TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Software- und Multimediatechnik, Lehrstuhl Softwaretechnologie

RACR - MQUAT

The journey has just begun

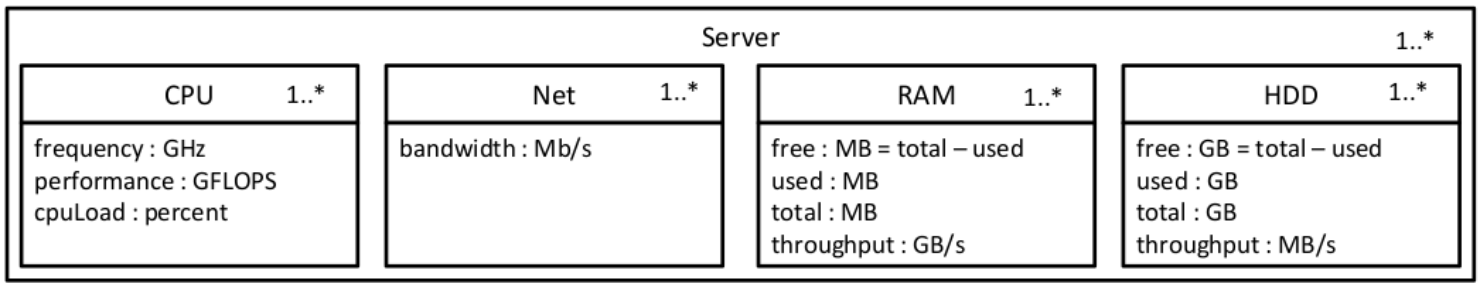
René Schöne

Dresden, September 8, 2015

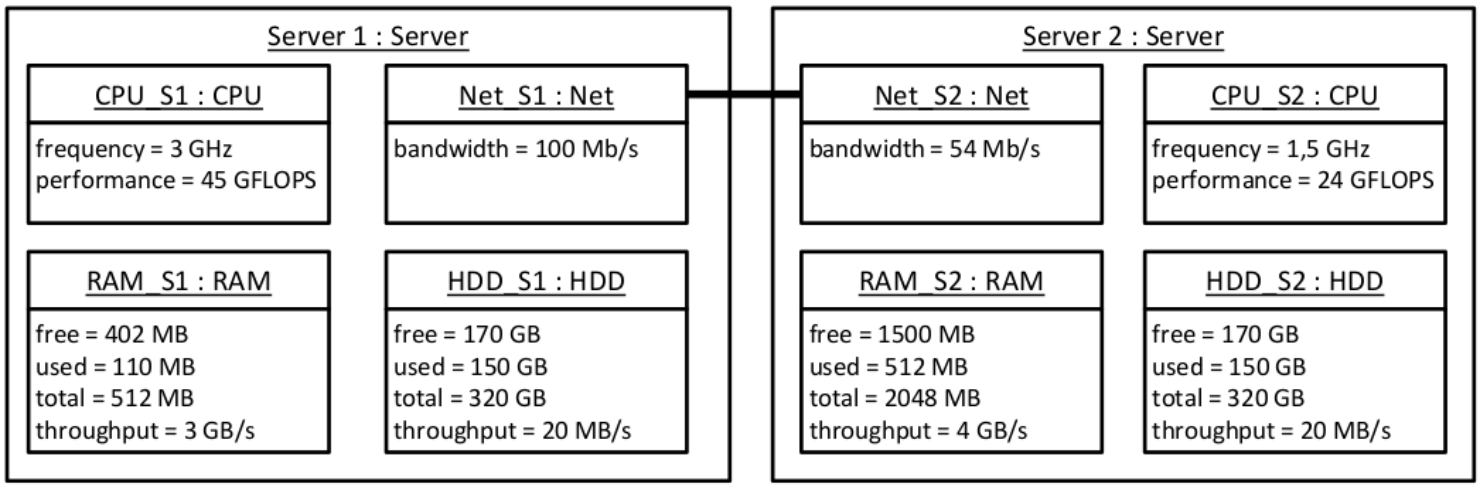
Where we started in Phase I

- MQuAT concept [GKP+14]
  - Self-adaptive system optimizing for multiple qualities
  - Component-based design for both hardware and software
  - Quality contracts capturing requirements and guarantees of components
- THEATRE [GWC+10] as a Java-based implementation of MQuAT
  - Knowledge represented using EMF-(Meta)Models
  - Optimization problem solved by transformation to ILP
  - Designed for distributed operation (see HAECubie) using Master-Slave-Pattern [Sah96]

Structure and variant model

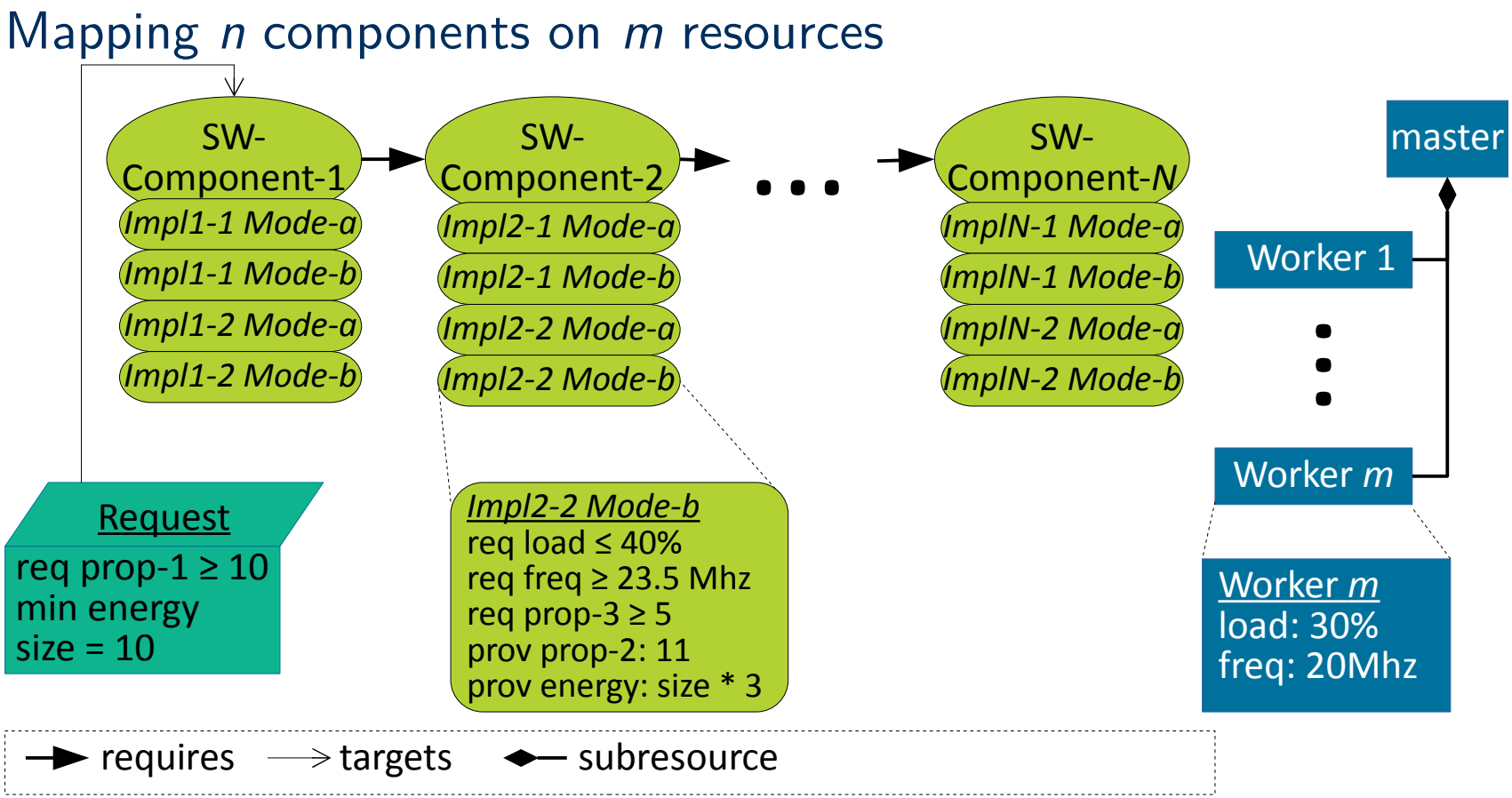


(a) CCM Structure Model for Hardware Landscapes.

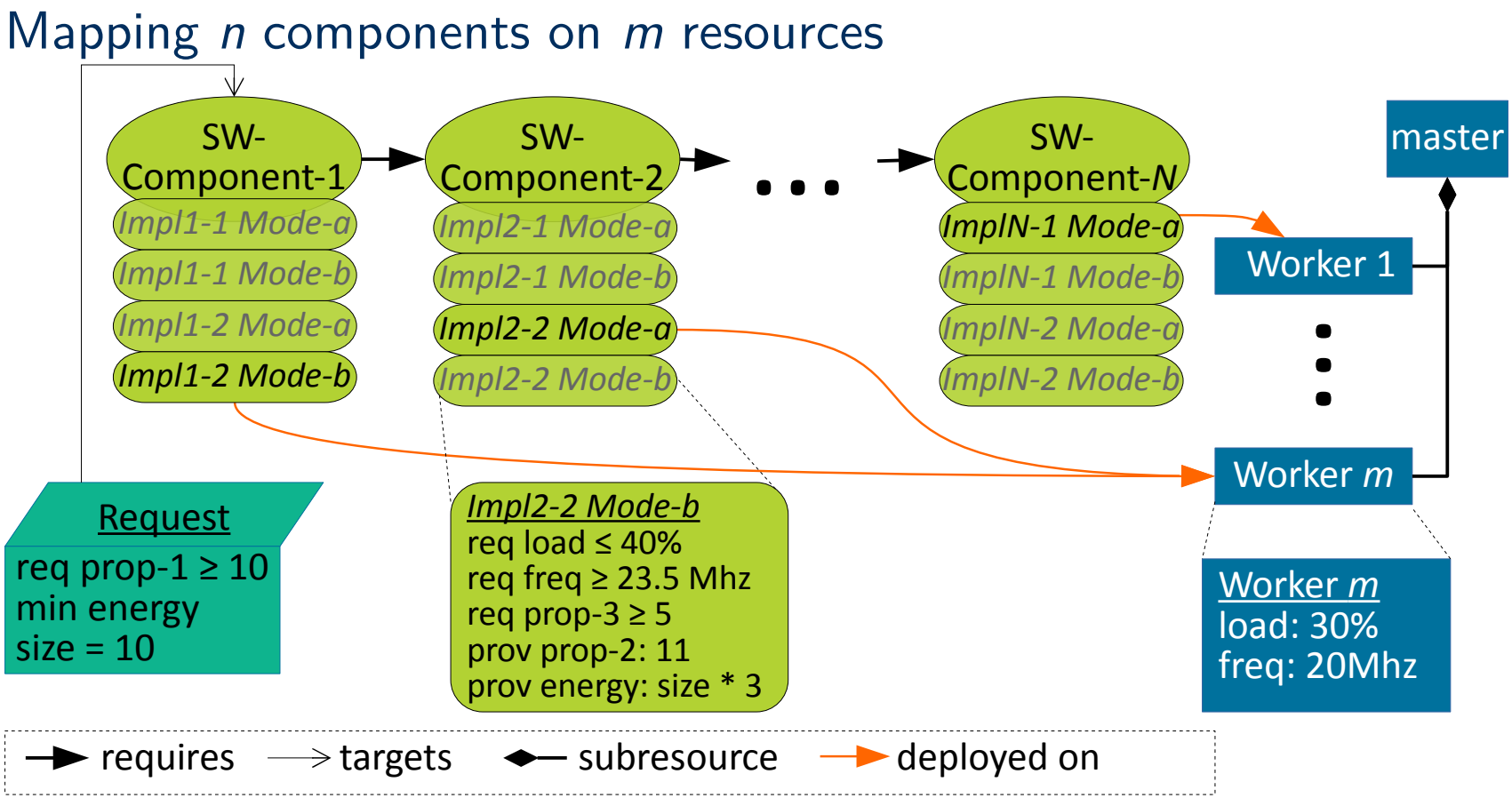


(b) CCM Variant Model of a Hardware Landscape Comprised of 2 Servers.

The optimization problem



The optimization problem solved



What was the problem

- Usage of ILP thought *unusable* for bigger systems
  - Current measurements disprove this, see slides 11 – 13
- EMF-Models and some of their elements ambiguous or redudant
  - Component requirement possible on both, component- and mode-level
  - Structure and variant model contain similar information
  - Approach of structural model not easy to use (especially for ILP-Generation)
- Currently, ILP generated from scratch for each request

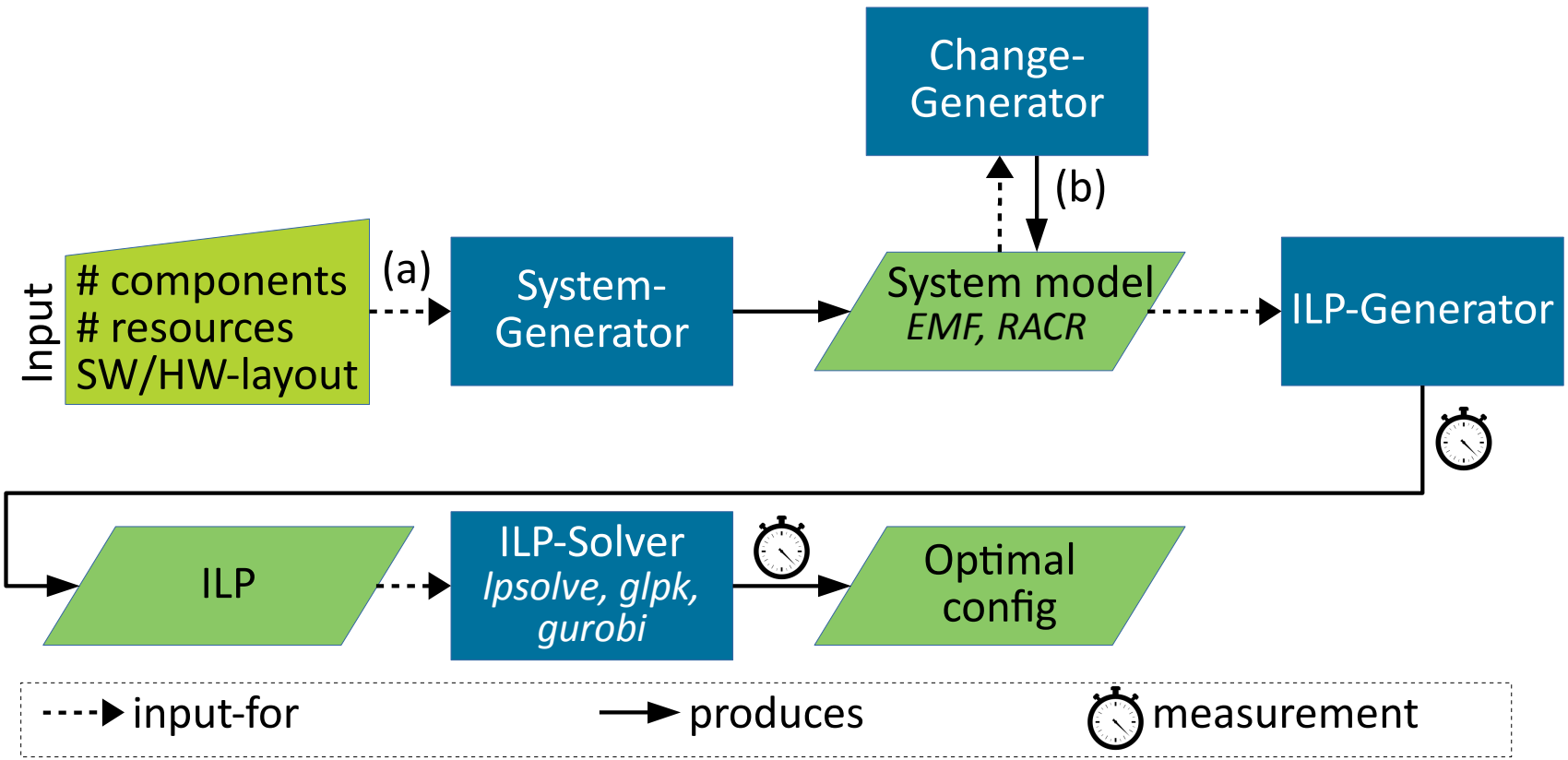
How we want to achieve scalability

- Use RACR [Bür12]
  - **R**eference **A**ttribute Grammer **C**ontrolled **R**ewriting
  - Specify knowledge as an ASG[1] whose structure is defined by a RAG[2]
  - RAG is a combination of structural and variant model, avoiding duplicate information
  - Analyses run on ASG now run inherently **incremental** and are defined **declaratively**

[1] Abstract Syntax Graph, i.e. an Abstract Syntax Tree with references

[2] Reference Attribute Grammar

Test setup



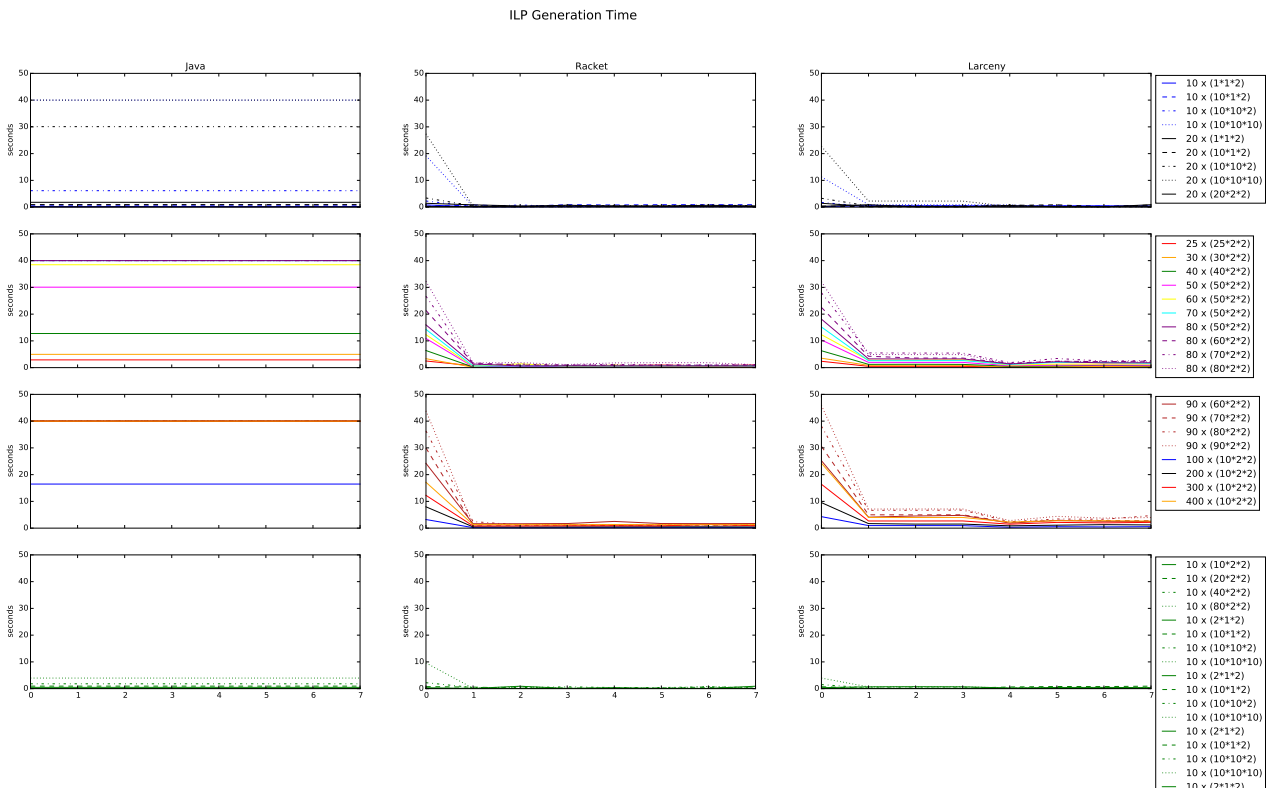
(a) Initial creation, (b) HW changes

Measurements

- Setup
  - System Generator to generate ILP for increasing size of systems
  - 23 different sizes of systems
  - ILP-Solving with 40sec timeout
- ILP-Solving using existing Java/EMF-based, old ILP format
  - Timeouts: glpk 10/23, lpsolve 8/23
- ILP-Solving using Scheme/RACR-based, enhanced ILP format
  - All but one systems solved within 5sec (outlier 12sec)

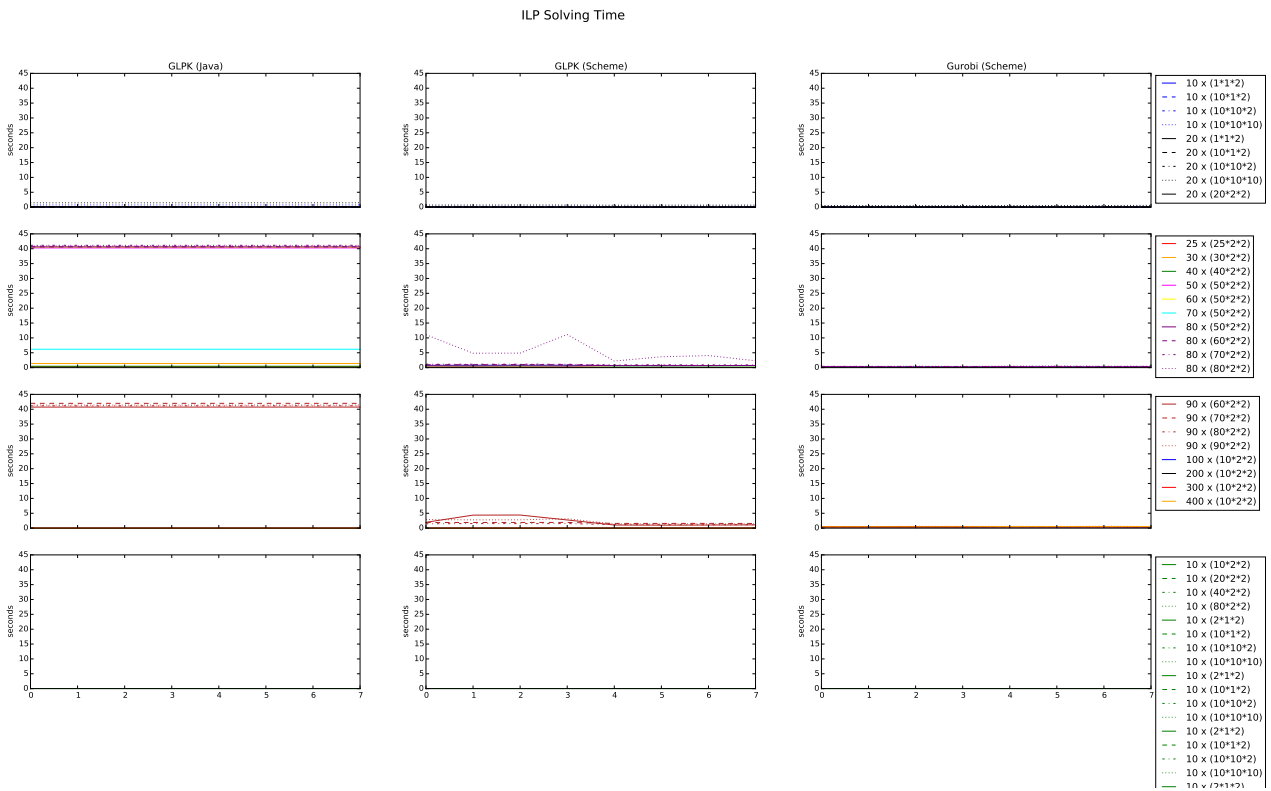
Measuring generation times

- Not quite there yet:

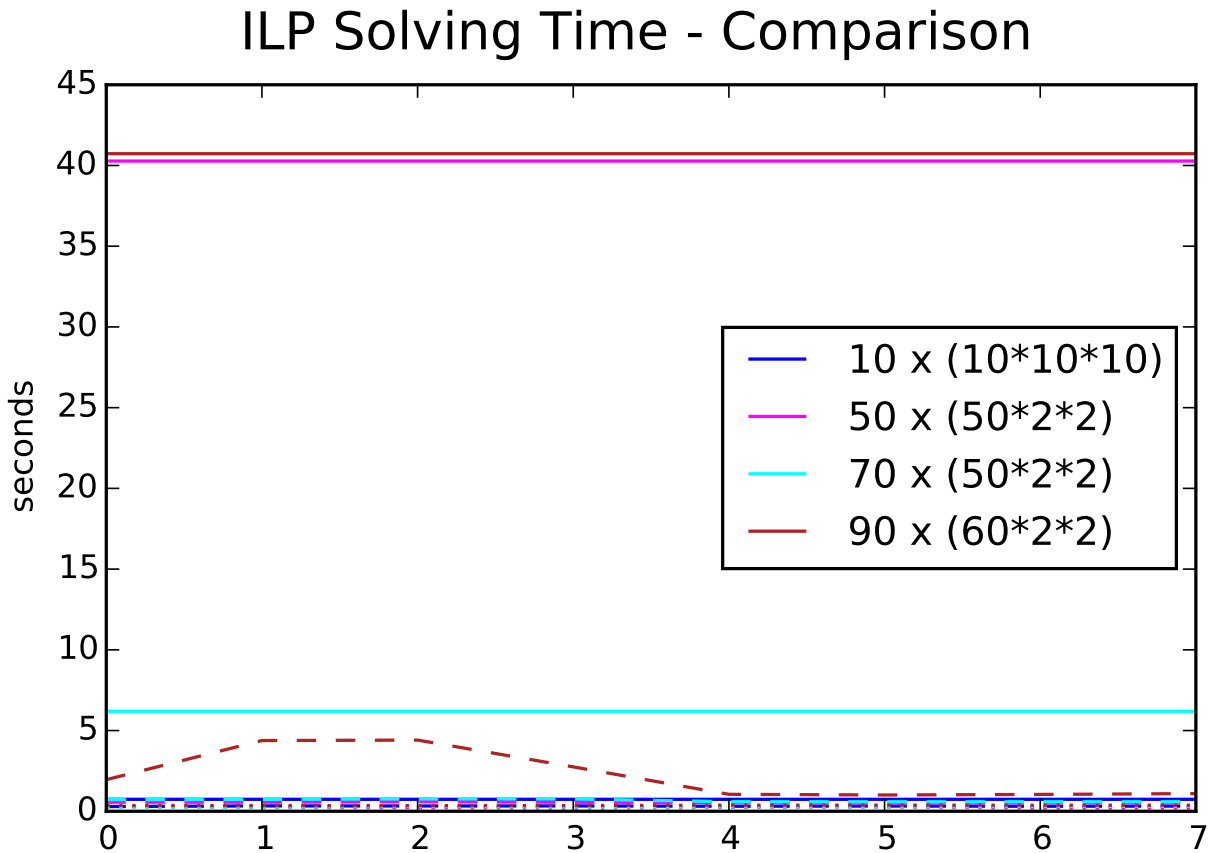


Measuring solving times

- Promising results



Measuring solving times (Detailed)



- solid = GLPK, old format, dashed = GLPK, enhanced format
- dotted = Gurobi, enhanced format ($\leq$ 1sec)

Current pitfalls

- Different input formats accepted by lp_solve and glpk
  - Transformation (mostly syntactical) needed
- Slow running Larceny
  - Unexpected as Larceny compiles to machine code
- Caching not fully exploited
  - Some constraints still unnecessarily recomputed

General Facts

- `https://bitbucket.org/rschoene/racr-mquat`
- Main language: Scheme

Language	files	blank	comment	code
Scheme	12	159	168	1222
Python	6	49	16	284
SUM:	19	212	185	1546

Where we should go next

- Do not transform to ILP
  - Implement a heuristic similar to RACRtune demo[3] of Daniel Langner and Johannes Mey
- Apply static analysis where appropriate, e.g.
  - Abstract Interpretation [CC77, Ros90] to estimate energy consumption [JM06, RMM03]
  - Describe decisions [Dan15]
  - Eliminate unreachable configurations
  - Unify constraints (in contracts) of modes
- Extend AG
  - Describe multiple systems and their interaction, e.g. [WSG+13]
  - Include behavior model for more fine grained description

[3] Shown at HAEC review and OUTPUT'15, paper in progress

An example application of static analysis

Worst Case Execution Time (WCET) squeezing [KKZ13]

- Combines ILP solving with Symbolic Execution (SE) [Kin76]
- Iterative, alternating, automatic approach
- SE either tightens found bound – or proves it precise

Application to HAEC use case

- Do Worst Case Energy Consumption (WCEC) squeezing
  - based on energy contracts

References I

[Bür12] Christoff Bürger. Racr: A scheme library for reference attribute grammar controlled rewriting. 2012.

[CC77] Patrick Cousot and Radhia Cousot. Abstract interpretation. In *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages - POPL '77*, pages 238–252, New York, New York, USA, January 1977. ACM Press.

[Dan15] Antonia Danylenko. *Decision Algebra: A General Approach to Learning and Using Classifiers*. PhD thesis, Linnaeus University, Växjö, Sweden, 2015.

[GKP+14] Sebastian Götz, Thomas Kühn, Christian Piechnick, Georg Püschel, and Uwe Aßmann. A models@ run. time approach for multi-objective self-optimizing software. In *Adaptive and Intelligent Systems*, pages 100–109. Springer, 2014.

[GWC+10] Sebastian Götz, Claas Wilke, Sebastian Cech, Johannes Waltsgott, Ronny Fritzsche, Jan Reimann, and Matthias Schmidt. *THEATRE Resource Manager Interface Specification v. 1.0*. Techn. Univ., Fakultät Informatik, 2010.

[JM06] R. Jayaseelan and T. Mitra. Estimating the Worst-Case Energy Consumption of Embedded Software. In *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*, pages 81–90. IEEE, 2006.

[Kin76] James C. King. Symbolic execution and program testing. *Communications of the ACM*, 19(7):385–394, July 1976.

References II

[KKZ13] Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. Wcet squeezing: on-demand feasibility refinement for proven precise wcet-bounds. In *Proceedings of the 21st International conference on Real-Time Networks and Systems*, pages 161–170. ACM, 2013.

[RMM03] Cosmin Rusu, Rami Melhem, and Daniel Mossé. Maximizing rewards for real-time applications with energy constraints. *ACM Transactions on Embedded Computing Systems*, 2(4):537–559, November 2003.

[Ros90] Mads Rosendahl. Abstract interpretation using attribute grammars. pages 143–156, September 1990.

[Sah96] Sartaj Sahni. Scheduling master-slave multiprocessor systems. *IEEE transactions on Computers*, (10):1195–1199, 1996.

[WSG+13] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M. Göschka. On patterns for decentralized control in self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II*, volume 7475 LNCS, pages 76–107. Springer, 2013.