{i1}, \ldots, b_{iR})$

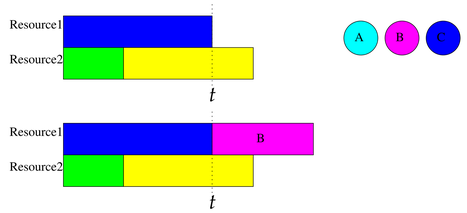
$\forall r$, load on M_r resources of type r within the makespan l : $\sum_i b_{ir} T_{ir} \leq l \times M_r$

Add paths which are larger than l as constraints and repeat the whole procedure



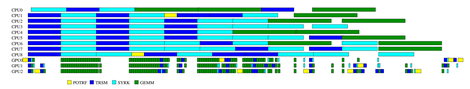
Resource Centric Dynamic Scheduling

- heteroprio scheduling
- Based on affinities of resources
- A is highest priority ready task at t
- Resource1 is best suited to task B
- Resource1 selects task B



Cholesky Factorization with *heteroprio*

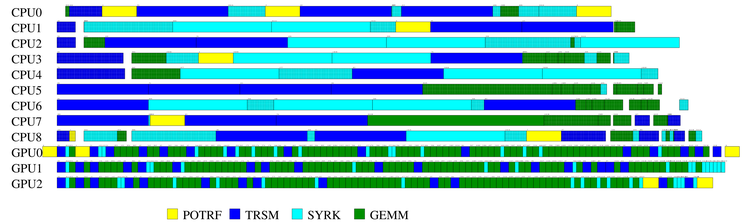
- Good CPU utilization
- Significant idle time on GPUs
- Ignorance of critical path
- Poor performance: 431 GFlop/s



heteroprio trace for 12 ×12 tile matrix

Corrections to *heteroprio* Scheduler

- Spoliation: An idle GPU restarts highest priority CPU task if it finishes earlier
- Progress: GPU selects highest priority task among all tasks of similar acceleration factors
- NonBlocking CPU: CPU selects lowest priority task among all tasks of same acceleration factor
- POTRF preempted on CPU (less accelerated on GPU)
- NonBlocking: GPU restarts highest priority GEMM/SYRK task if it finishes earlier



"*heteroprio* with corrections" trace for 12 ×12 tile matrix
(Aborted tasks due to spoliation in non solid boxes)

- Good Resource utilization
- Performance: 760 GFlop/s

Performance Comparison & Future Work

Experimental Setup

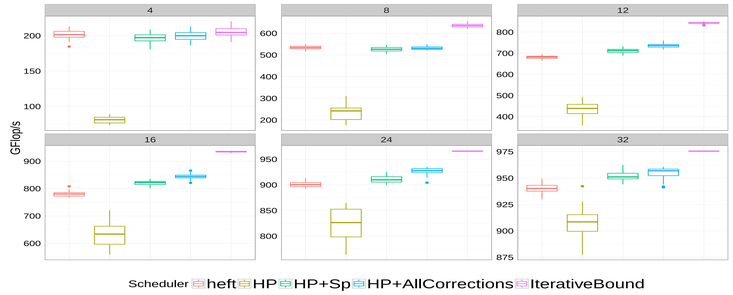
- Double precision Cholesky factorization
- Tile Size = 960
- Heterogeneous platform
  - 3 Nvidia Tesla M2070 GPUs & 9 CPU cores of Intel Xeon X5650 processor

POTRF	TRSM	SYRK	GEMM
≃2×	≃11×	≃26×	≃29×

GPU acceleration ratio over CPU core

Performance Comparison

±10% noisy execution timings(30 runs)



Ongoing Work

- Actual execution with heteroprio schedule
- Consider communication costs
- Implementation of heteroprio scheduler in actual execution
- Sparse linear algebra kernels