Data Structures for Stochastic Scheduling with Applications in GPGPUs

Jakub Mareček

Starting August 7th, 2012
IBM Research, Damastown Industrial Estate, Dublin 15, Ireland

August 2, 2012

In this talk

Introduction:

- ARM and GPGPUs
- Stochastic task-graph scheduling

The Meat:

- Priority queues for scheduling along the critical path
- Interval trees for scheduling with multiple resources
- The combination of the two and extensions

Acknowledgements

Motivation I: ARM and GPGPUs

ARM

- A processor design company
- A very successful company (FTSE 100, + 440% in 5yrs)
- In Q2 2012, 2B chips were manufactured using ARM IP

General-purpose computing on graphics processing units (GPUs)

- In 2011, Mali became the most widely-licensed GPU architecture
- Mali-T658: "desktop-class performance" (8×512 jobs)

Joint work

- Six months on-site in Cambridge (2009, 2010)
- Scheduling in OpenCL drivers for Mali GPGPUs, ...

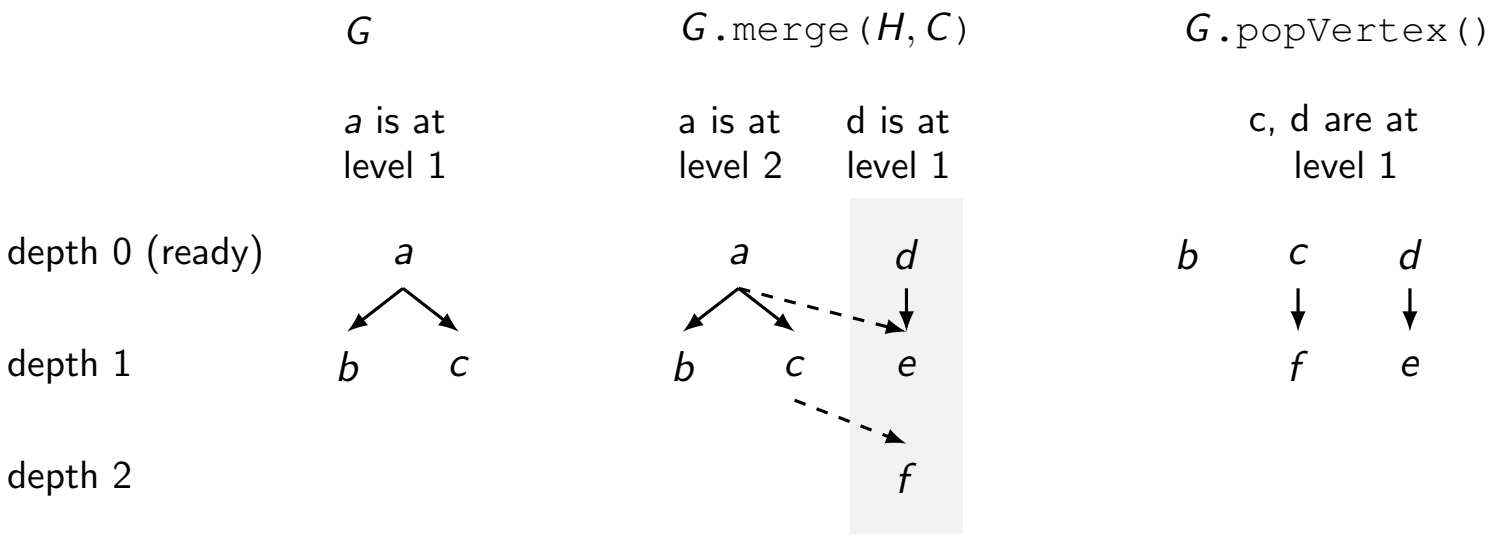
Motivation II: OpenCL

- Numerous standards for task- and data- parallel computing
- OpenCL is an open standard developed by ARM, AMD/ATI, IBM, Intel, ..., and nVidia
- A brain-child of Apple engineers, who dislike nVidia CUDA
- In both CUDA and OpenCL, one can specify data dependencies between "jobs":

```
memobjs[0] = clCreateBuffer(c, ...);
p = clCreateProgramWithSource(c, 1, src, ...);
clBuildProgram(p, ...);
k = clCreateKernel(p, "name", ...);
clSetKernelArg(k, ..., (void *)&memobjs[0]);
event = clEnqueueNDRangeKernel(queue, k, ...);
clFlush(queue);
```

Motivation III: Scheduling in GPGPUs

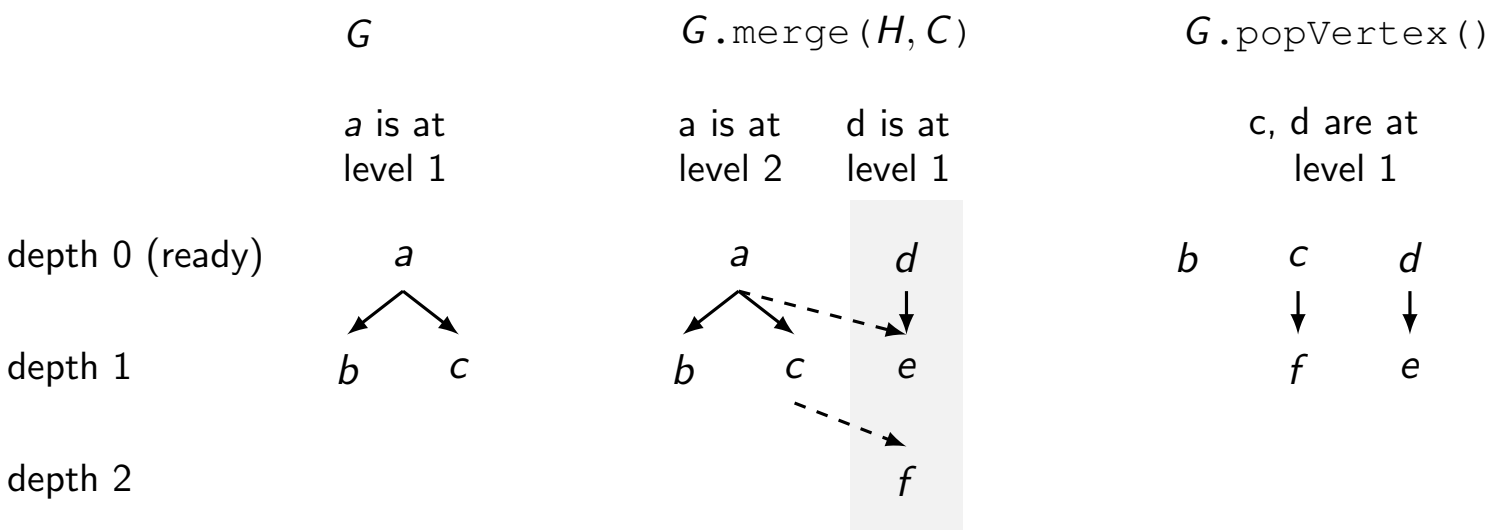
- Jobs are enqueued in batches, we processes the queue
- A DAG capturing data dependencies ("taskgraph")
- Implicit structure in the "taskgraph"
- Reliable resource utilisation estimates



Motivation III: Scheduling in GPGPUs

"Critical path" policies need to maintain longest paths in a DAG:

- `popVertex`: return a source on (one of) the longest path(s) and delete it together with all out-going edges
- `merge(`H, C`)`: add vertex-disjoint DAG H to the current DAG G together with $|C|$ "cross" edges from G to H



Stochastic Scheduling with Precedencies I

- n enqueued so far, m jobs scheduled already
- $G[m, n]$ is the subgraph induced by vertices $\{m, m+1, \ldots, n\}$ corresponding to $n - m$ jobs in the queue
- d_{mn} the length of longest path in $G[m, n]$ and $\beta_n = \mathbb{E}[d_{1n}]/n$

Theorem (Attributed to Bruce Hajek by Tsitsiklis et al. (1986))

The limit $\lim_{n\to\infty}(d_{mn}/n)$ *exists almost surely. Let us use* β^* *to denote the limit for any m where it exists. A system with infinitely many servers is stable if the arrival rate* $\lambda < 1/\beta^*$.

Stochastic Scheduling with Precedencies II

Theorem (Papadimitriou and Tsitsiklis (SICOMP, 1987))

Processing the job with the highest level ("along the critical path") first, as soon as any machine is available, is asymptotically optimal with respect to weighted throughput, under certain conditions (*), among non-anticipative non-delay non-preemptive policies and non-anticipative non-delay preemptive policies with zero cost of preemption.

The conditions (*) are spelled out in the paper. Essentially:

- there are jobs with independent identically distributed processing times, drawn from a common binomial or Poisson distribution
- data dependency graph G is a forest of in-trees.

Data Structures for the Critical Path I

The approach:

- Dynamic data structures for maintaining both the length of the longest in-coming path (depth) and the length of the longest out-going path (level) in each node of a DAG
- Dijkstra's "contraption under gravity" (1960s):
  1. traverse the out-going subgraph ("down") from a vertex along a topological order of vertices
  2. traverse the in-coming subgraph in the reverse direction ("up") along a topological order of vertices, updating the level as long as it is necessary
- The non-trivial part is the maintenance of the topological order of vertices using priority queues.

Data Structures for the Critical Path II

The pieces of the puzle:

- The maintanence of the topological order using priority queues
  Katriel, Michel, Van Hentenryck, Constraints, 2005
- Strict Fibonacci heaps
  Brodal, Lagogiannis, Tarjan, STOC, 2012
- An implementation of strict Fibonacci heaps
  Sivr and JM, 2012
- Input-output model of analysis
  Ramalingam and Reps, TCS, 1996

The overall picture:

- Pseudocode and analyses
  JM et al, MISTA, 2011
- Empirical results etc.
  Sivr, JM, 2012

Data Structures for the Critical Path III

Table: Upper bounds on the run-time of key operations.

Operation	Linked lists	Our data structure
`merge`(H, C)	$O(\|C\|n)$	$O(\|C\| \log \|C\| + \|\delta\| Q(\|\delta\|) + \|\|\delta\|\|)$
`popVertex`	$O(n^2)$	$O(\|\delta\| Q(\|\delta\|) + \|\|\delta\|\|)$
"What next" query	$O(n^2)$	$O(1)$
Vertex insertion	$O(1)$	$O(1)$
Edge insertion	$O(n)$	$O(\|\delta\| Q(\|\delta\|) + \|\|\delta\|\|)$
Vertex deletion	$O(n)$	$O(\|\delta\| Q(\|\delta\|) + \|\|\delta\|\|)$

Note n is the number of vertices, $|\delta|$ is the number of vertices on the affected longest paths, and $||\delta||$ is the number of vertices in the neighbourhoods of the affected vertices. $Q(n)$ is the complexity of insertion and extraction of an element from a priority queue with n elements.

Open Problems I

Open problems in lower bounds:

- In prioritising vertices of a DAG, where each vertex is assigned a priority such that, for each oriented edge (v, w), $\text{priority}(v) < \text{priority}(w)$, there is a lower bound of $\Omega(n \log n)$ on the insertion of m edges in a graph on n vertices. Does it apply when one req. arbitrary edge addition?

  Ramalingam and Reps, TCS, 1996

- Does the arbitrary edge and vertex addition make a difference? Can one use some "lazy" approach with the "queue like" modifications?

Major open problems in priority queues:

- an amortised analysis of strict Fibonacci heaps
- multi-threaded strict Fibonacci heaps with delayed self-reorganisation.

Open Problems II

Major open problems in stochastic scheduling:

- Can the conditions of Papadimitriou and Tsitsiklis (on processing times, task-graph structure) be relaxed so that the asymptotic optimality is maintained?

  Papadimitriou, Tsitsiklis, SICOMP, 1987

- Can one extend the stability results of Hajek to a finite number of servers?

  Tsitsiklis, Papadimitriou, Humblet, Pierre, JACM, 1986

- For certain closed queuing systems, the control is EXP-Complete.
  Could the control of certain open queuing systems be k-EXP-Complete?

  Papadimitriou and Tsitsiklis, Math. Oper. Res., 1999

  JM, 2010

Open Problems III

Further problems in stochastic scheduling:

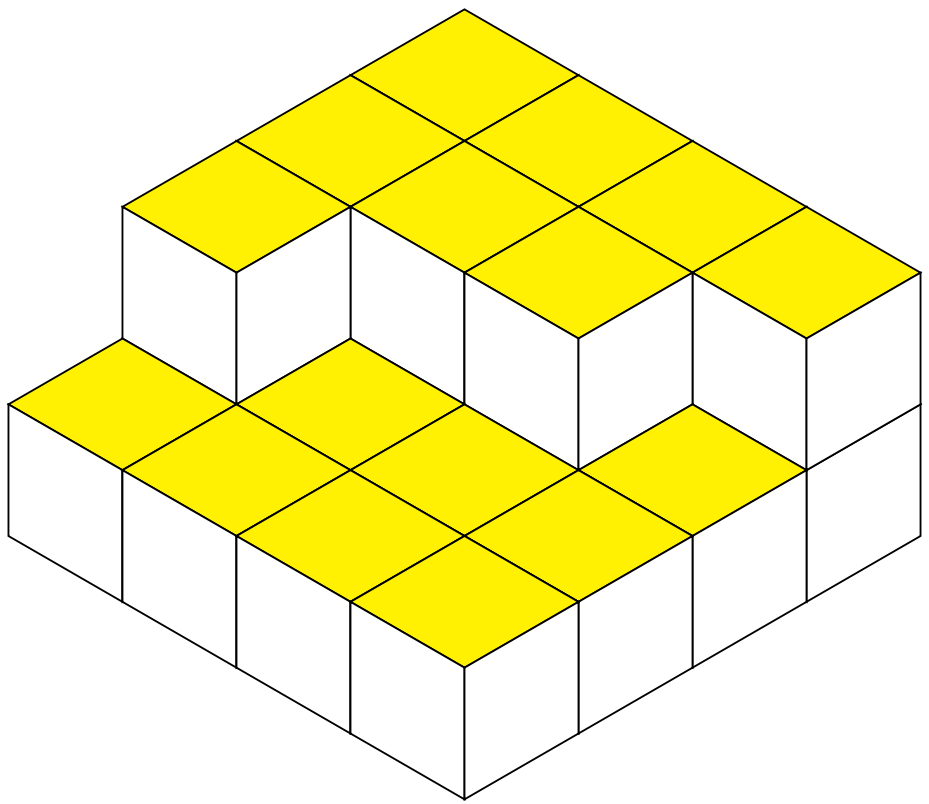
- Stability in realistic models of queuing networks with depedencies, where there are jobs partitioned into groups. What if there are no intra-group dependencies and jobs within each share the dependencies? What if the probability of dependence of group a on a group b is inversely proportional to the difference in their release dates $r_a - r_b$?
- What are the benefits of dynamic scheduling policies, e.g. the multi-mode multi-armed bandits?

  Niño-Mora, ValueTools 2008
- Can we exploit structure in the taksgraph? Scheduling along the critical path is not optimal, when we can make educated guesses about the future changes of the taskgraph.

  Chekuri, Johnson,Motwani, et al., MICRO, 1996

Packing Boxes into a Larger Box

- A "box": orthogonal parallelepiped in dimension d
- Large box with $d-1$ dimensions known ($d-1$ resources)
- Smaller non-rotatable boxes (jobs)
- "Strip packing" smaller boxes into the larger box



Packing Boxes into a Larger Box

- A "box": orthogonal parallelepiped in dimension d
- Large box with $d-1$ dimensions known ($d-1$ resources)
- Smaller non-rotatable boxes (jobs)
- "Strip packing" smaller boxes into the larger box

On-line and Stochastic:

- Rules for packing the extra box
  (e.g. deepest-bottom-left, max-contact)

  Burke, Kendall, Whitwell, Oper. Res., 2004

Off-line:

- Dynamic programming

  Martello, Pisinger, Vigo, Oper. Res., 2000
- Local search sequencing boxes and simple rules

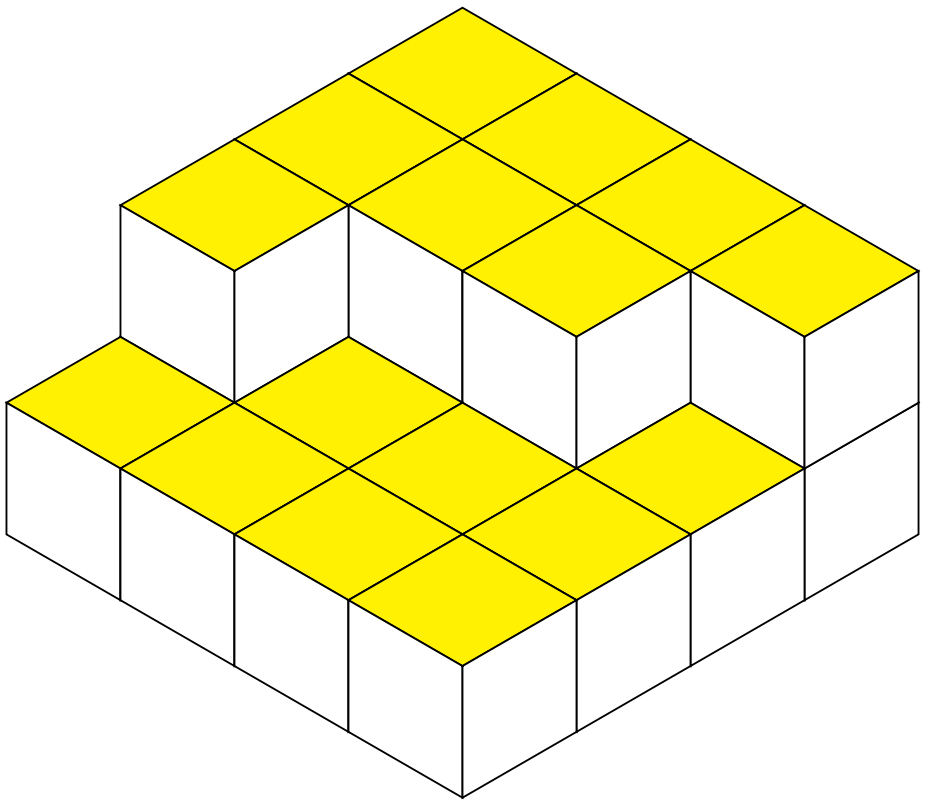
  Allen, Burke, Kendall, EJOR, 2011
- Math. programming uncompetitive until recently

  Padberg, Math. Meth. Oper. Res, 2000

  Allen, Burke, JM, OR Letters, 2012

Data Structures for the Skyline

Implementations of rules for packing an additional box maintain the "skyline", i.e. the $(d-1)$D projection of the top-most boxes:



Data Structures for the Skyline

Implementations of rules for packing an additional box maintain the "skyline", i.e. the $(d-1)$D projection of the top-most boxes:

- `lowestGaps`: return a list of gaps at the lowest position in the skyline
- `neighbouringGaps`(G): return gaps touching G that are at a higher position
- `splitGap`(B, G): split gap G by packing box B in
- `changeHeight`(G, v): update the height of the skyline after packing a box
- `isContained`(B, G): does box B fit gap G

Data Structures for the Skyline I

The approach of Allen and Burke (2011):

- Maintain the skyline as an ordered list L of $(d-1)D$ boxes and their heights, sorted by the heights
- Store lines of all $(d-1)D$ boxes in an interval tree T interlinked with L
- The implementation of interval trees using red-black and AVL trees is standard

  de Berg, Cheong, van Kreveld, Overmars, Springer, 2008

- `lowestGaps`: lookup in the list L
- `neighbouringGaps`(G): query the interval tree T
- `changeHeight`(G, v): move an element in the list L

Data Structures for the Skyline II

Interval trees are a considerable improvement over axis-aligned bounding-box (AABB) trees, proposed earlier also by Allen et al.

Table: Upper bounds on the run-time of key operations.

Operation	AABB Tree	Interval Trees
`lowestGaps`	$O(\log(nd) + k)$	$O(\log(n) + k)$
`isContained`	$O(nd)$	$O(df)$
`splitGap`	$O(nd^2)$	$O(f^2 d^3 2^d)$
`neighbouringGaps`	$O(nd)$	$O(df \log(f) + k)$
`changeHeight`	$O(d \log(nd))$	$O(f^2 d^3 2^d)$

n is the number of boxes, d is the dimension, k is the size of the output, f is the number of facets defining the gap

Data Structures for Taskgraph Scheduling with $d-1$ Resources

- One should like to combine all of the above
- An issue: d-criteria make even shortest paths hard
  Papadimitriou, FOCS 2000
- An idea: convexify the d-criteria using d-weights, where the d-weights are obtained by recoding utilisation
- Work in progress

The references:

- Maintain the skyline using axis-aligned bounding-box trees
  Allen, Burke, Kendall, EJOR, 2011
- Maintain the skyline using interval trees
  Allen, Burke, INFORMS JOC, 2011
- The d-criteria critical path and the combined data structure
  Kulovaný and JM, 2012

More Open Problems I

Open problems:

- Are there non-trivial bounds on the numbers of "gaps"?
- What is the performance of the "simple rules" (on-line, in the stochastic model)?
- Can the work on 1D reordering buffers be generalised to dD?

  Adamaszek et al., STOC, 2011
- Is the complexity of packing in the stochastic model still EXP-Complete?

  Papadimitriou and Tsitsiklis, Math. Oper. Res., 1999
- Can the work of Okounkov on random plane partitions be used to describe the performance of randomised algorithms for packing?

More Open Problems II

- What if we allow for further, "divisible" resources?

  Feldmann, Kao, Sgall, Teng, J. Combin. Optim, 1998
- What if we allow for restarts?
- What if we allow for a small number of distinct sequence-dependent set-up times?
- What if there is no open dimension, but we allow for multiple, progressively more conservative estimates of each dimension ("levels of criticality") and want to maximise the minimum "criticality" across boxes packed in?

  Baruah et al., IEEE Trans. Computers, 2012

Acknowledgements

At ARM:

- Hedley Francis (GPGPU technical lead)
- Anton Lokhmotov (head of a team, GPGPU drivers)
- Robert Elliot (implemented the actual scheduler)

At the University of Nottingham:

- Edmund K. Burke (former Dean of Science)
- Andrew J. Parkes (my other supervisor)
- Sam D. Allen (work on n-D packing)

At Czech Technical University:

- Jirka Kulovaný (masters dissertation)
- Vojta Sivr (masters dissertation)

Conclusions

- The importance of Computer Science in scheduling:
  Even "simple" policies require non-trivial implementation
- People from the industry may actually read what you write,
  but won't consider it w/o empirical results and pseudocode
- There is much more to be done!

- Any answers, questions, and comments are very welcome!

jakub@marecek.cz

Stochastic Scheduling: Some Definitions I

- Long-run properties of "policies"
- Weighted throughput of the system for policy π:

$$J(\pi) = \limsup_{t \to \infty} \frac{1}{t} \sum_{q \in Q} w(q) \mathbb{E}[a_q^\pi(t)]$$

where a_q is the number of jobs from queue q completed by time t, and $C_i(q)$ is the completion time of ith job in queue q, both of which are well-defined random variables, and $0 < \alpha \leq 1$ is the discount rate.

Stochastic Scheduling: Some Definitions II

- For an objective function f, input σ, and the optimum of f on σ, $OPT_f(\sigma)$, the asymptotic approximation ratio of policy π of is:

$$R_f^\infty(\pi) = \limsup_{n\to\infty} \left\{ \frac{\pi_f(\sigma)}{OPT_f(\sigma)} \mid OPT_f(\sigma) = n \right\}.$$

- Within a family of policies P, policies with approximation ratio:

$$R_f^\infty = \inf_{\pi\in P} R_f^\infty(\pi).$$

are asymptotically optimal.