Online Scheduling of Task Graphs on Hybrid Platforms

Louis-Claude Canon Loris Marchal Bertrand Simon
Frédéric Vivien

Univ Lyon, CNRS, ENS de Lyon, Inria, Université Claude-Bernard Lyon 1, LIP UMR5668,
F-69342, LYON Cedex 07, France.

MAO – June 7th, 2019

Breaking down the title

Hybrid Platforms

- Many CPUs + few accelerators (GPUs, Xeon Phis, . . .)

Task Graphs (DAGs)

- Used in runtime schedulers (StarPU, OmpSs, XKaapi, . . .)

Online Scheduling

- Unknown graph
  - tasks not submitted yet
  - depends on results
- Advantages vs offline
  - quicker decisions
  - robust to inaccuracies
- *Semi-online: partial information, e.g., bottom-levels (≈ critical path)*

Main challenge: take binary decisions without knowing the future

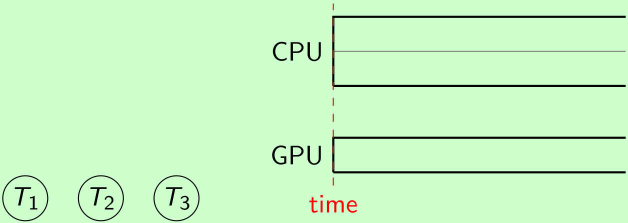
Model and toy example

Model

- m CPUs $\geq$ k GPUs
- Graph of tasks $T_i : \left\{ \overline{p_i} = \text{ CPU time };\ \underline{p_i} = \text{ GPU time} \right\}$
- Online: only available tasks are known

Objective: minimize makespan

Example (2 CPUs, 1 GPU)

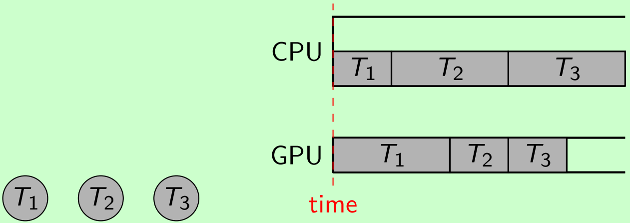


Model

- m CPUs $\geq$ k GPUs
- Graph of tasks $T_i : \left\{ \overline{p_i} = \text{ CPU time } ; \ \underline{p_i} = \text{ GPU time} \right\}$
- Online: only available tasks are known

Objective: minimize makespan

Example (2 CPUs, 1 GPU)



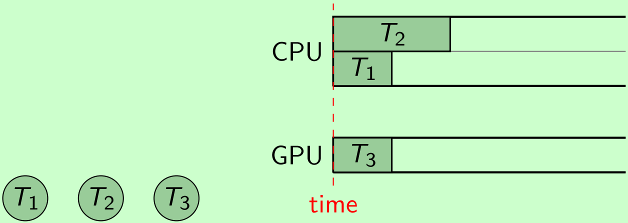
Model and toy example

Model

- m CPUs $\geq$ k GPUs
- Graph of tasks $T_i : \left\{ \overline{p_i} = \text{ CPU time } ; \ \underline{p_i} = \text{ GPU time} \right\}$
- Online: only available tasks are known

Objective: minimize makespan

Example (2 CPUs, 1 GPU)



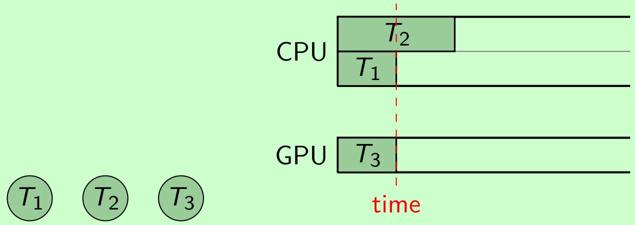
Model and toy example

Model

- m CPUs $\geq$ k GPUs
- Graph of tasks $T_i : \left\{ \overline{p_i} = \text{ CPU time } ; \ \underline{p_i} = \text{ GPU time} \right\}$
- Online: only available tasks are known

Objective: minimize makespan

Example (2 CPUs, 1 GPU)



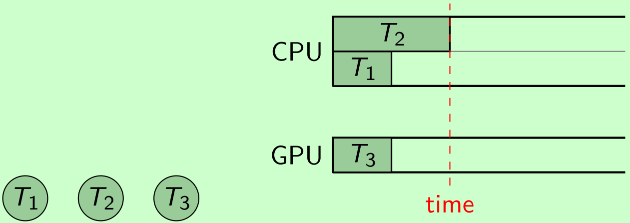
Model and toy example

Model

- m CPUs $\geq$ k GPUs
- Graph of tasks $T_i : \left\{ \overline{p_i} = \text{ CPU time } ; \ \underline{p_i} = \text{ GPU time} \right\}$
- Online: only available tasks are known

Objective: minimize makespan

Example (2 CPUs, 1 GPU)

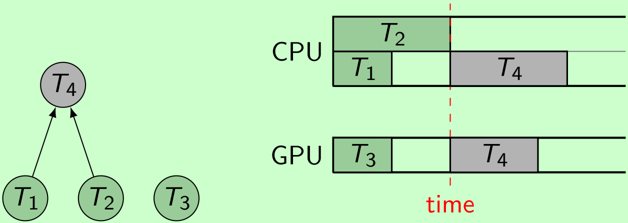


Model

- m CPUs $\geq k$ GPUs
- Graph of tasks $T_i : \left\{ \overline{p_i} = \text{ CPU time ; } \underline{p_i} = \text{ GPU time} \right\}$
- Online: only available tasks are known

Objective: minimize makespan

Example (2 CPUs, 1 GPU)

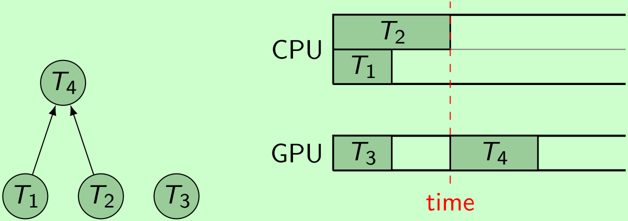


Model

- m CPUs $\geq$ k GPUs
- Graph of tasks $T_i : \left\{ \overline{p_i} = \text{ CPU time ; } \underline{p_i} = \text{ GPU time} \right\}$
- Online: only available tasks are known

Objective: minimize makespan

Example (2 CPUs, 1 GPU)



Related work

m CPUs, k GPUs

Existing offline algorithms (NP-Complete)

- Independent tasks:
  - $\frac{4}{3} + \frac{1}{3k}$ - approx [Bleuse, Kedad-Sidhoum, Monna, Mounié, Trystram 2015]
    Expensive PTAS [Bonifaci, Wiese 2012]
  - Low-complexity: 2 - approx [Canon, Marchal, Vivien 2017]
    3.41 - approx [Beaumont, Eyraud-Dubois, Kumar 2017]
- DAG: 6 - approx (LP rounding) [Kedad-Sidhoum, Monna, Trystram 2015]

Existing online algorithms

- Independent tasks: 4 - competitive [Imreh 2003]
  3.85 - competitive [Chen, Ye, Zhang 2014]
- DAG: $4\sqrt{\frac{m}{k}}$ - compet. ER-LS [Amarís, Lucarelli, Mommessin, Trystram 2017]

Outline

m CPUs, k GPUs

1. **Lower bounds on online algorithms**
   - No online algorithm can be $< \sqrt{m/k}$ - competitive
2. **Propose improvements of ER-LS**
   - Competitive ratio
   - Average performance
   - Validation on simulations

Lower bound

m CPUs, k GPUs

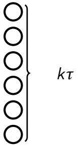
Theorem

No online algorithm $\mathscr{A}$ is $< \sqrt{m/k}$-competitive for any m, k.

Proof (where $\tau = \sqrt{m/k} = 3$): graph built in $n\tau$ phases.

Phase 1 - $k\tau$ independent tasks $\left\{\overline{p_i} = \tau \ ; \ \underline{p_i} = 1\right\}$: $\mathscr{A}$ needs a time τ

Graph with
$k = 2$, $n = 3$



$\square$

Lower bound

m CPUs, *k* GPUs

Theorem

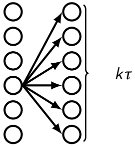
No online algorithm $\mathscr{A}$ *is* $< \sqrt{m/k}$*-competitive for any* m, k.

Proof (where $\tau = \sqrt{m/k} = 3$): graph built in $n\tau$ phases.

Phase 1 - $k\tau$ independent tasks $\left\{\overline{p_i} = \tau \ ; \ \underline{p_i} = 1\right\}$: $\mathscr{A}$ needs a time τ

Phase 2 - same as phase 1, but are successors of the last task

Graph with
$k = 2$, $n = 3$



□

Lower bound

m CPUs, k GPUs

Theorem

No online algorithm $\mathscr{A}$ is $< \sqrt{m/k}$ -competitive for any m, k.

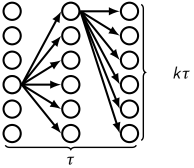
Proof (where $\tau = \sqrt{m/k} = 3$): graph built in $n\tau$ phases.

Phase 1 - $k\tau$ independent tasks $\left\{\overline{p_i} = \tau \ ; \ \underline{p_i} = 1\right\}$: $\mathscr{A}$ needs a time τ

Phase 2 - same as phase 1, but are successors of the last task

Phase 3 - same as phase 2, but are successors of the last task

Graph with
$k = 2$, $n = 3$



$\square$

Lower bound

m CPUs, k GPUs

Theorem

No online algorithm $\mathscr{A}$ is $< \sqrt{m/k}$ - competitive for any m, k.

Proof (where $\tau = \sqrt{m/k} = 3$): graph built in $n\tau$ phases.

Phase 1 - $k\tau$ independent tasks $\left\{\overline{p_i} = \tau \ ; \ \underline{p_i} = 1\right\}$: $\mathscr{A}$ needs a time τ

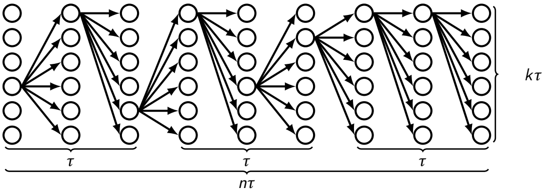
Phase 2 - same as phase 1, but are successors of the last task

Phase 3 - same as phase 2, but are successors of the last task

Phase x - ...

$\Longrightarrow$ Makespan obtained by $\mathscr{A}$: $n\tau^2$

Graph with
$k = 2$, $n = 3$



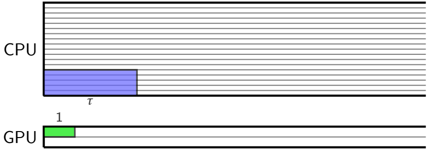
Lower bound

m CPUs, k GPUs

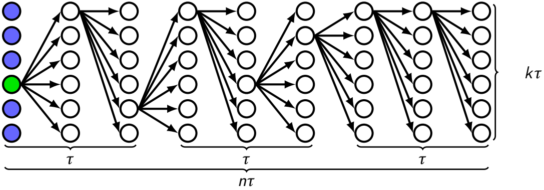
Theorem

No online algorithm $\mathscr{A}$ is $< \sqrt{m/k}$ - competitive for any m, k.

Proof (where $\tau = \sqrt{m/k} = 3$): graph built in $n\tau$ phases.



Graph with
$k = 2,\ n = 3$



□

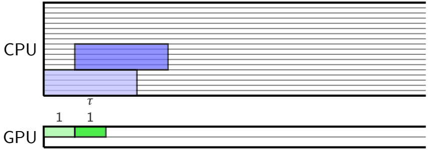
Lower bound

m CPUs, *k* GPUs

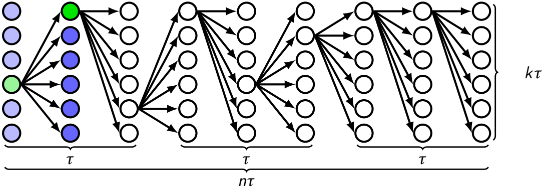
Theorem

No online algorithm $\mathscr{A}$ *is* $< \sqrt{m/k}$*-competitive for any* m, k.

Proof (where $\tau = \sqrt{m/k} = 3$): graph built in $n\tau$ phases.



Graph with
$k = 2$, $n = 3$



□

m CPUs, k GPUs

Theorem

No online algorithm $\mathscr{A}$ is $< \sqrt{m/k}$ - competitive for any m, k.

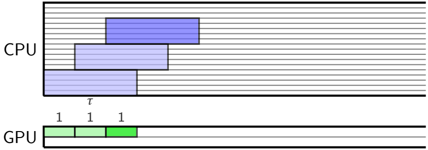
Proof (where $\tau = \sqrt{m/k} = 3$): graph built in $n\tau$ phases.

τ phases

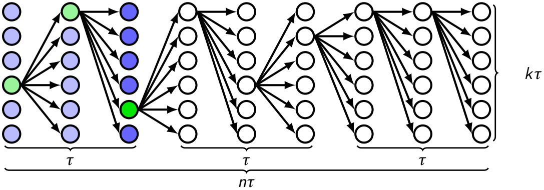
OPT $\Rightarrow 2\tau$

$\mathscr{A} \Rightarrow \tau^2$

Lower bound: $\frac{1}{2}\tau$



Graph with
$k = 2$, $n = 3$



Lower bound

m CPUs, k GPUs

Theorem

No online algorithm $\mathscr{A}$ is $< \sqrt{m/k}$-competitive for any m, k.

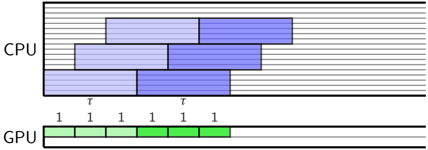
Proof (where $\tau = \sqrt{m/k} = 3$): graph built in $n\tau$ phases.

2τ phases

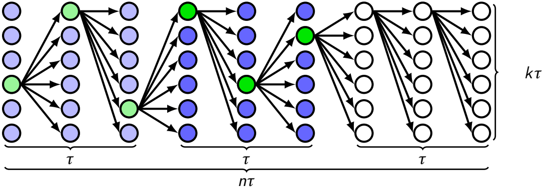
OPT $\Rightarrow 3\tau$

$\mathscr{A} \Rightarrow 2\tau^2$

Lower bound: $\frac{2}{3}\tau$



Graph with
$k = 2$, $n = 3$



□

Lower bound

m CPUs, k GPUs

Theorem

No online algorithm $\mathscr{A}$ is $< \sqrt{m/k}$-competitive for any m, k.

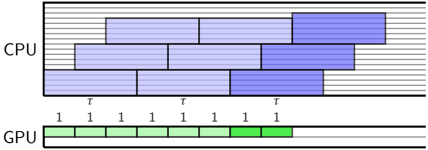
Proof (where $\tau = \sqrt{m/k} = 3$): graph built in $n\tau$ phases.

$n\tau$ phases

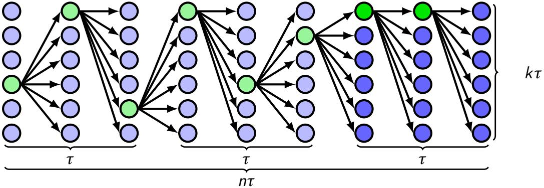
OPT $\Rightarrow (n+1)\tau$

$\mathscr{A} \Rightarrow n\tau^2$

Lower bound: $\frac{n}{n+1}\tau$



Graph with
$k = 2$, $n = 3$



□

Generalized lower bounds

Recall previous lower bound: $\sqrt{m/k}$, for m CPUs, k GPUs

Precomputed information

- Bottom-level (≈ remaining critical path) does not help
- All descendants: non-constant LB $= \Omega\left((m/k)^{1/4}\right)$

Powerful scheduler

- *Kill* + *migrate* does not help
- *Preempt* + *migrate* hardly helps

Note: allocation is difficult

- How to choose which tasks to speed-up?
- Fixed allocation: 3 - competitiveness

ER-LS algorithm ($4\sqrt{m/k}$-competitive, [Amarís et al.])

m CPUs, k GPUs

Main concept

- Pick any available task T_i
- Allocate T_i to CPUs or GPUs
- Schedule it as soon as possible

Where to allocate an available task T_i

If T_i can be executed on GPU before time $\overline{p_i}$:

- put T_i on GPU

Otherwise:

- if $\frac{\overline{p_i}}{\underline{p_i}} \leq \sqrt{\frac{m}{k}}$: put it on CPU
- else : put it on GPU

Our proposition: QA (Quick Allocation) algorithm

m CPUs, k GPUs

Main concept

- Pick any available task T_i
- Allocate T_i to CPUs or GPUs
- Schedule it as soon as possible

Where to allocate an available task T_i

~~If T_i can be executed on GPU before time $\overline{p_i}$:~~

- ~~put T_i on GPU~~

~~Otherwise:~~

- if $\frac{\overline{p_i}}{\underline{p_i}} \leq \sqrt{\frac{m}{k}}$: put it on CPU
- else : put it on GPU

Our proposition: QA (Quick Allocation) algorithm

m CPUs, k GPUs

Main concept

- Pick any available task T_i
- Allocate T_i to CPUs or GPUs
- Schedule it as soon as possible

Where to allocate an available task T_i

~~If T_i can be executed on GPU before time $\overline{p_i}$:~~

- ~~put T_i on GPU~~

~~Otherwise:~~

- if $\frac{\overline{p_i}}{\underline{p_i}} \leq \sqrt{\frac{m}{k}}$: put it on CPU
- else : put it on GPU

Theorem

QA is $2\sqrt{m/k}+1$ - competitive. This ratio is (almost) tight.

What about *easy* cases?

m CPUs, k GPUs

Problem with QA

- Expect the worse: aim at $\Theta(\sqrt{m/k})$-competitiveness
- ☹ Poor performance on *easy* graphs

Well-known EFT algorithm (Earliest Finish Time)

- `Terminate each` T_i `as soon as possible;`
- ☺ Greedy version, works great on non-pathological cases
- ☹ Can be really bad: $\geq (\frac{m}{k} + 2)$ OPT

Can we have both benefits? MIXEFT

- `Run EFT and simulate QA;`
  `When EFT is` λ `times worse than QA: switch to QA;`
- Tunable: $\lambda = 0 \rightarrow$ QA ; $\lambda = \infty \rightarrow$ EFT
- $(\lambda + 1)(2\sqrt{m/k} + 1)$-competitive — conjectured $\max(\lambda, 2\sqrt{m/k} + 1)$
- Same idea as ER-LS but pushed to the extreme

Simulations

m CPUs, k GPUs

Heuristics (makespan normalized by offline HEFT's)

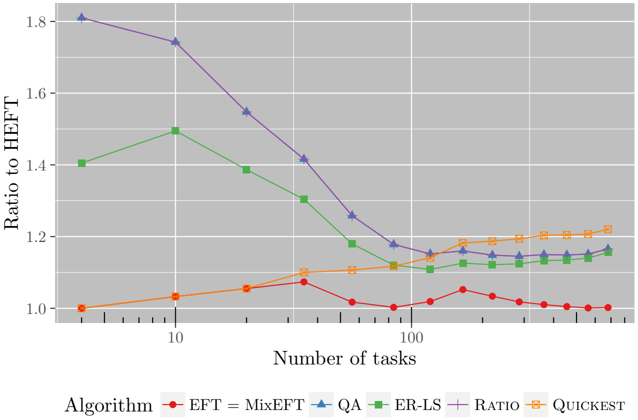
- EFT (= MIXEFT as EFT better than QA here)
- QA (switch at $\sqrt{m/k}$)
- ER-LS (= QA + greedy rule: slightly more tasks on GPUs)
- QUICKEST (= QA with switch at 1: more tasks on GPUs)
- RATIO (= QA with switch at m/k: more tasks on CPUs)

Datasets for $m = 20$ CPUs and $k = 2$ GPUs

Cholesky	4 types of tasks
Synthetic	STG set, 300 tasks, random GPU acceleration ($\mu = \sigma = 15$)
Ad-hoc	one chain & independent tasks

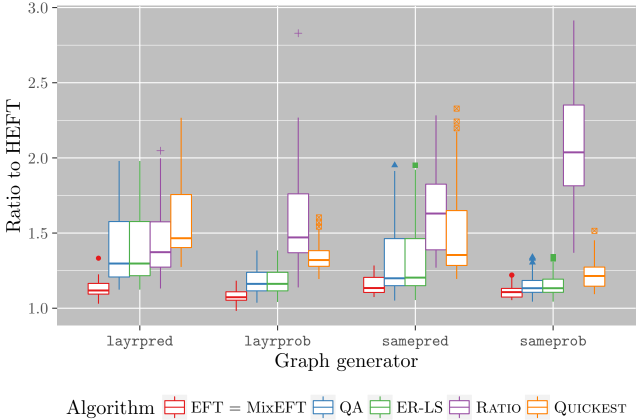
Results for Cholesky graphs (lower is better)

m CPUs, *k* GPUs

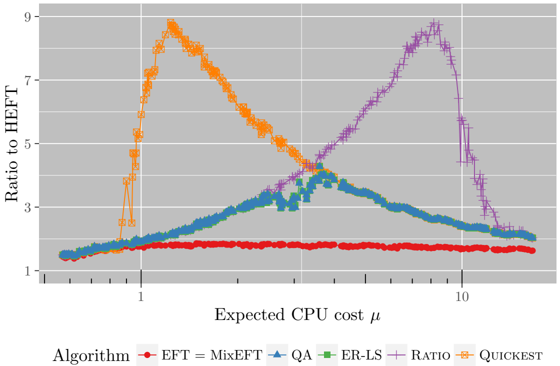


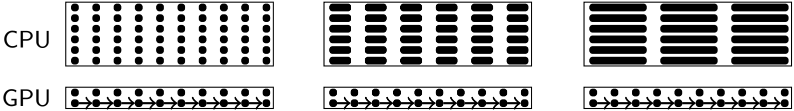
$$\frac{m}{k} = 10 \qquad \sqrt{\frac{m}{k}} \approx 3.3 \qquad \frac{\text{CPU time}}{\text{GPU time}} \in \{28, 26, 11, \underbrace{2}_{\text{POTRF}}\}$$

Results for synthetic graphs (lower is better)



Results for 300-tasks ad-hoc graphs (lower is better)





OPT (left, $\overline{p_i} \approx \underline{p_i}$) OPT (middle) OPT (right, $\overline{p_i} \approx \frac{m}{k}\underline{p_i}$)

Conclusion

m CPUs, k GPUs

Summary

- No online algo. is $< \sqrt{m/k}$ - competitive
  Additional knowledge or power hardly helps
- QA: $(2\sqrt{m/k}+1)$ - competitive
  MIXEFT: compromise effectiveness / guarantees
- Extended to multiple types of processors (not in this talk)

Perspectives

- Low-cost offline algorithm with constant ratio
- Communication times
- Parallel tasks