Compiling Packet Programs to Reconfigurable Switches: Theory and Algorithms

EuroP4 2020

Balázs Vass, Budapest Univ. of Technology and Economics, Budapest, Hungary
Erika Bérczi-Kovács, Eötvös Loránd University, Budapest, Hungary
Costin Raiciu, University Politehnica of Bucharest, Romania
Gábor Rétvári, Budapest Univ. of Technology and Economics, Budapest, Hungary

Introduction: Pipeline Embedding Problem

Reconfigurable Switch Pipelines

- Programming pipelines using a high-level domain-specific language like P4 is increasingly adopted
- Applications booming→
  - dataplane programs
    - grow in complexity
  - new programmable switch ASICs:
    - more dataplane resources
    - more pipeline stages
- →*Algorithmic issues*

Pipeline Embedding Problem

- Dataplane programming: top-down approach
  - required behavior of the network described in a declarative **P4 program**
  - mapped to hardware by a ***P4 compiler***
- The compiler must analyze the P4 program
  - given an abstract model of the hardware:
    - limits of memory space, width, types,
    - # processing stages
    - max. level of concurrency at each stage, ...
  - finds the best encoding such that:

    all constraints are met
  - 'best': min. # stages, min power, etc.
- We call this the **Pipeline Embedding** Problem

Pipeline Embedding

- Stage for Pipeline Embedding set by [NSDI'15]:

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Compiling Packet Programs to Reconfigurable Switches

Lavanya Jose and Lisa Yan, *Stanford University;*
George Varghese, *Microsoft Research;* Nick McKeown, *Stanford University*

https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/jose

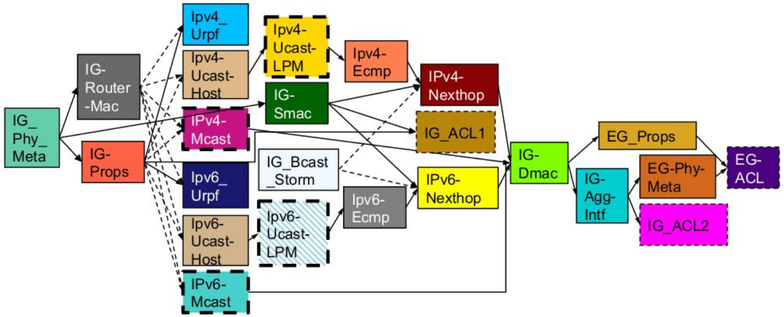
This paper is included in the Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15).
May 4–6, 2015 • Oakland, CA, USA
ISBN 978-1-931971-218

Open Access to the Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15) is sponsored by USENIX

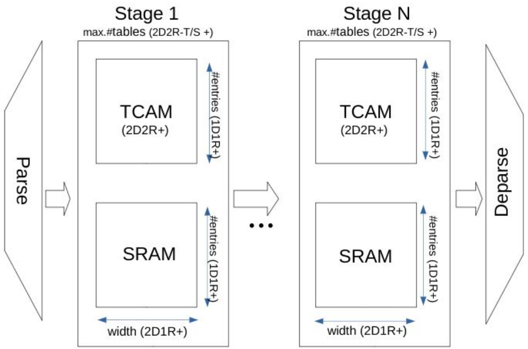
- Proposed:
  - Abstract model for Pipeline Embedding
  - ILP + heuristic algorithms
- Issues:
  - ILP: possibly exponential runtime (*runs for hours for a moderate-sized pipeline)*
  - heuristics: no proven guarantees of 'goodness'
- **Unfolding the algorithmic landscape of Pipeline Embedding was required**

Models of programs and pipelines

- Control-flow dependencies of a P4 **program**
  - represented by a directed acyclic graph (DAG)
  - called Table Dependency Graph (TDG)
  - vertices: logical match-action tables (MATs)
  - arcs: dependencies between the MATs (match, action, etc.)



- Packet processing **pipeline**:
  - modeled as a directed path
  - nodes s_1, s_2,... represent the pipeline stages
  - arcs (s_i, s_{i+1}) encode succession
  - For simplicity:
    - the switch has infinitely many stages,
    - objective: minimize the # stages in the embedding.



Hardware constraints: Simplified models

Full hardware model: very complex→ simplifications →gained some insight→some constraints put back → reanalysed

Model name	**INF-CAP**	**1D1R**	**1D1R-*hsplit***	**2D1R**	**2D2R**	**2D2R-T/S**	**2D2R-PISA**
New feature on top of the previous model	(mapping concurrency due to dependency)	1D capacity/ demands	*hsplit* (table entries split between stages)	2D capacity/ demands	2 kinds of resources per stage	limited number of tables per stage	crossbar constraints, word packing, etc.

- INF-CAP: a directed path of stages, each with infinite capacities (no arc of TDG mapped to just one stage)
- 2D2R-PISA: a full-blown PISA model (RMT described in [NSDI'15])

Results

Results - Complexity

Model name	INF-CAP	1D1R	1D1R-*hsplit*	2D1R	2D2R	2D2R-T/S	2D2R-PISA
New feature on top of the previous model	(mapping concurrency due to dependency)	1D capacity/ demands	*hsplit* (table entries split between stages)	2D capacity/ demands	2 kinds of resources per stage	limited number of tables per stage	crossbar constraints, word packing, etc.
Complexity	P	NPC	NPC	NPC	NPC	NPC strongly NP-hard	NPC strongly NP-hard

- NP complete even with simple capacity constraints (1D1R)
- Hint of proof: some NP-hard problems are apparently special cases

More bad news: Inapproximability

Model name	**INF-CAP**	**1D1R**	**1D1R-*hsplit***	**2D1R**	**2D2R**	**2D2R-T/S**	**2D2R-PISA**
New feature on top of the previous model	(mapping concurrency due to dependency)	1D capacity/ demands	*hsplit* (table entries split between stages)	2D capacity/ demands	2 kinds of resources per stage	limited number of tables per stage	crossbar constraints, word packing, etc.
Bad news: (unless P=NP,) Inapproximable better than . . .	OPT	3/2∗OPT	5/4∗OPT	5/4∗OPT	5/4∗OPT	?	?

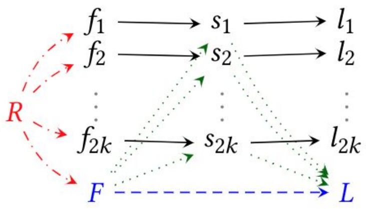
- No Polynomial Time Approximation Scheme (PTAS) exists (no poly. alg. with arbitrary multiplicative error)
- Bird's view of proofs:
  - showing a problem instance family s.t:
  - we can embed each instance in k stages exactly if a related NP-hard problem has a solution
  - otherwise we need (k+1) stages→inapprox. better than(k+1)/k *OPT

- E.g. for 1D1R (oversimplified):
  - no TDG arcs
  - Σ (TDG node sizes) = 2* (stage size)
  - We can embed in k=2 stages exactly if the PARTITION has a solution over the table sizes
  - ...that is NP-hard.

Good news : constant(!)-approximability in quasi-linear time

Model name	INF-CAP	1D1R	1D1R-*hsplit*	2D1R	2D2R	2D2R-T/S	2D2R-PISA
New feature on top of the previous model	(mapping concurrency due to dependency)	1D capacity/ demands	*hsplit* (table entries split between stages)	2D capacity/ demands	2 kinds of resources per stage	limited number of tables per stage	crossbar constraints, word packing, etc.
Good news: Constant-approximable in. . .	OPT	3∗OPT	2∗OPT	3∗OPT	(5 to 8)∗OPT (*)	(6 to 9)∗OPT (*)	?

- Approximation idea: (First Fit by Level and Size)
  - group the TDG nodes by their level
    - node **v** on level **i** if the longest directed path from the root **R** to **v** has a length of **i**
  - nodes in each level can be mapped in the same stage
  - for each level: ~bin packing (without dependency constraints)
    - ...or combinig bin packings



Levels: 0 1 2 3

Conclusion & Future Work

Take-away: Pipeline Embedding is

-NP-hard 😔
-inapproximable in poly. time 😞
(unless P=NP)
-constant-approximable 😌

We will investigate:

- Optimality gaps of Chipmunk, Domino, & others?
- How to write better P4 programs?

Thank you for your attention
Q&A