GUROBI
OPTIMIZATION

Recent developments in MIP solvers

Gurobi Background

- Gurobi Optimization, founded July 2008
  - Zonghao Gu, Ed Rothberg, Bob Bixby
- Gurobi Version 1.0 released May 2009
- History of rapid, significant performance improvements
  - Close to 2x average speedup year-over-year
- History of continuing, significant innovations
  - Free academic licenses
  - First cloud offering
  - Compute Server for client-server applications
  - Distributed algorithms

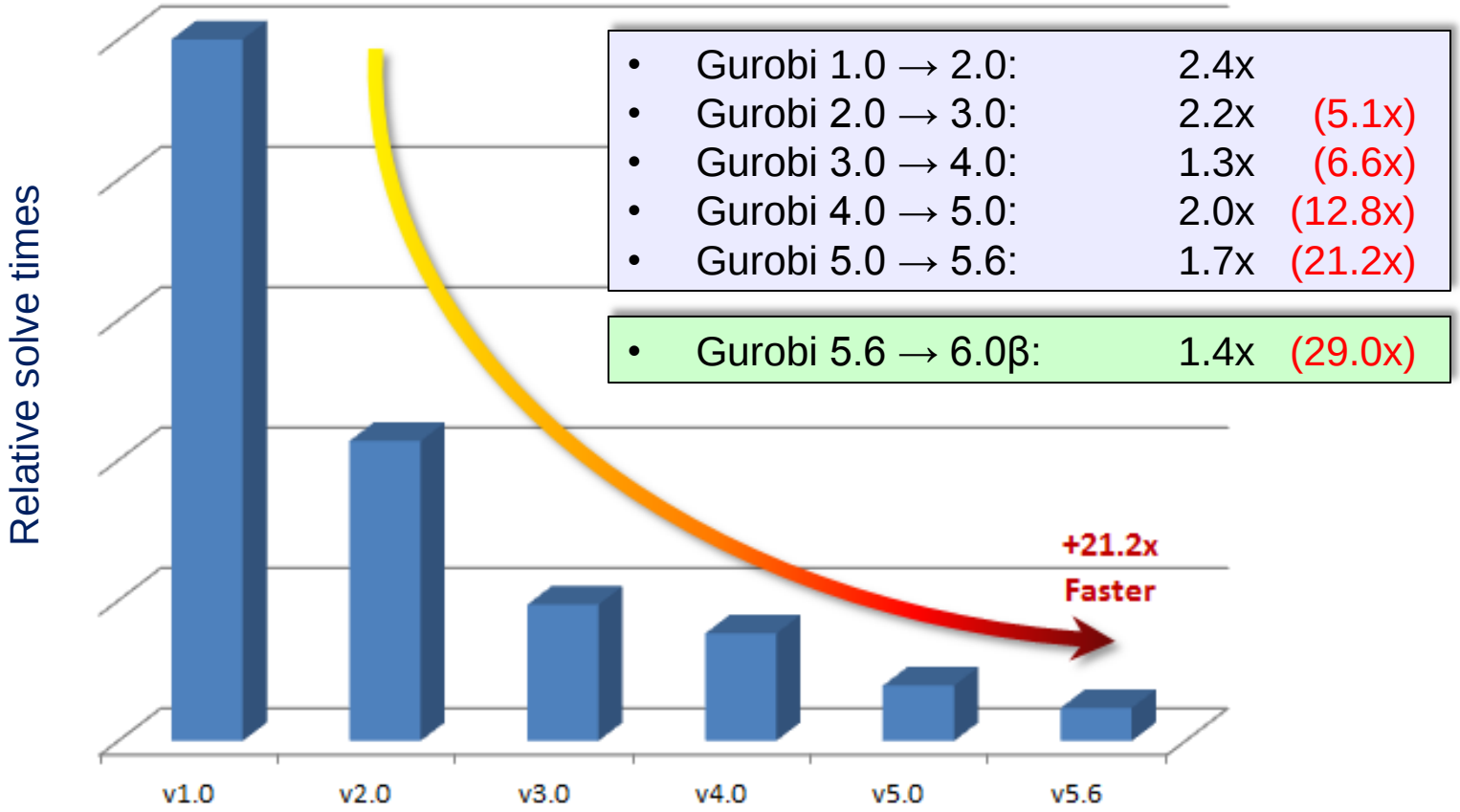
Recent Areas of Significant Change

- Performance
- Enabling Remote Computation
- Direct Modeling of Piecewise-Linear Approximations
- Constraint “Hints”

Performance

MIP Performance Improvements

- Version-to-version improvements:
  (Geometric mean speedup for models in our internal model set where at least one of the solvers takes more than 100s to solve)



Performance Improvements (6.0 vs 5.6)

Problem Class	> 1 sec	> 100 sec
	Speed-up	Speed-up
INTERNAL		
LP	1.06x	1.07x
MIP	1.20x	1.35x
MIQP	1.18x	2.30x
MIQCP	1.28x	2.07x
EXTERNAL		
Mittelmann "Easy"	1.15x	1.23x
Mittelmann Optimality Benchmark*	1.12x	1.09x

Notes:

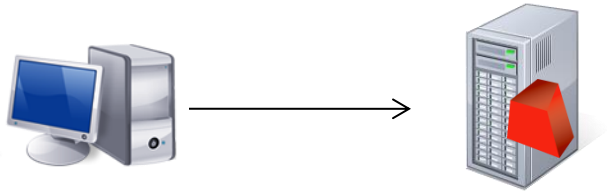
1. 4 threads, 10000s time limit
2. * Average over 5 random seeds

Remote Computation – Gurobi Compute Server

Gurobi Compute Server

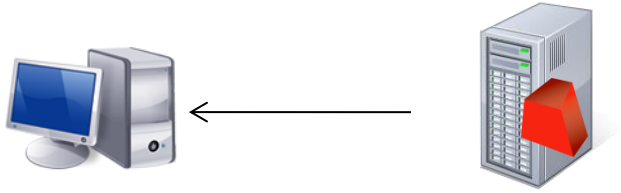
- Client computer uses Gurobi API to build model

- Client computer passes model data to server



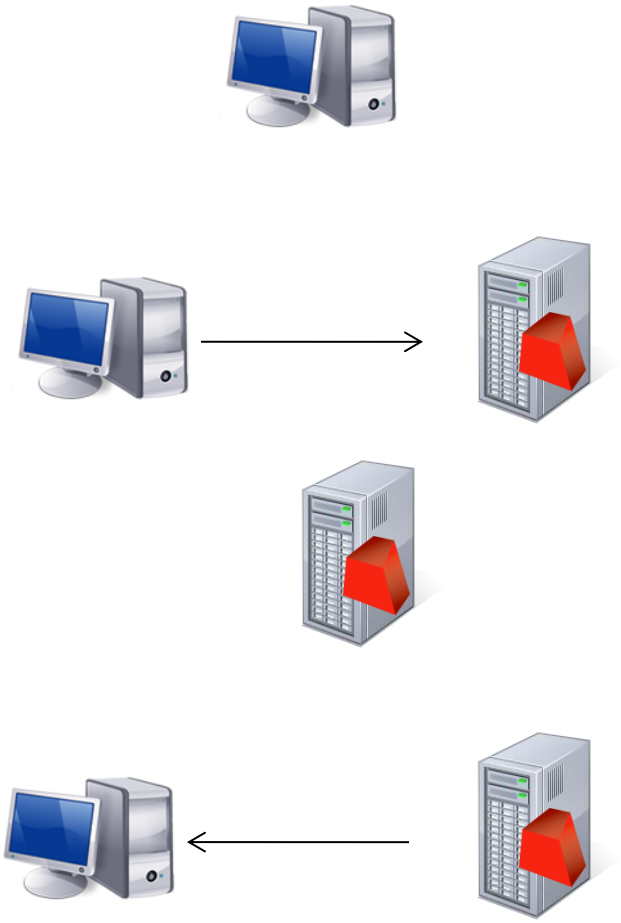
- Server solves the model

- Result values returned to client computer



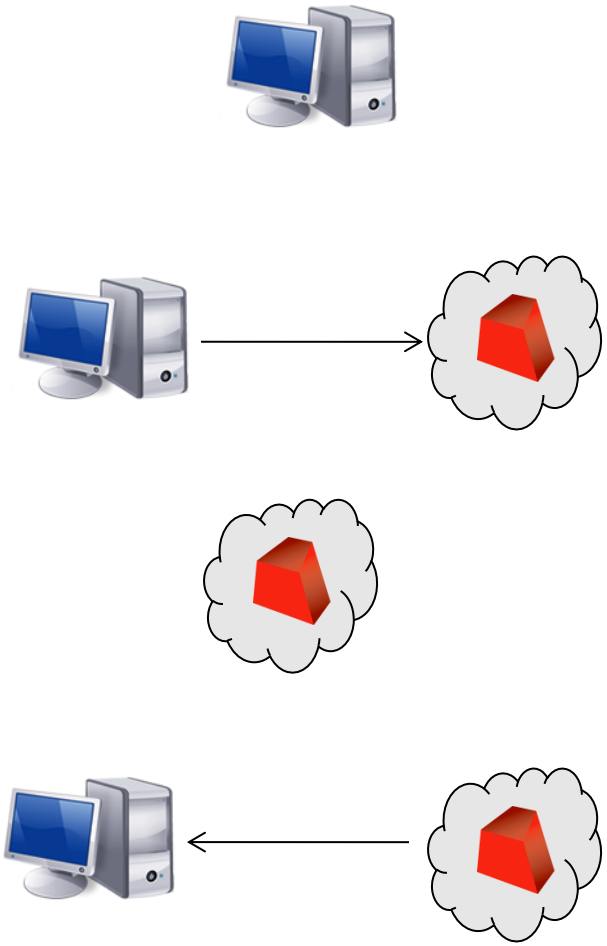
Use Case – Thin Client

- NYISO uses Gurobi Compute Server
  - Offload optimization from slow clients to fast servers
    - 4X improvement in MIP solution times
    - No specialized implementation work required



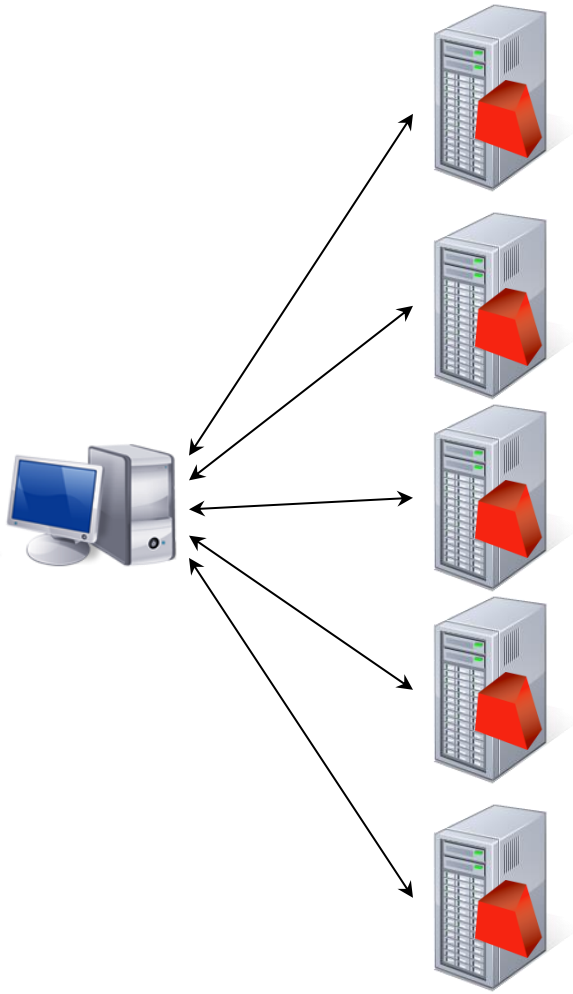
Use Case – Cloud Computing

- Gurobi Cloud hosted on Amazon EC2
- Launch an instance (or 100)
- Offload computation to it over the Internet
- Pay by the hour



Use Case – Distributed Computing

- One manager offloads computation to multiple workers
- Multiple machines cooperate to solve a single MIP model



Distributed Algorithm Performance

- On a few well-known hard models:
  - Model *a1c1s1*: 13.3X on 49 machines
  - Model *seymour*: 14.9X on 32 machines
- On a broad test-set:
  - 2.9X mean improvement on 8 machines over all models that take 100s-1hr from the MIPLIB 2010 benchmark set

- MIP search tree must have the "right" structure to benefit
  - Bad news: UC models don't

Piecewise-Linear Objective Functions

Motivation

- Applications
  - Approximations of non-linear objective functions
- Traditional approaches
  - One variable for each piece
- Objective of our “new” approach:
  - Extend simplex algorithm (primal and dual)
  - Handle PWL terms directly, without introducing extra variables and constraints, with the goal of enabling accurate and yet computationally tractable approximations of nonlinear functions

Performance Impact – Quadratic Objectives

- Solve convex quadratic objective model using PWL approximation
  - 100 pieces: 2X faster
  - 1000 pieces: 5X faster
  - We see significant opportunity for further improvement
- Ultimate goal:
  - General purpose tool for building accurate PWL approximations of arbitrary (separable) non-linear objectives and constraints

Constraint “Hints”

Constraint Hints

- Label each constraint based on how aggressively it should be pulled into the active model
  - Always in (default)
  - Pull it into model to cut off an integral solution
  - Pull it into model to cut off a continuous solution
- On customer SCUC model
  - Labeling <1% of constraints in model reduces time to first feasible solution by ~20%

Observations on Solution of Unit Commitment Models

UC Model Observations

- Unusual characteristics
  - Root bound is quite tight
    - Challenge is to find a good feasible solution
  - Heuristics are extremely expensive
    - Lots of continuous variables, plus slow LP relaxations
- Ripe for better heuristic approach

Variable Hints?

- Need a new abstraction to pass hints about feasible solutions from modeler to solver?

- What could users provide?
  - Likely preferred values for variables?
    - Fix this variable to 0
    - Fix this variable to relaxation value
    - Fix thie variable to 0 if relaxation value is 0
  - Prioritization on variable fixings?
    - To give MIP solver the freedom to choose how many to fix
  - Likely relationships between variable values?
    - "If x=1 then y is probably > 5"

- Better to handle this within MIP framework (if possible)
  - MIP solver already handles lower bound, upper bound, local improvement on new solutions, threading, distributed optimization, …