A Makespan Lower Bound for the Scheduling of the Tiled Cholesky Factorization based on ALAP Schedule

Olivier Beaumont, Julien Langou, Willy Quach,
Alena Shilova, **Mathieu Vérité**

Inria
University of Colorado Denver
Northeastern University
université de BORDEAUX

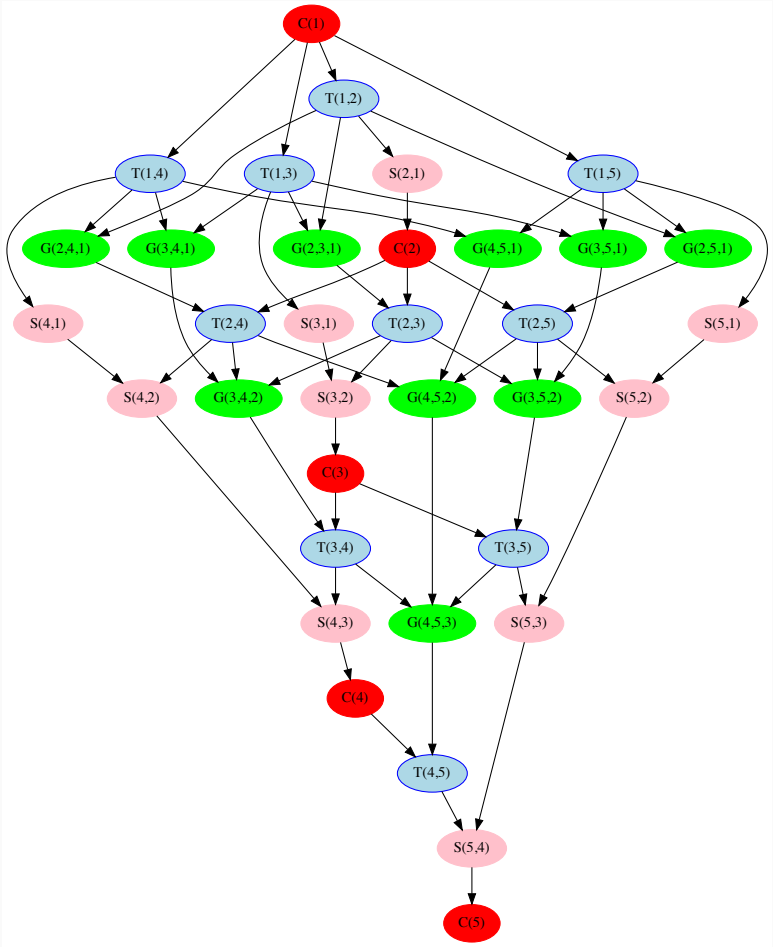
7-8 Décembre 2020
ANR Solharis

Introduction

We consider tiled Cholesky factorization

- typical linear algebra applications
- many tasks ($N^3/6$) but very few different kernels
  - POTRF, TRSM, SYRK and GEMM
- a very regular, but complex, structure

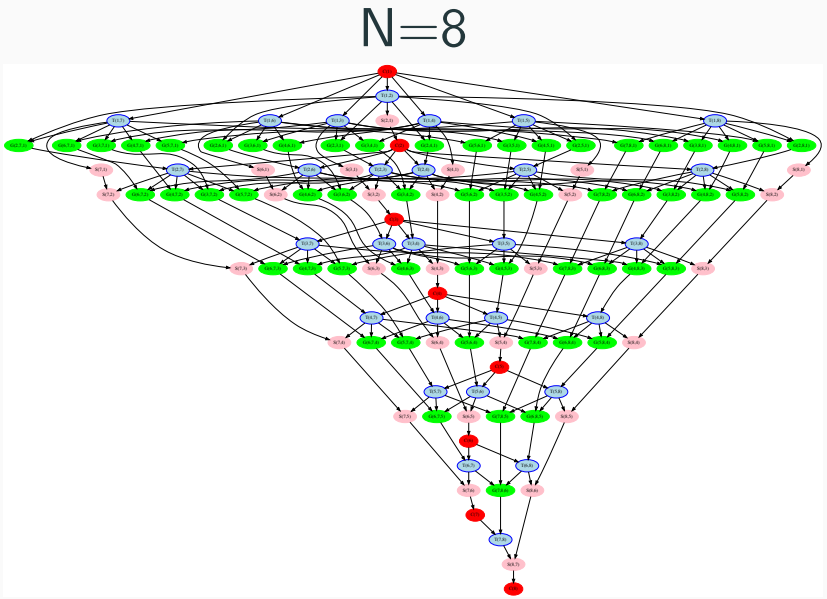
N=5



Introduction

We consider tiled Cholesky factorization

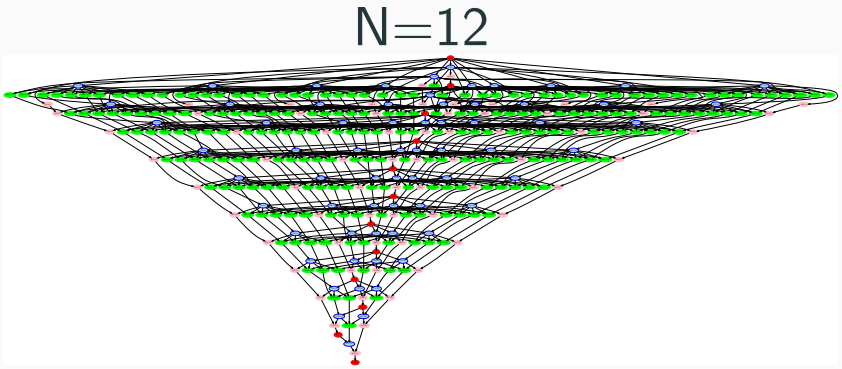
- typical linear algebra applications
- many tasks ($N^3/6$) but very few different kernels
  - POTRF, TRSM, SYRK and GEMM
- a very regular, but complex, structure



Introduction

We consider tiled Cholesky factorization

- typical linear algebra applications
- many tasks ($N^3/6$) but very few different kernels
  - POTRF, TRSM, SYRK and GEMM
- a very regular, but complex, structure



Objective

Our goal in this paper:

- find a lower bound on execution time of Cholesky factorization
  - that depends only of C, T, G, S (durations of the different kernels)
  - and N (size of the problem)
  - no schedule can achieve better makespan
- assuming that we can overlap computations and communications

Objective

Our goal in this paper:

- find a lower bound on execution time of Cholesky factorization
  - that depends only of C, T, G, S (durations of the different kernels)
  - and N (size of the problem)
  - no schedule can achieve better makespan
- assuming that we can overlap computations and communications
- that is tighter than the trivial bound $\max(CP, W/P)$
  - CP critical path: longer path from the first task to the last one
  - W total work: sum of the durations of all tasks
  - P: number of (homogeneous) processors

Objective

Our goal in this paper:

- find a lower bound on execution time of Cholesky factorization
  - that depends only of C, T, G, S (durations of the different kernels)
  - and N (size of the problem)
  - no schedule can achieve better makespan
- assuming that we can overlap computations and communications
- that is tighter than the trivial bound $\max(CP, W/P)$
  - CP critical path: longer path from the first task to the last one
  - W total work: sum of the durations of all tasks
  - P: number of (homogeneous) processors

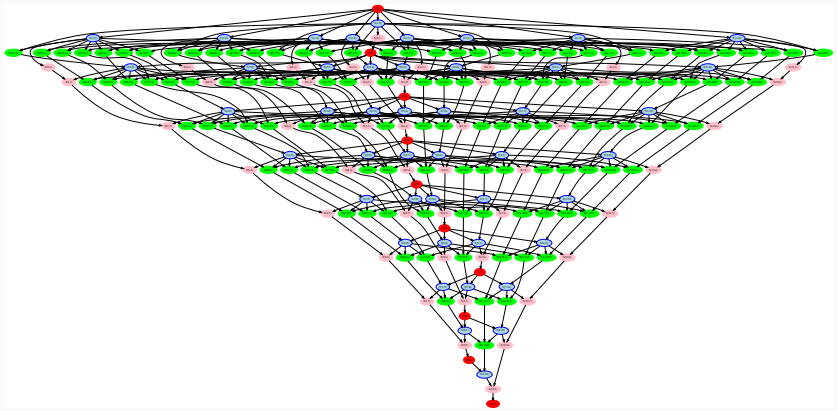
Why is it useful?

- better understand the computation dependencies and parallelism
- better estimate the quality of the implementations
  - How far down is an implementation from the bound?
  - Cholesky is a typical evaluation kernel for dynamic schedulers (StarPU, DaGUE, QUARK,...)

Analysis of Cholesky DAG

Intuition: N size, P number of processors

- The graph widens very quickly
  - First 3 layers of width (roughly) $1, N$ and $N^2/2$:
  - it is easy to keep processors busy
- In the middle part, there is a lot of work
  - it is enough that the processors are active all the time
- The graph is pretty thin at the end
  - with complex dependencies:
  - a careful schedule is required so as to avoid idle time

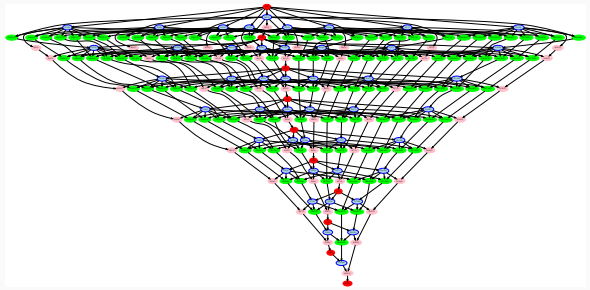


Cholesky Factorization: ASAP and ALAP Schedules

Characteristics:

- beginning: easy, middle: easy, end: complicated.

- Candidate #1: **ASAP** as soon as possible
  - variants used in runtime schedulers
  - list schedule (keep resources busy if possible)
  - efficient at the beginning of the schedule
  - the schedule at the end is not so efficient (front of processed tasks)

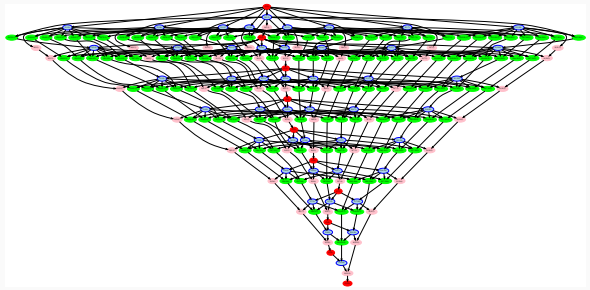


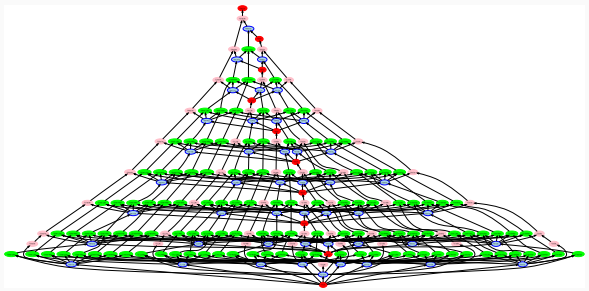
Cholesky Factorization: ASAP and ALAP Schedules

Characteristics:

- beginning: easy, middle: easy, end: complicated.

- Candidate #1: **ASAP** as soon as possible
  - variants used in runtime schedulers
  - list schedule (keep resources busy if possible)
  - efficient at the beginning of the schedule
  - the schedule at the end is not so efficient (front of processed tasks)





- Candidate #2: **ALAP:** as late as possible
  - ASAP on the graph with reversed edges
  - unambiguously defined with unlimited resources
  - and optimal without resource constraints
    - each task is executed exactly at instant t
    - t length of the longest path to the last task
    - including for the first one that defines CP
  - ALAP with resource limitation is a priori not optimal anymore

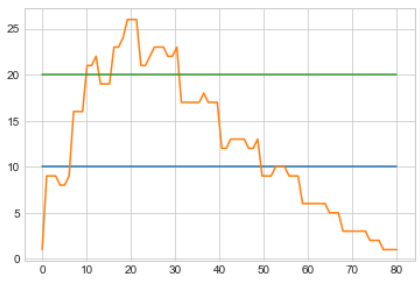
How to do better than $\max(CP, W/P)$?

Tip: find at any instant the number of processors actually used by ALAP when you have an unlimited number of resources at your disposal.

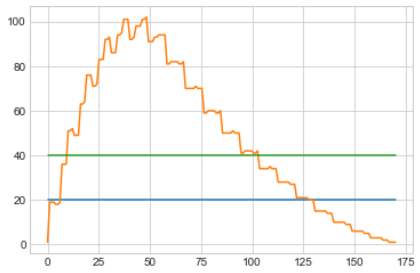
Plots:

- number of running tasks with ALAP
- and no resource limitations

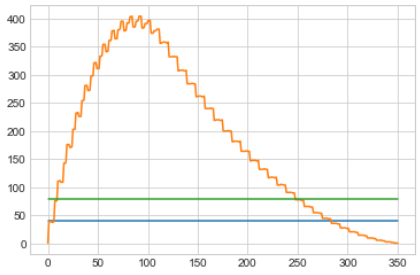
N=10



N=20



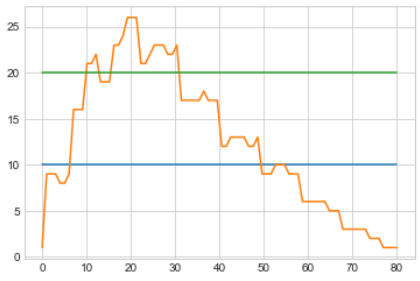
N=40



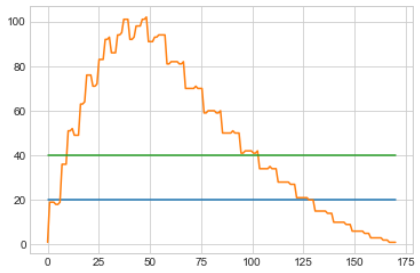
How to do better than $\max(CP, W/P)$?

Tip: find at any instant the number of processors actually used by ALAP when you have an unlimited number of resources at your disposal.

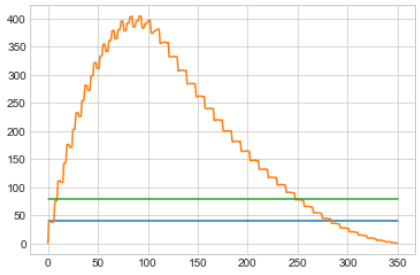
N=10



N=20



N=40



Plots:

- number of running tasks with ALAP
- and no resource limitations

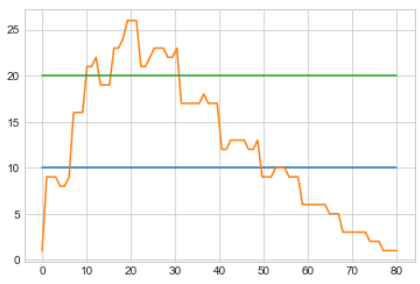
For the three zones:

- at the beginning, when the number of resources is $< P$
  - easy, in practice we just isolate the first POTRF
- in between
  - it is easy to use all processors and to be optimal, there are "too many tasks to do".
- until the number is P
  - how long does it last?

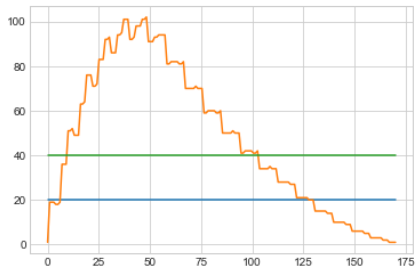
How to do better than $\max(CP, W/P)$?

Tip: find at any instant the number of processors actually used by ALAP when you have an unlimited number of resources at your disposal.

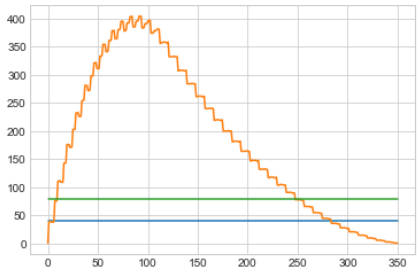
N=10



N=20



N=40



Plots:

- number of running tasks with ALAP
- and no resource limitations

For the three zones:

- at the beginning, when the number of resources is $< P$
  - easy, in practice we just isolate the first POTRF
- in between
  - it is easy to use all processors and to be optimal, there are "too many tasks to do".
- until the number is P
  - how long does it last?

Goal:

find (as tight as possible) overestimation of the orange plot

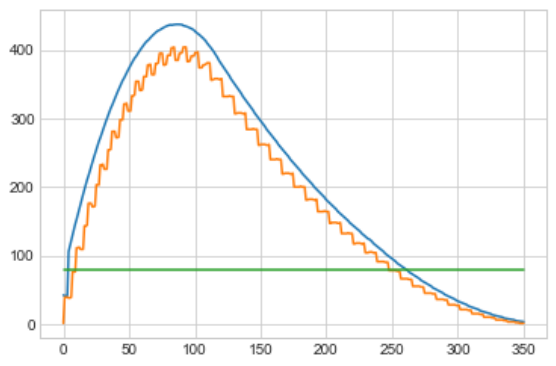
How to finish the proof and find the lower bound?

Indeed, once we know

- an upper bound on the number of tasks of each type
- that are processed at time $M_{ALAP} - d$ in ALAP
- as a polynomial of degree 2: $P(d)$.

- Find d such that $P(d_P) = P$
- No schedule can do more work than ALAP during the last d_P time units

N=40

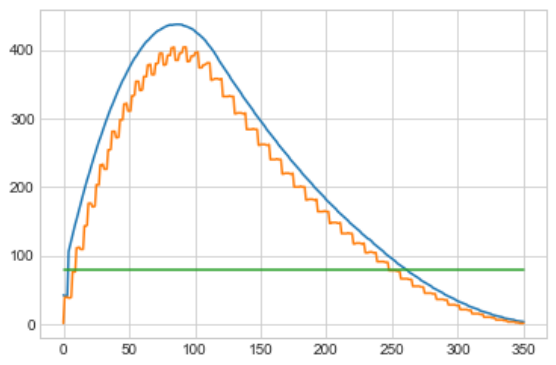


How to finish the proof and find the lower bound?

Indeed, once we know

- an upper bound on the number of tasks of each type
- that are processed at time $M_{ALAP} - d$ in ALAP
- as a polynomial of degree 2: $P(d)$.

N=40



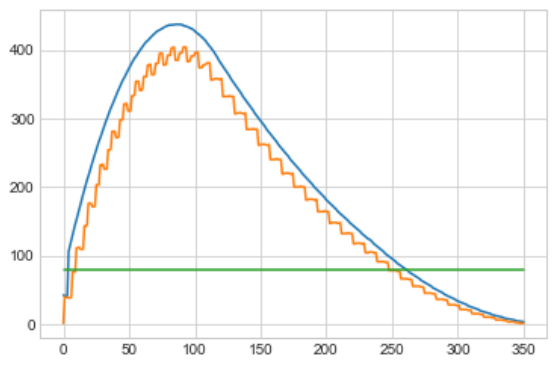
- Find d such that $P(d_P) = P$
- No schedule can do more work than ALAP during the last d_P time units
- If $W_{ALAP}(d_P)$ denotes the work performed by ALAP during the last d_P time units
- No schedule can last less than $\frac{W_{total} - W_{ALAP}(d_P)}{P} + d_P$
- which provides a lower bound.

How to finish the proof and find the lower bound?

Indeed, once we know

- an upper bound on the number of tasks of each type
- that are processed at time $M_{ALAP} - d$ in ALAP
- as a polynomial of degree 2: $P(d)$.

N=40



- Find d such that $P(d_P) = P$
- No schedule can do more work than ALAP during the last d_P time units
- If $W_{ALAP}(d_P)$ denotes the work performed by ALAP during the last d_P time units
- No schedule can last less than $\frac{W_{total} - W_{ALAP}(d_P)}{P} + d_P$
- which provides a lower bound.
- can be slightly improved by considering what happens at the beginning

Critical Paths (1)

- For each task,
  - first compute the length of the longest path from the beginning of this task to the end task
- Good news: the equations depend only on a condition
  - $S + C \leq G$ (CPU case) or $S + C > G$ (GPU case)
    - Typical (normalized) values for the kernels:

	$C(\mathrm{POTRF})$	$S(\mathrm{SYRK})$	$T(\mathrm{TRSM})$	$G(\mathrm{GEMM})$
GPU	1.00	0.11	0.30	0.15
CPU	1.00	4.24	3.91	7.77

  - and we get equations of the type (CPU case)
    - $\mathrm{CP} = 2C + T + S + (n-2)(T+G)$
    - $\mathrm{POTRF}(i)$, $L(C,i) = \mathrm{CP} - (i-1)(T+G)$.
    - $\mathrm{TRSM}(i,j)$,: $L(T,i,j) = \mathrm{CP} - C - (i-1)(T+G)$.
    - $\mathrm{SYRK}(i,j)$,: $L(S,i,j) = \mathrm{CP} - (i-1)(T+G) + (i-j)S$.
    - $\mathrm{SYRK}(n,j)$,: $L(S,n,j) = (n-j)S + C$.
    - $\mathrm{GEMM}(i,j,k)$,: $L(G,i,j,k) = \mathrm{CP} - C + G + T - iT - kG$.

Critical Paths (2)

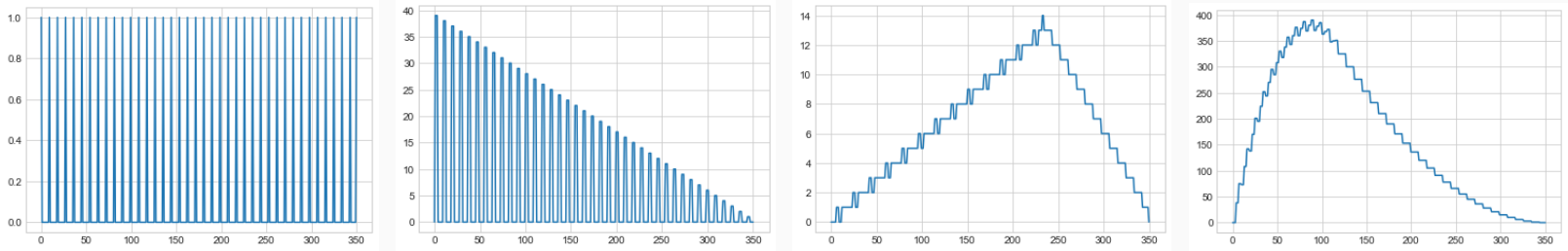
- In ALAP,
  - if the scheduling length is M
  - and $CP(Task)$ denotes its critical path
  - then the task runs at all instants d such that
    - $M - CP(Task) \leq d \leq M - CP(Task) + length(Task)$ $(Eq1)$.
- Our goal is to find the set of running tasks for a fixed value d,
  - Requires to invert $(Eq1)$ to find:
  - for any given value of d the number of tasks (C,T,S or G)
  - that are running at time d!
- More precisely, to get a lower bound at the end
  - We need to overestimate the number of tasks running at instant d

Example

Critical Paths (2)

- In ALAP,
  - if the scheduling length is M
  - and $CP(Task)$ denotes its critical path
  - then the task runs at all instants d such that
    - $M - CP(Task) \leq d \leq M - CP(Task) + length(Task)$ $(Eq1)$.
- Our goal is to find the set of running tasks for a fixed value d,
  - Requires to invert $(Eq1)$ to find:
  - for any given value of d the number of tasks (C,T,S or G)
  - that are running at time d!
- More precisely, to get a lower bound at the end
  - We need to overestimate the number of tasks running at instant d

Example N=40, POTRF, TRSM, SYRK, GEMM



Example: CPU Case, GEMM Tasks (1), small d

Task $\mathrm{GEMM}(i,j,k)$, $1 \leq k < i < j \leq n$ runs at all instants such that $\mathrm{CP} - C + T - iT - kG \leq d \leq \mathrm{CP} - C + T - iT - kG + G$, so that

$$k = \left\lceil \frac{\mathrm{CP} - d - C + T}{G} - \frac{iT}{G} \right\rceil.$$

We simply need to count the number of integer points such that

$$1 \leq \left\lceil \frac{\mathrm{CP} - d - C + T}{G} - \frac{iT}{G} \right\rceil < i < j \leq n.$$

- in general: count the number of integer points in a 2D polyhedron
- It is not very funny (a dozen pages in the research report for all cases) but not very difficult either!

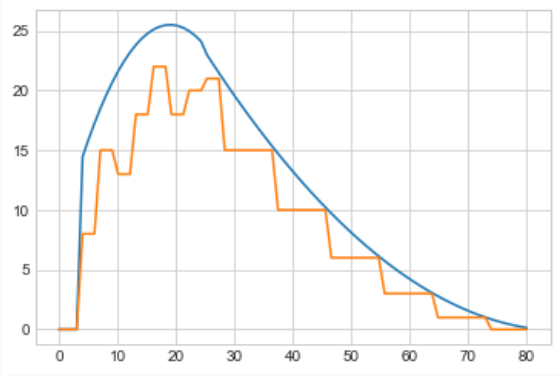
We obtain $\#\mathrm{GEMM}(d) \leq B_1^{\mathrm{GEMM}} d^2 + C_1^{\mathrm{GEMM}} d + D_1^{\mathrm{GEMM}}$, where
$B_1^{\mathrm{GEMM}} = \frac{1}{2(G+T)^2}$,
$C_1^{\mathrm{GEMM}} = \frac{(3G+T-2C-2S)}{2(G+T)^2}$,
$D_1^{\mathrm{GEMM}} = \frac{(G-C-S)(2G+T-C-S)}{2(G+T)^2}$.

Example: CPU Case, GEMM Tasks (2)

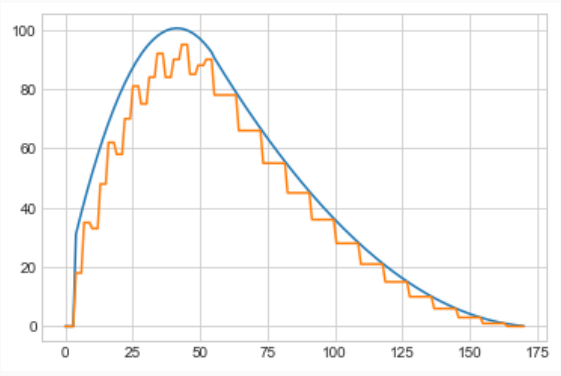
With above bound, we obtain these overestimations

- above bound is valid for small d, another 2D polynomial for large d

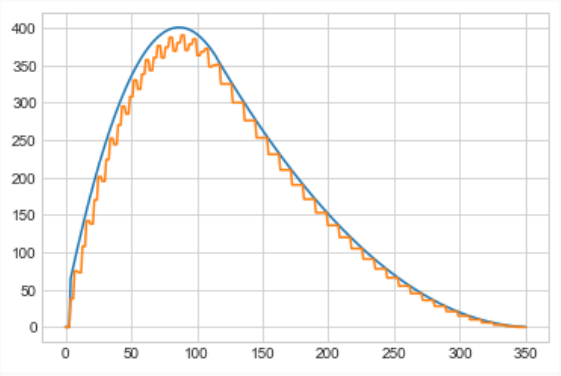
N=10



N=20



N=40

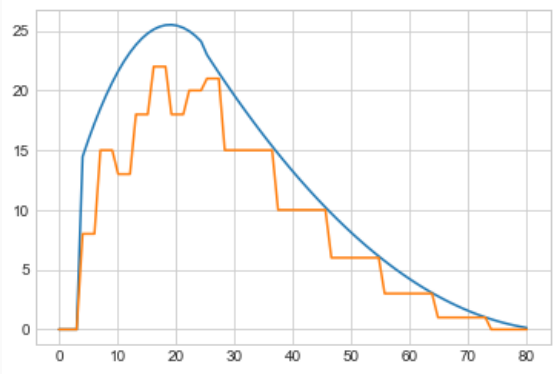


Example: CPU Case, GEMM Tasks (2)

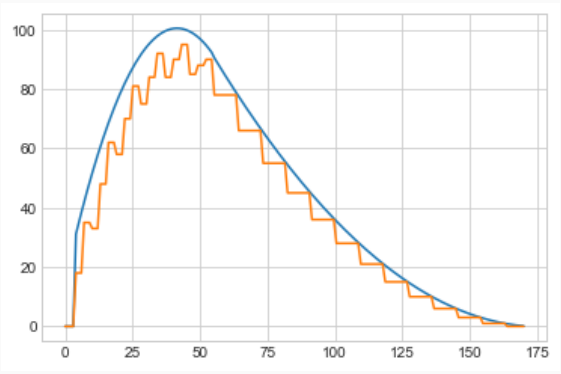
With above bound, we obtain these overestimations

- above bound is valid for small d, another 2D polynomial for large d

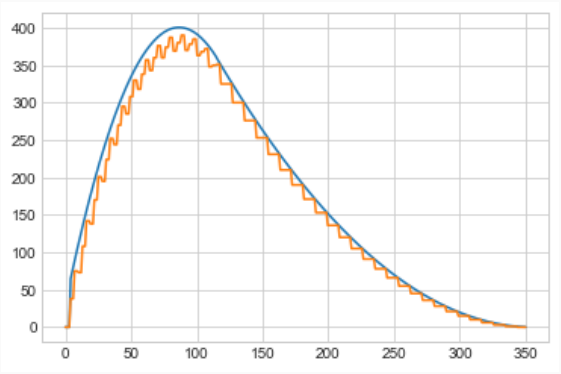
N=10



N=20

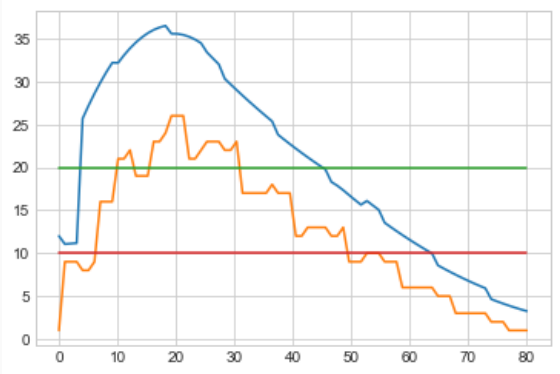


N=40

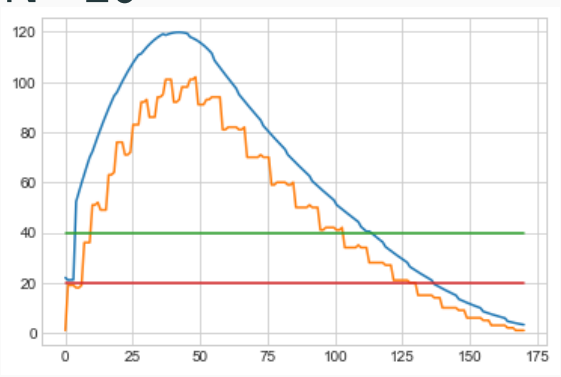


- We did the same for all type of tasks and both CPU and GPU cases
- Upper bounds (CPU case) for the overall number of tasks

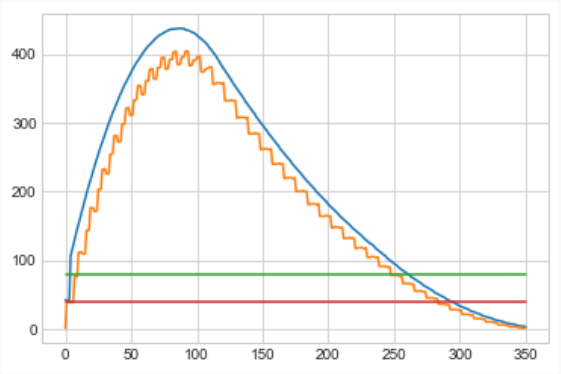
N=10



N=20



N=40

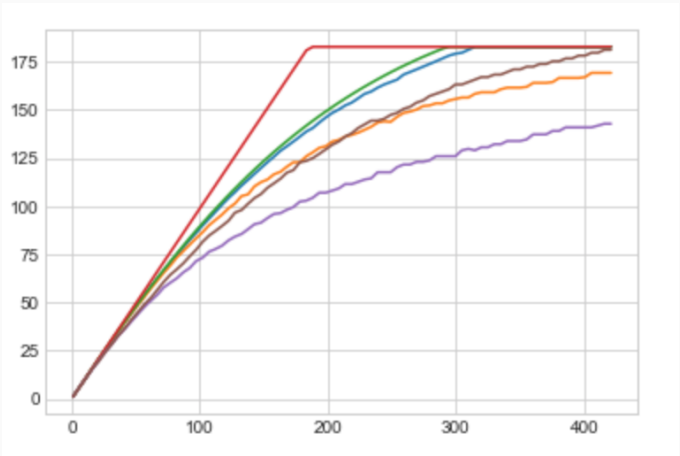


Quality of the bound in practice

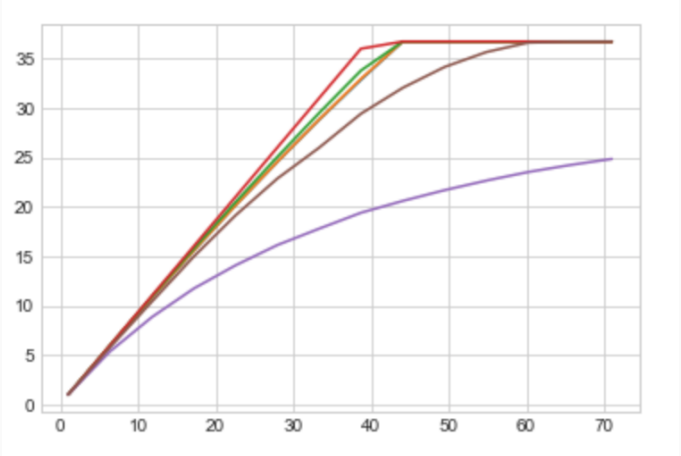
Remember:

- Once we have an upper bound on the number of running tasks $\forall d$
  - we can find a lower bound on the makespan
- we expect that ALAP is a good schedule
- using ALAP, we built an upper bound valid for all possible schedules
- Bounds and Schedules
  - **Bounds:** Old Bound, New Bound
  - **Schedules:** ALAP, ASAP, Kurzak, LAPACK

CPU Case, $N = 40$,

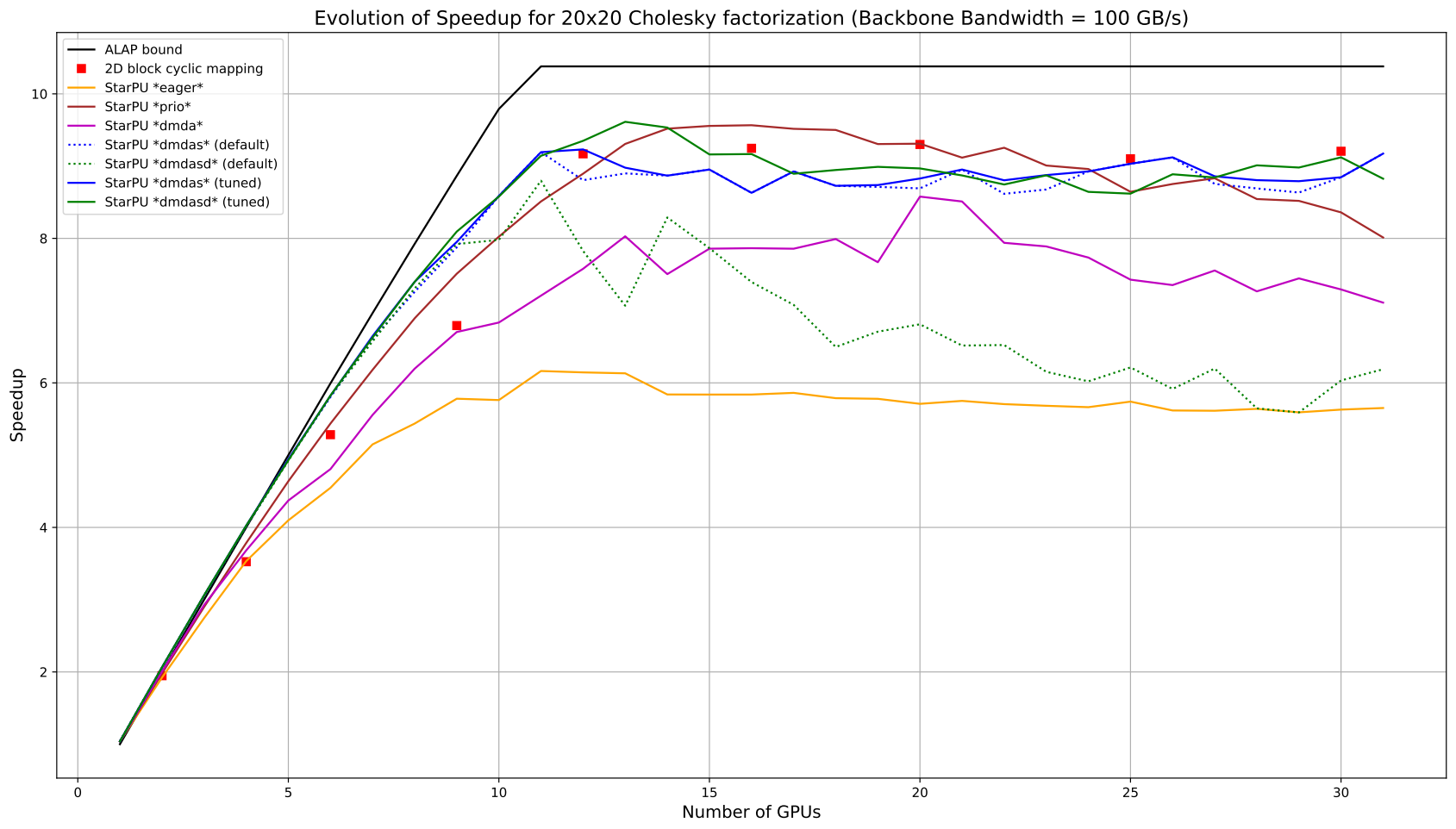


GPU Case, $N = 40$,

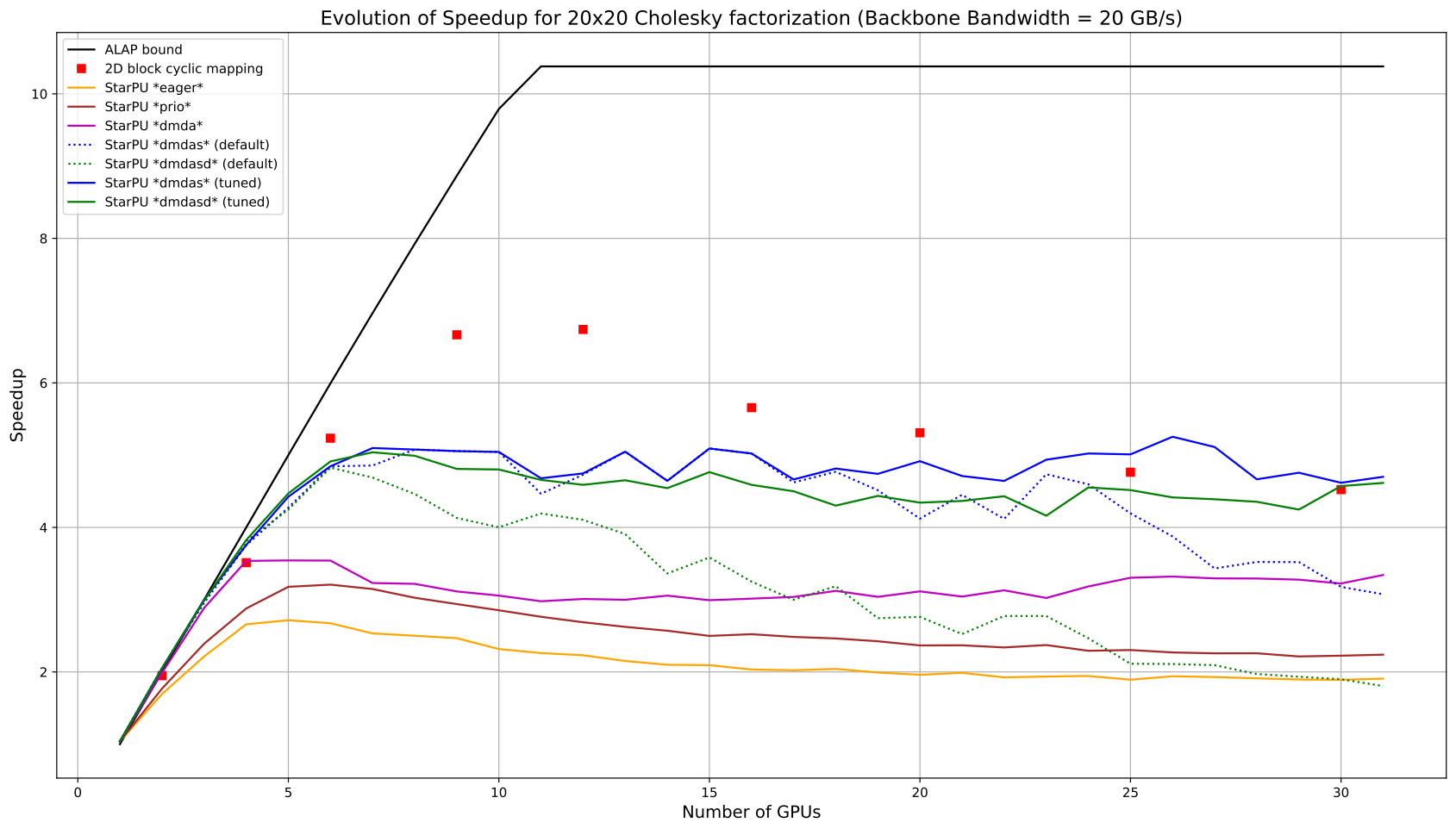


- The bound works very well in all cases
- ALAP is always very good, ASAP is very good too in the GPU Case

SimGrid simulations



SimGrid simulations



Conclusions

- Lower bound on the completion time for
  - Cholesky factorization
  - on P homogeneous resources
    - for all possible values of C, T, S, G and N
    - communications are fully overlapped by computations

Conclusions

- Lower bound on the completion time for
  - Cholesky factorization
  - on P homogeneous resources
    - for all possible values of C, T, S, G and N
    - communications are fully overlapped by computations
- The bound is based on a detailed analysis of
  - ALAP schedule
  - Cholesky DAG

Conclusions

- Lower bound on the completion time for
  - Cholesky factorization
  - on P homogeneous resources
    - for all possible values of C, T, S, G and N
    - communications are fully overlapped by computations
- The bound is based on a detailed analysis of
  - ALAP schedule
  - Cholesky DAG
- It shows that
  - ALAP is very good heuristic (CPU or GPU)
  - the bound is very tight
  - ASAP is a very good heuristic in the GPU case.
- In practice, communications are not negligible
  - big gap between the lower bound and SimGrid executions
  - thus lower bound could be further adjusted

Open Questions and perspectives

1. Prove an approximation factor for ALAP
    - not easy, we tried without success
    - Rem: ALAP is not always optimal

Open Questions and perspectives

1. Prove an approximation factor for ALAP
    - not easy, we tried without success
    - Rem: ALAP is not always optimal
2. Extend and automate this technique to other graphs
    - for the moment it is too much handmade, too hard and
    - too long to do it on other graphs.
    - for example, we could attack compressed versions with BLR (heterogeneous tiles)

Open Questions and perspectives

1. Prove an approximation factor for ALAP
   - not easy, we tried without success
   - Rem: ALAP is not always optimal
2. Extend and automate this technique to other graphs
   - for the moment it is too much handmade, too hard and
   - too long to do it on other graphs.
   - for example, we could attack compressed versions with BLR (heterogeneous tiles)
3. Extend these techniques to heterogeneous settings
   - k CPU and k' GPU
   - not obvious to generalize critical path in a heterogeneous context.