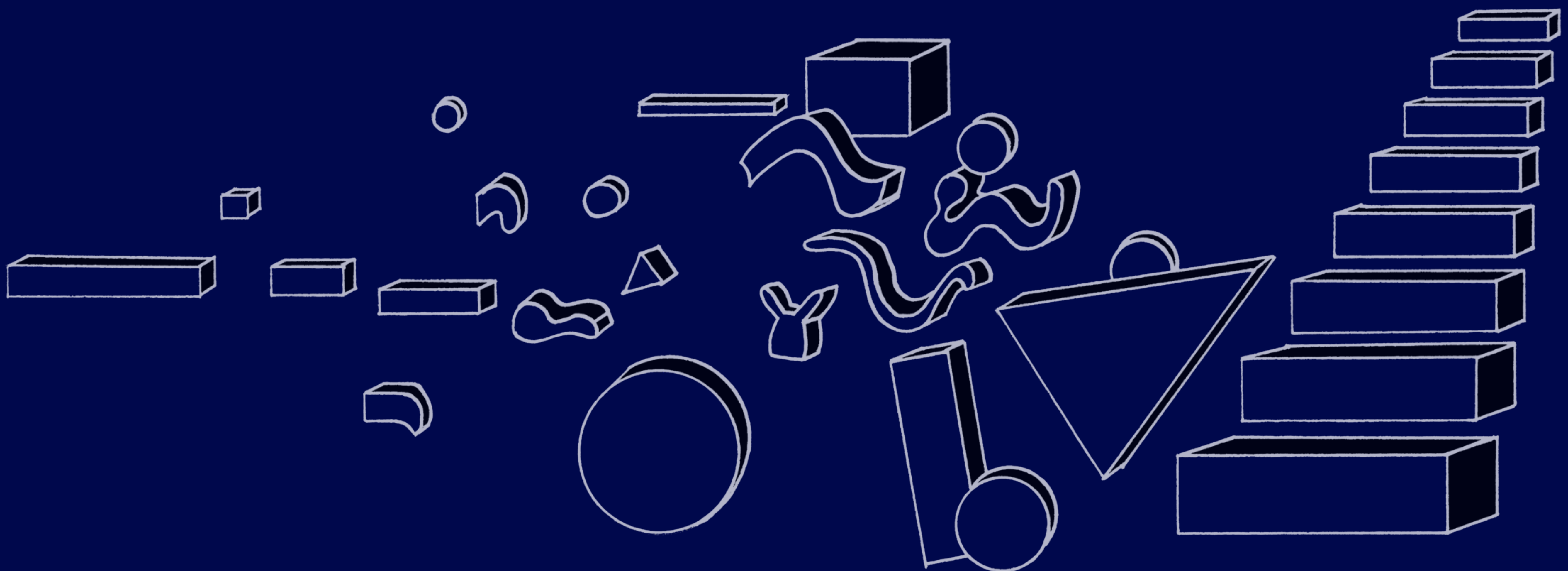
Parallelism in Constraint Programming

Carl Christian Rolf



Welcome...


Okay, people, let's get started on our orientation. I have 666 PowerPoint slides to cover.
Welcome to Hell
presented by Satan
This week's guest cartoonist: Wayno
Wayno
WAYNOCARTOONS.BLOGSPOT.COM

‘Tiny Sudoku’

X, Y, Z, Q must have values between 1 and 2
The values in each row and column must be unique

X	Y
Z	Q

Tiny Sudoku in Constraint Programming

X, Y, Z, Q must have values between 1 and 2
The values in each row and column must be unique

X	Y
Z	Q

$\Rightarrow$

$\{X, Y, Z, Q\} \in \{1..2\}$,

$X \neq Y$,

$X \neq Z$,

$Y \neq Q$,

$Z \neq Q$,

solve.

Solving Tiny Sudoku in CP

SOLVE
$\{X, Y, Z, Q\} \in \{1..2\}$
$X \neq Y, X \neq Z, Y \neq Q, Z \neq Q$

$\downarrow$ $X = 1$

$X = 1$
$\{Y, Z, Q\} \in \{1..2\}$

$\downarrow$

Evaluate constraints:

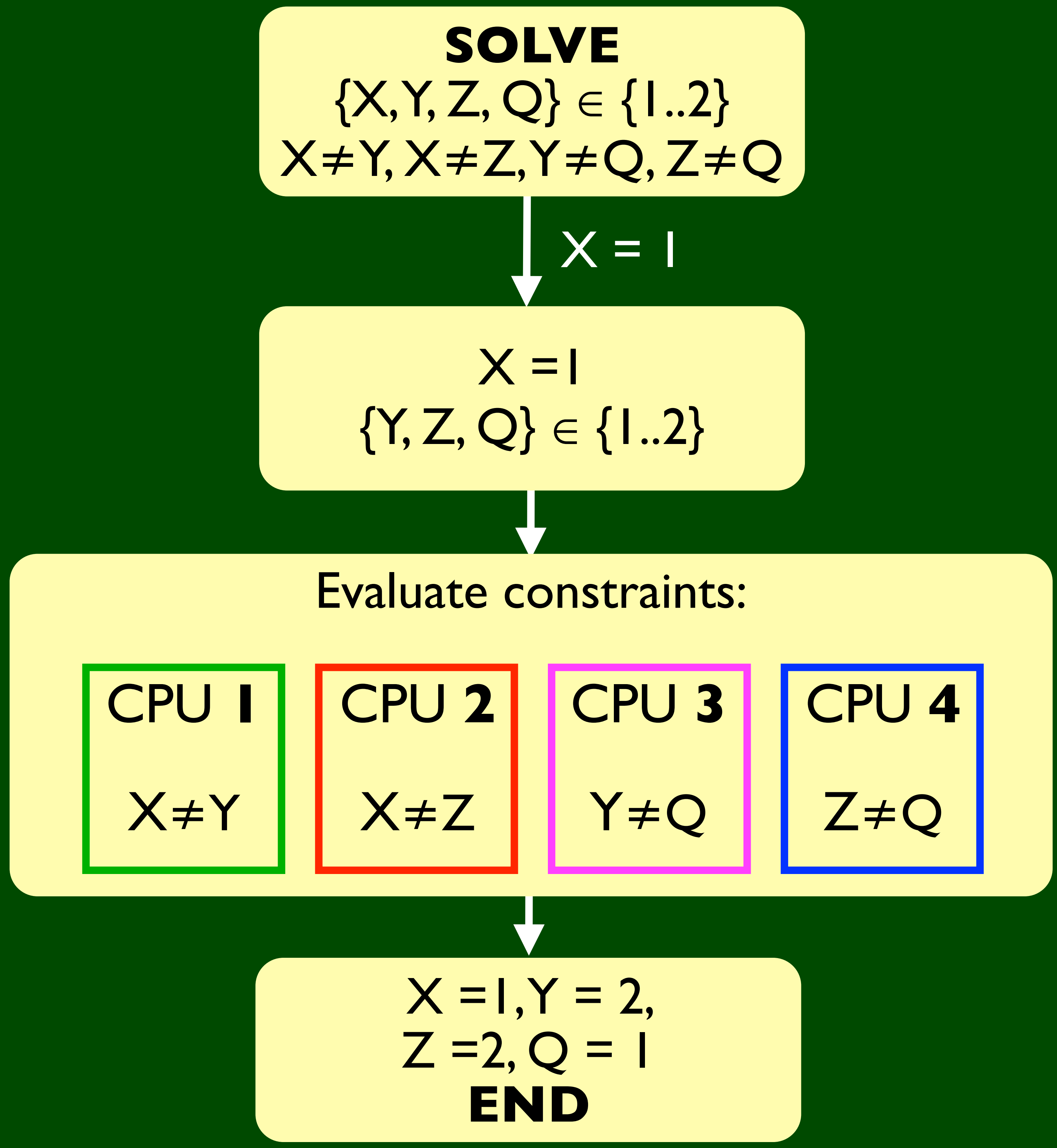
$X \neq Y, X \neq Z, Y \neq Q, Z \neq Q$

$\downarrow$

$X = 1, Y = 2,$
$Z = 2, Q = 1$
END

Parallel Consistency

Parallel Consistency for Tiny Sudoku

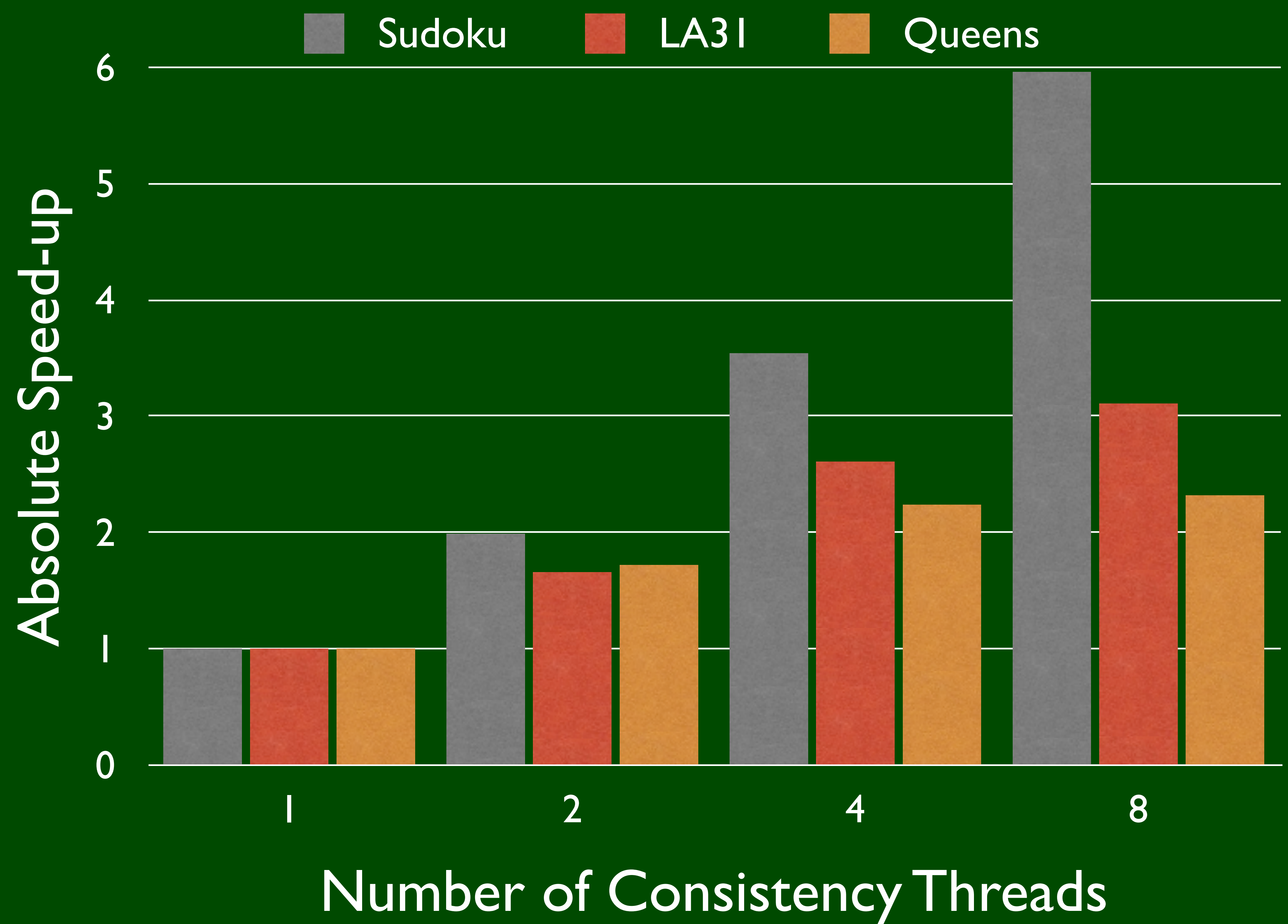


Parallel Consistency: Summary

- The ‘task-parallelism’ of constraint programming
- Previous work has not handled global constraints
- Might have to run more iterations of consistency
- Load-balancing depends on the problem model
- Hard to share data during consistency

Parallel Consistency: Performance

Run on an eight core Mac Pro

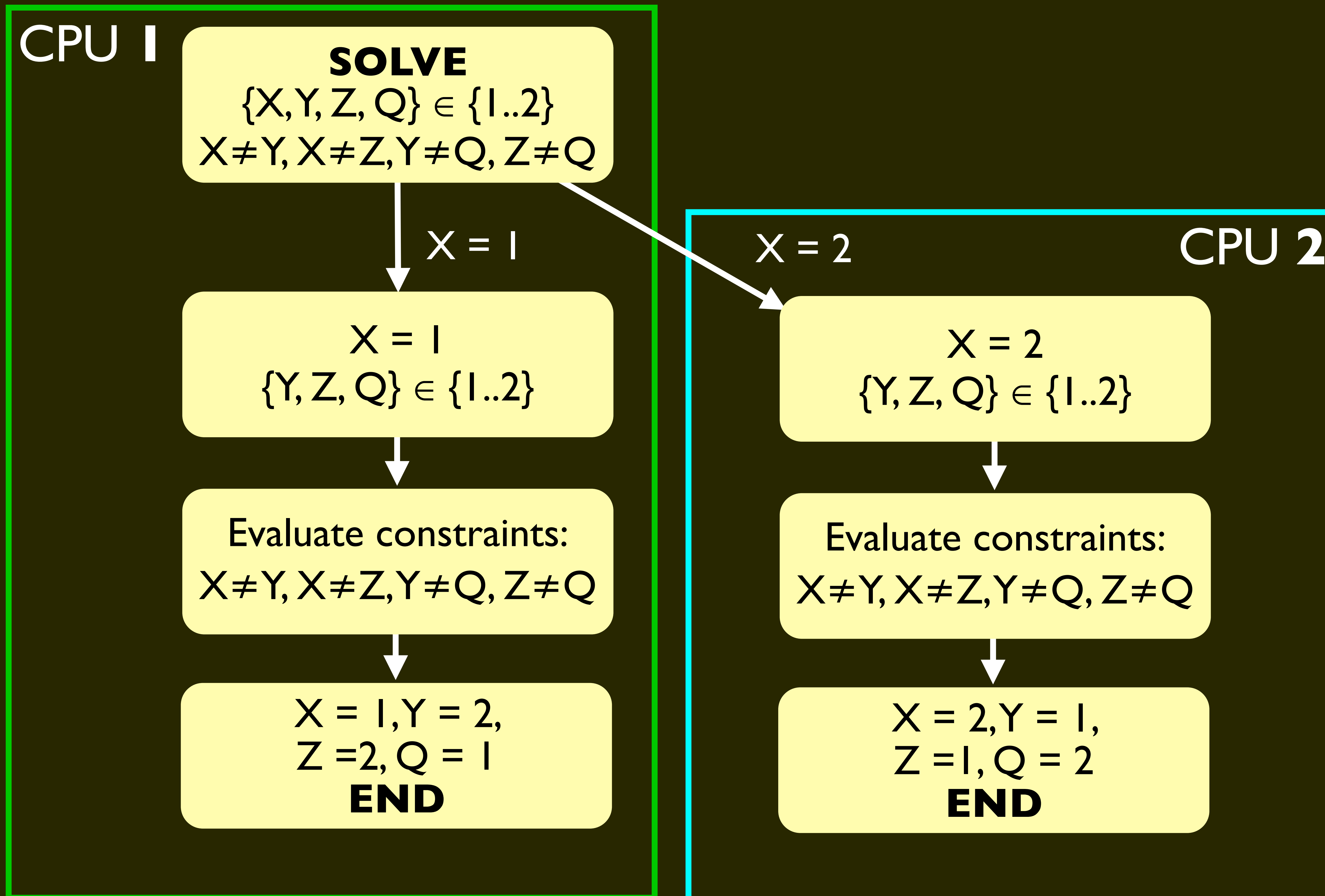


Parallel Consistency: Conclusions

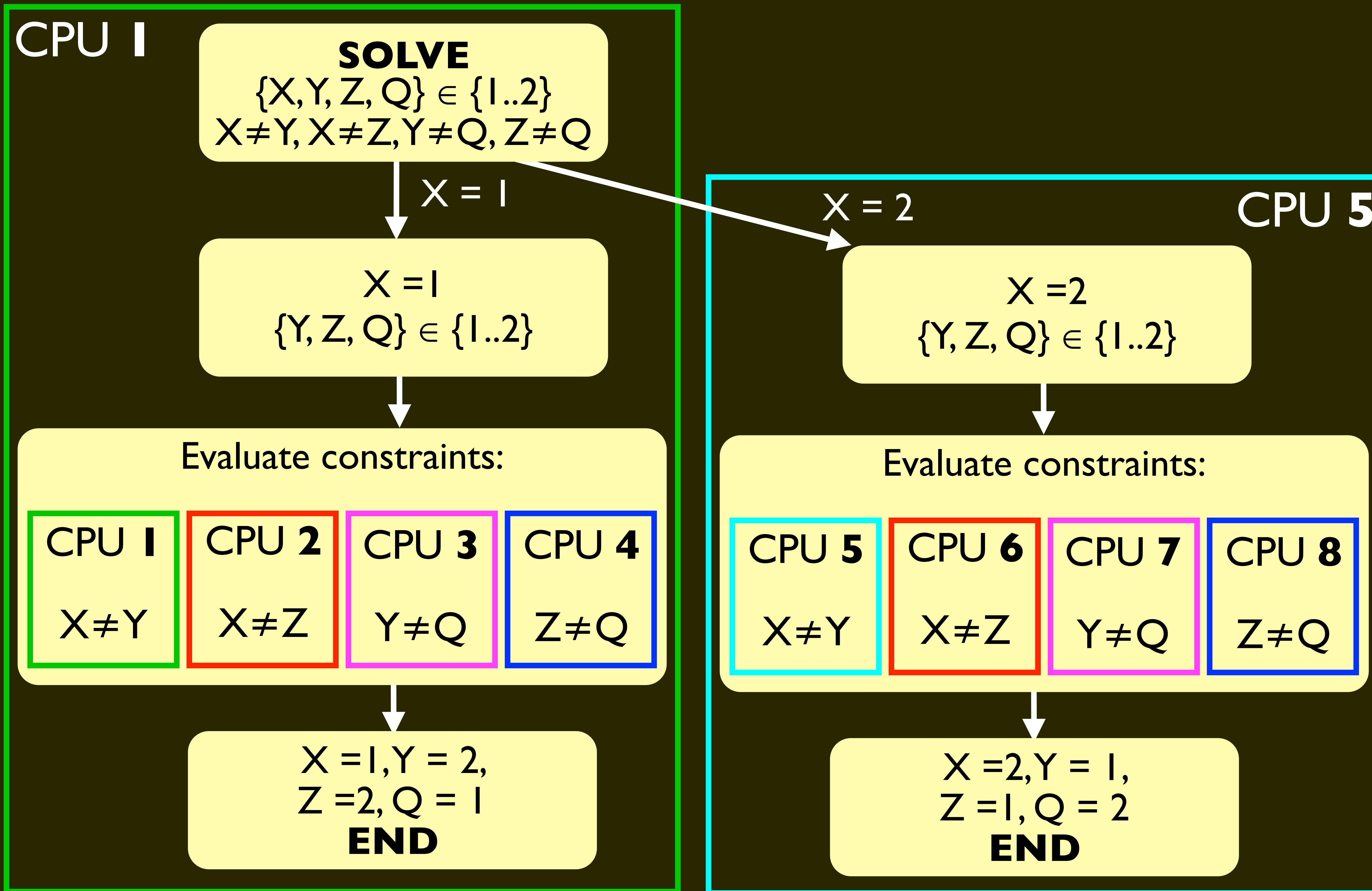
- Excellent performance for regular problems
- Some problems do not scale well, they need constraint-specific parallel consistency algorithms

Combining Parallel Consistency with Parallel Search

Parallel Search for Tiny Sudoku



Combining Parallelisms: Tiny Sudoku

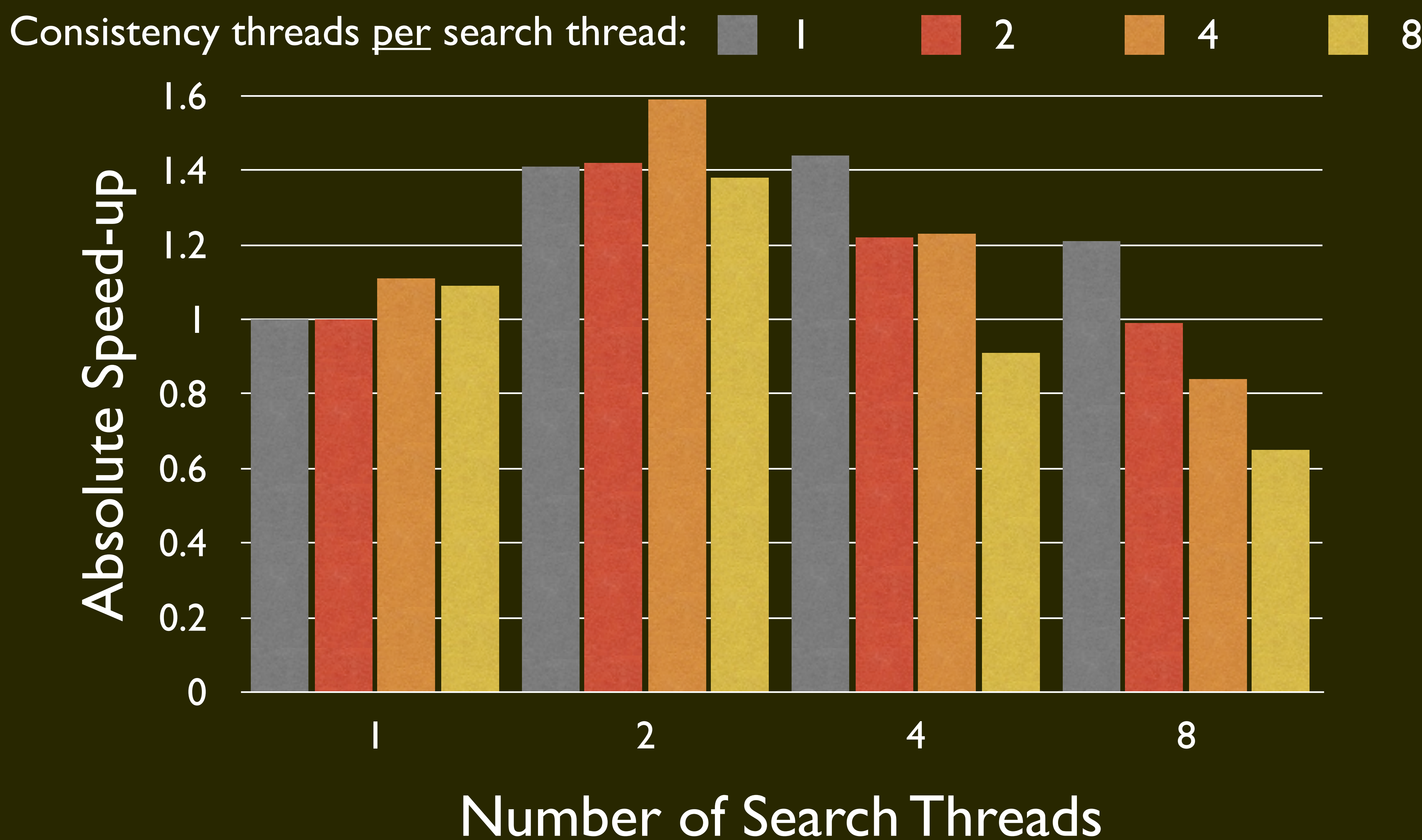


Combining Parallelisms: Summary

- Never studied before in constraint programming
- Easier to achieve a speed-up than if the solver only offers one type of parallelism
- Does not fit all types of problems, often needs problem-specific optimization of, e.g., thread allocation

Combining Parallelisms: Results

Finding 200 solutions to 100x100 Sudoku. Run on 8-core Mac Pro

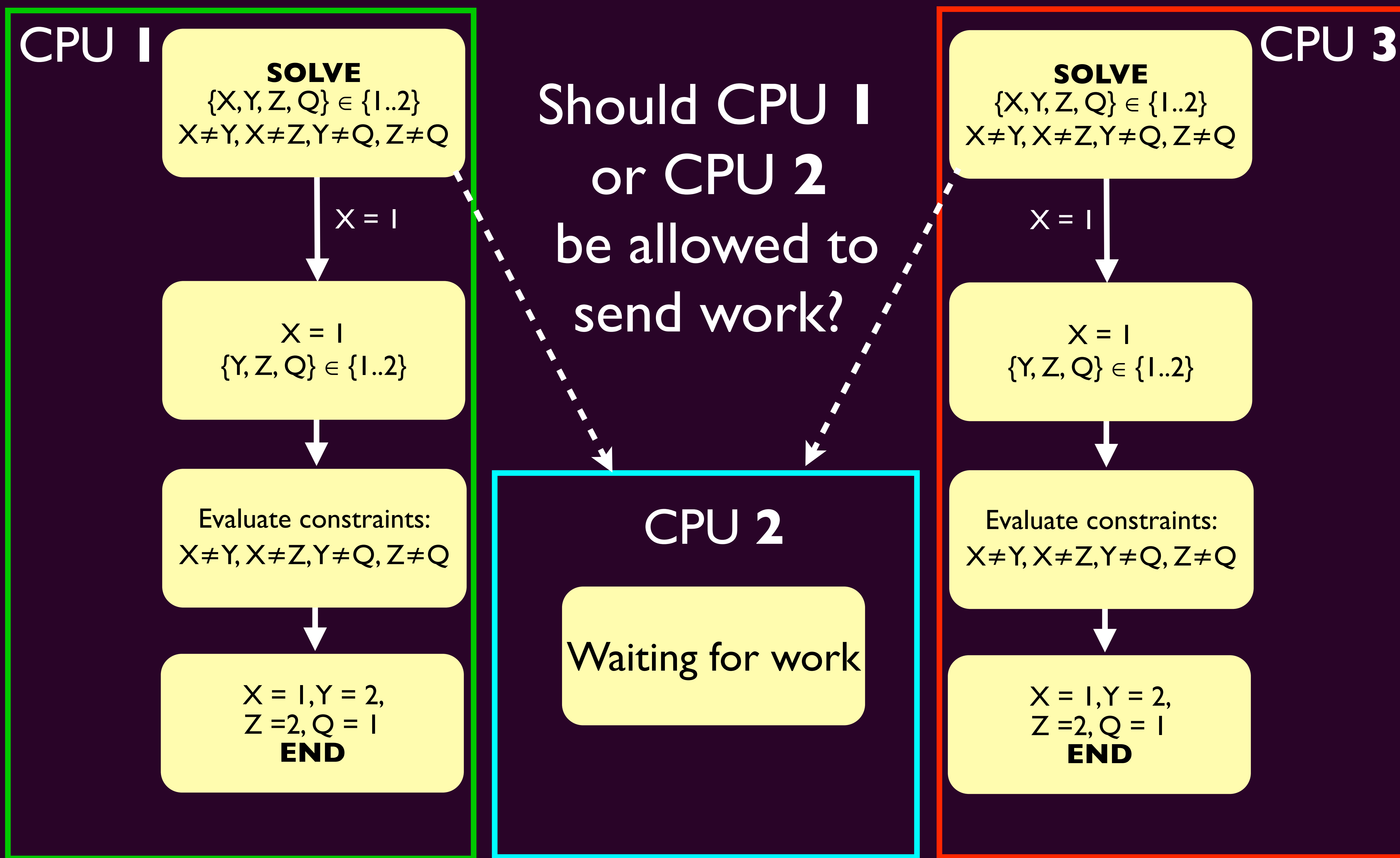


Combining Parallelisms: Conclusions

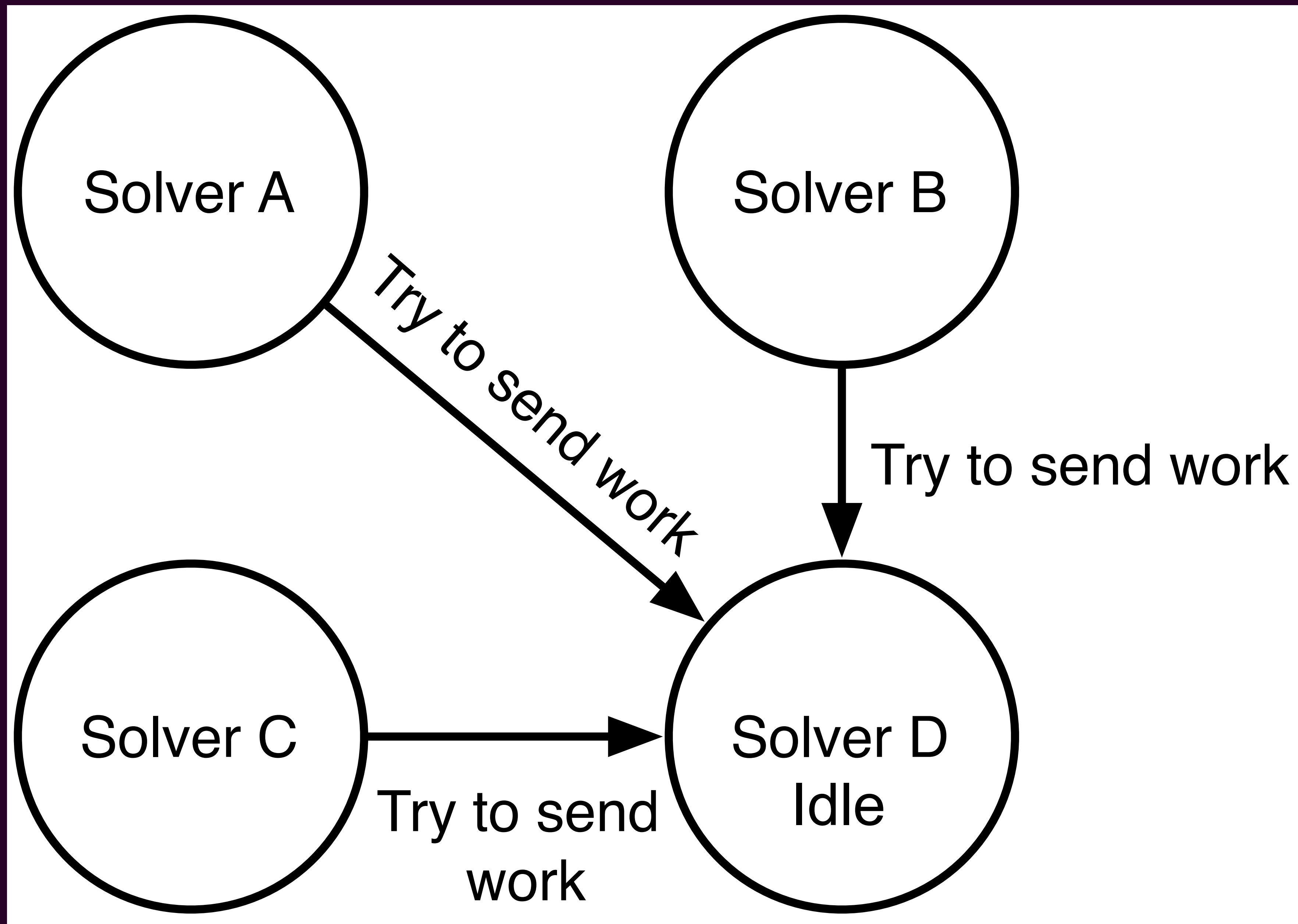
- Needs better control of mutual exclusion than currently offered by Java
- The problem must suit both types of parallelism to get a large performance benefit
- Problem-specific optimizations are necessary for good performance

Relative-Measured Load-Balancing

Load-Balancing for Tiny Sudoku



Relative-Measured Load-Balancing



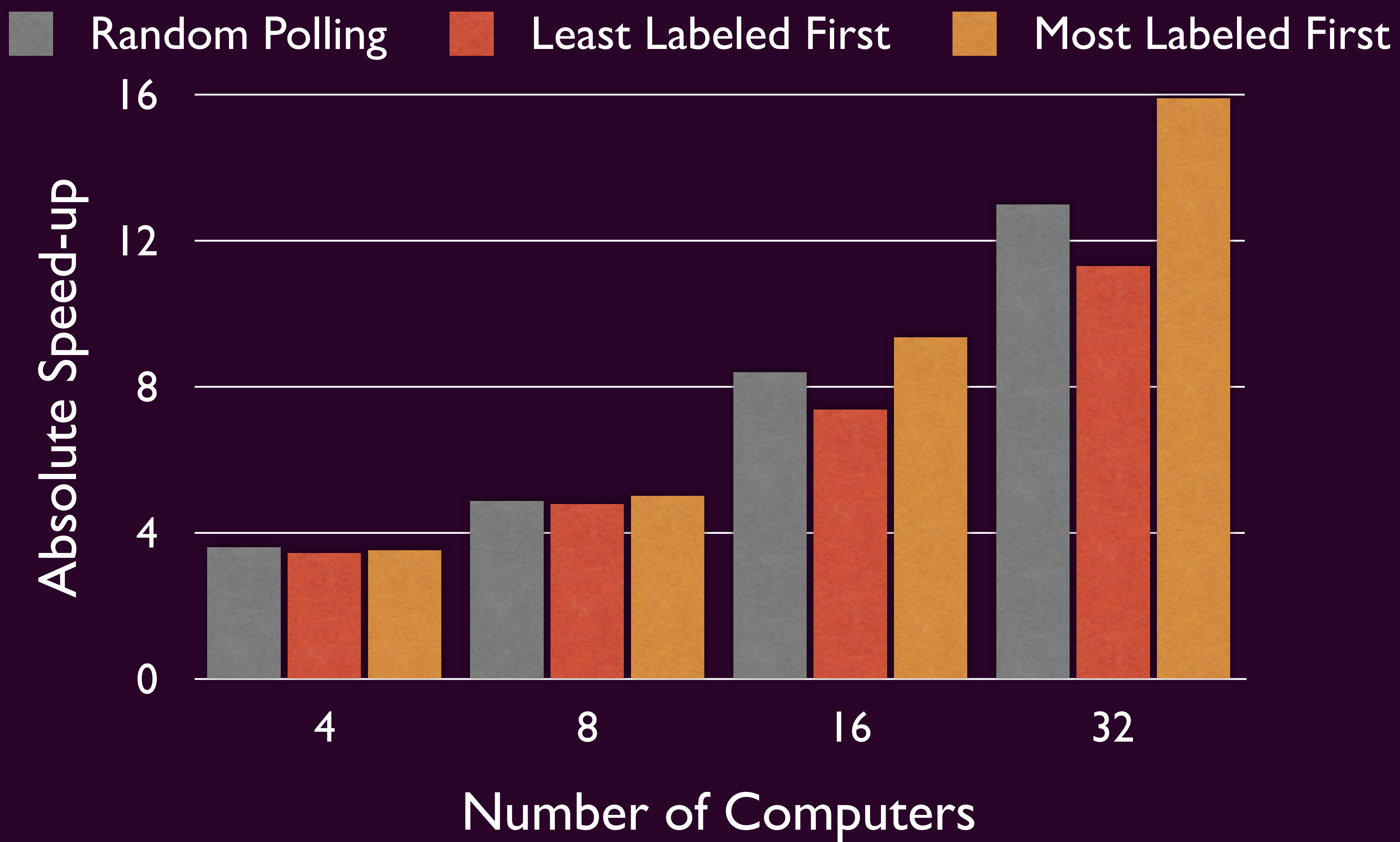
Busy solvers always compete for sending work to idle solvers

Relative-Measured Load-Balancing: Summary

- Infeasible to get an exact measure of the work-size due to the way CP solves problems
- We let solvers compete based on their work-size estimates
- We can use <u>any</u> measure that can be partially ordered
- Using measures from several solvers increases accuracy by eliminating systematic errors

Relative-Measured Load-Balancing: Performance

Golomb-12, proving optimality. Slowest and fastest measures

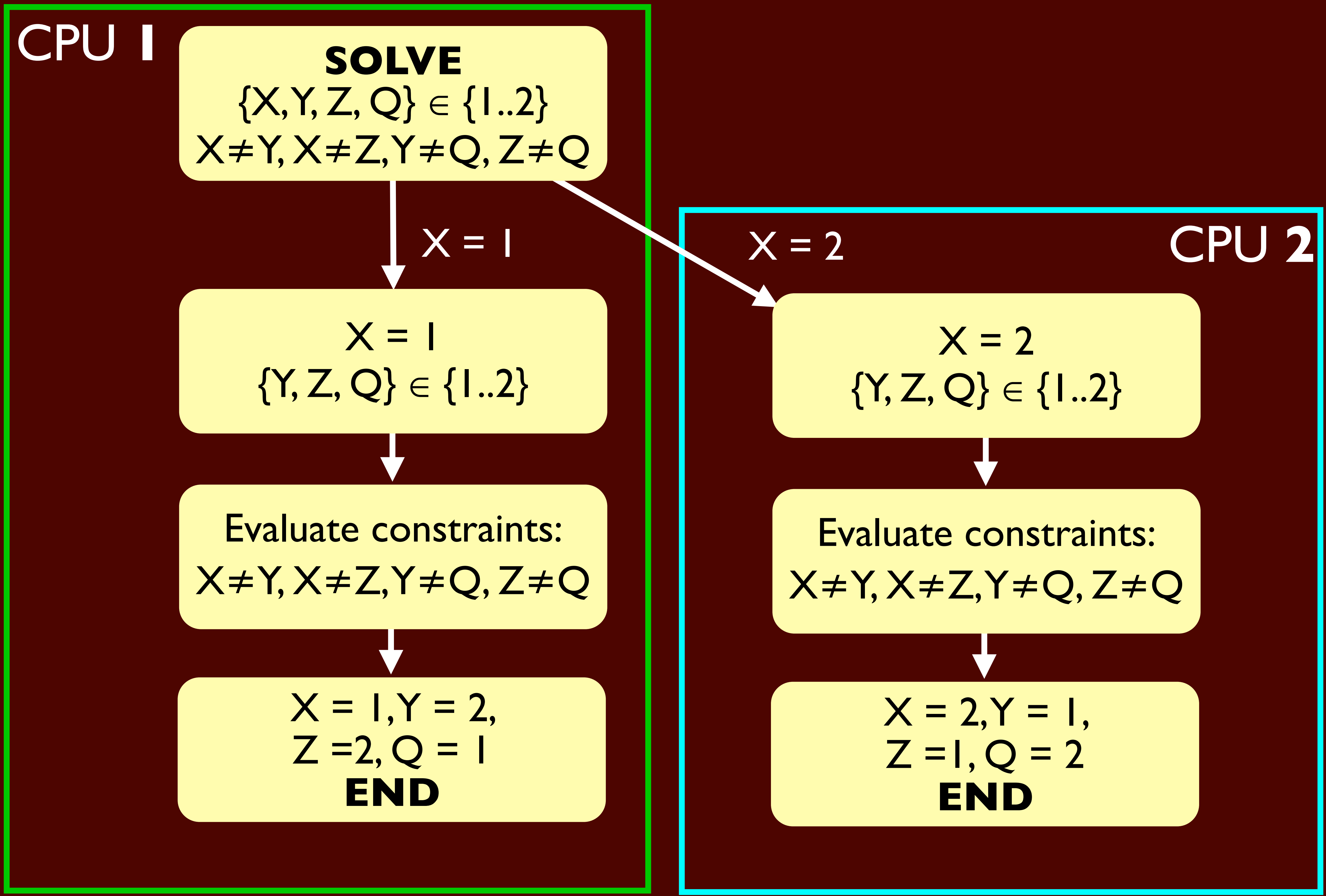


Relative-Measured Load-Balancing: Conclusions

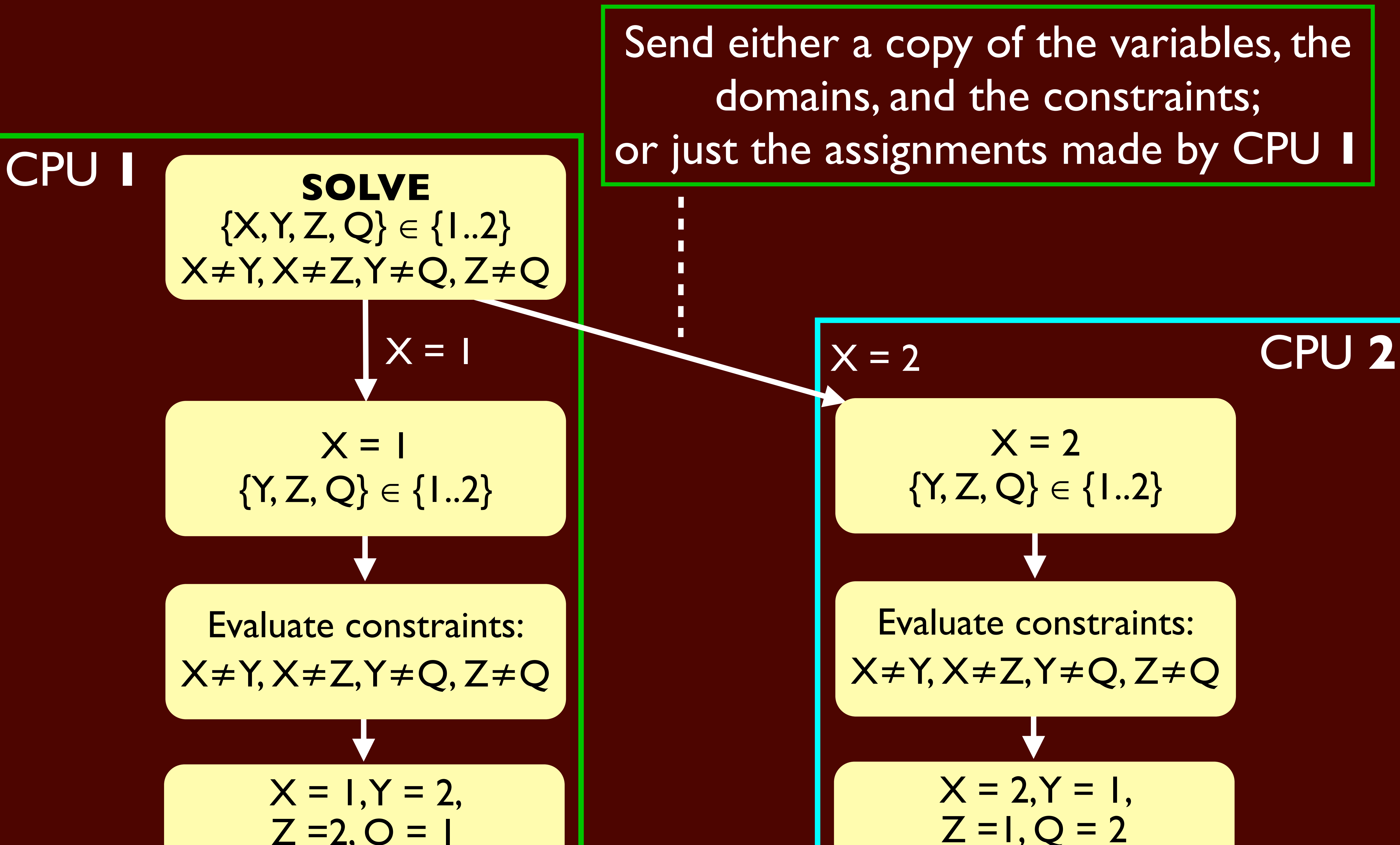
- Relative measures lets even simple estimates outperform random polling by over 20%
- Advanced measures can easily be used
- Performance benefit increases with the number of solvers

Dynamic Balancing of Communication and Computation

Dynamic Balancing for Tiny Sudoku



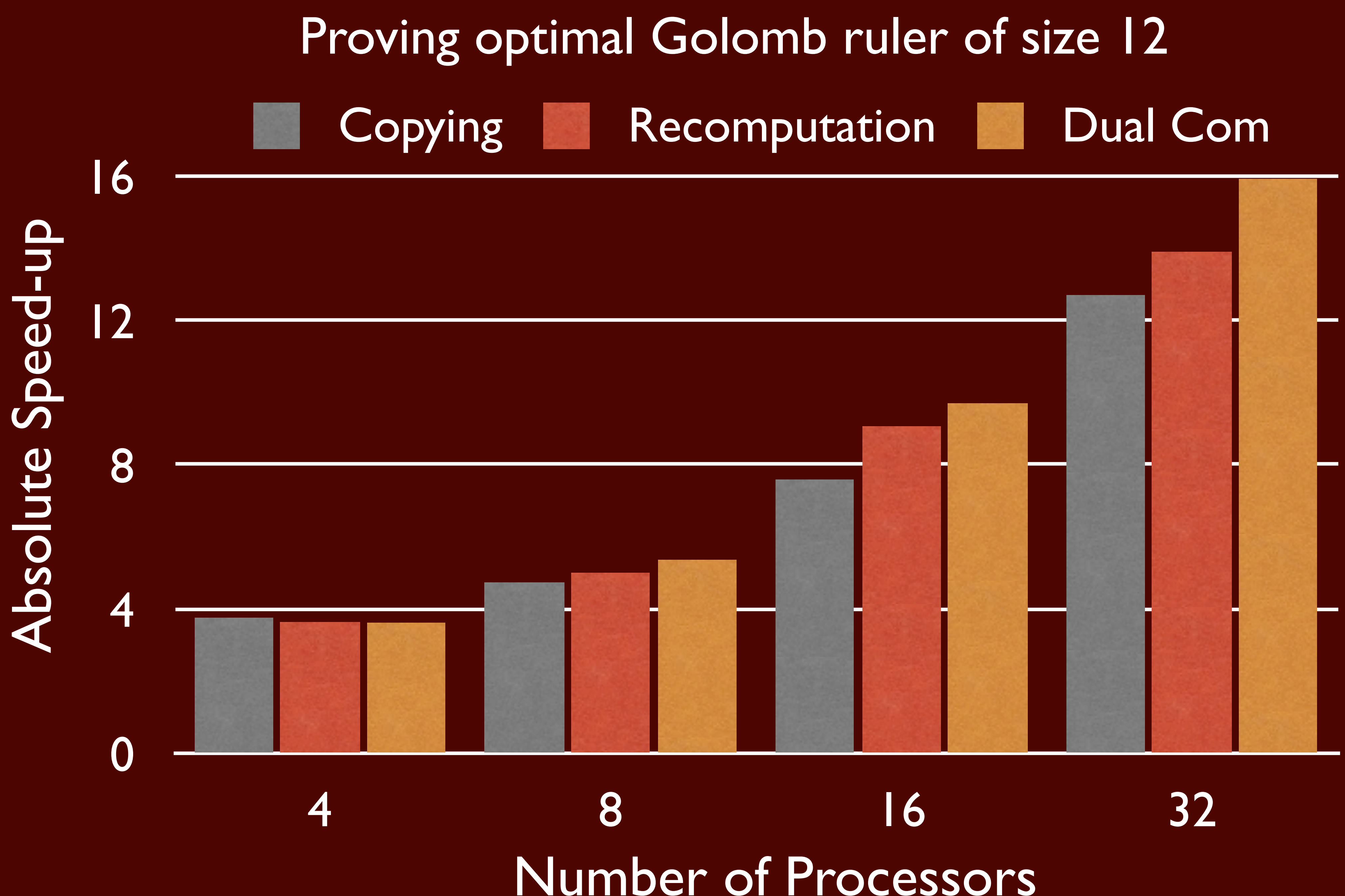
Tiny Sudoku: Zoomed in



Balancing Communication and Computation: Summary

- Copying sends a lot of data, but needs very little processing
- Recomputation often needs a lot of processing, but sends little information
- We estimate the network load to avoid getting stuck in performance bottlenecks

Balancing Communication and Computation: Results

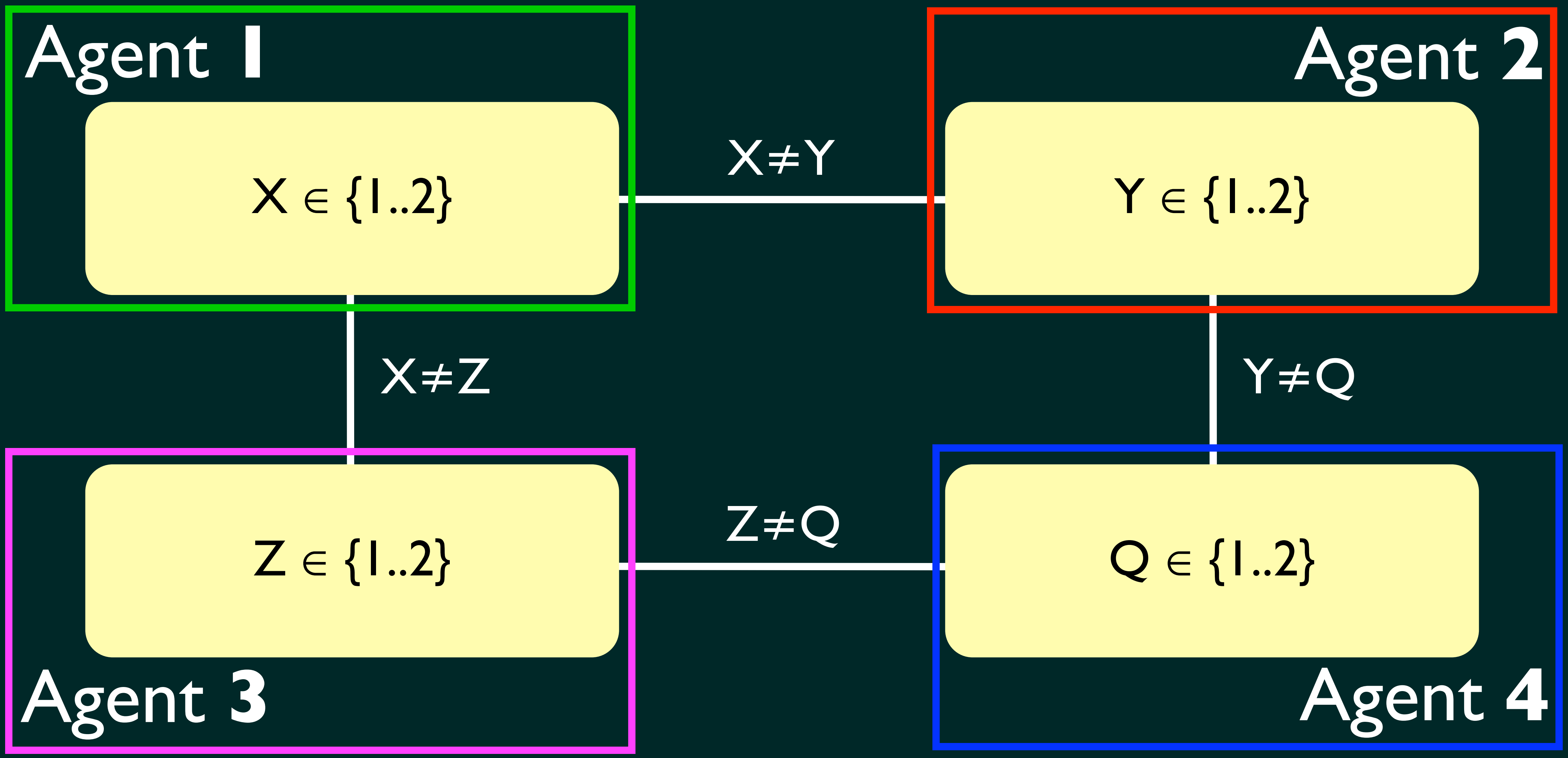


Balancing Communication and Computation: Conclusions

- Switching dynamically between copying and recomputation often increases performance
- Simple measure to estimate where the performance bottlenecks are

Distributed Constraint Programming with Agents (DCP)

Tiny Sudoku in DCP



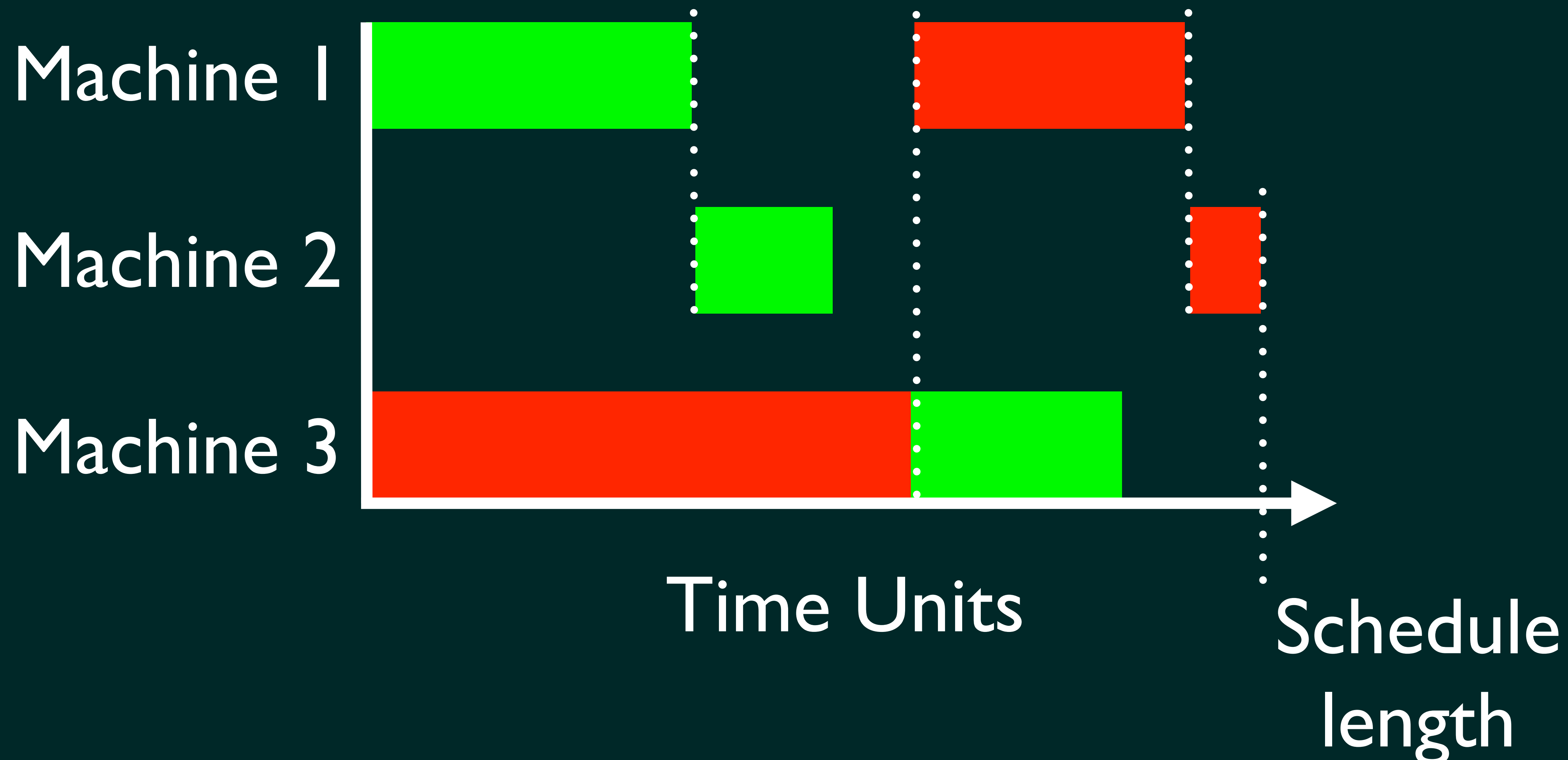
Solving starts in one agent, constraints communicate prunings

Our use of DCP

- To be used in UAVs in catastrophe areas
- We want independent agents to cooperate, to for instance share a heat camera
- We want to find good, preferably optimal, schedules

Example of Job Shop Scheduling

Two jobs, consisting of three tasks. Tasks have to execute on specific machines in a certain order



DCP: In Contrast to the Traditional Approach

- A full constraint solver in each agent
- A set of variables in each agent
- A set of *n*-ary (global) constraints in each agent
- No use of memory-demanding table constraints
- Advanced search methods

DCP: Experimental Results

Traditional Model

Problem	Time	Solution
LA01	>30min	936
LA04	>30min	976
LA05	>30min	720
MT06	87.7s	55

Our Model

Problem	Time	Solution
LA01	3.8s	666
LA04	10.8s	590
LA05	0.7s	593
MT06	3.0s	55

We <u>proved</u> the optimum of <u>all</u> problems, the traditional model of <u>none</u>

DCP: Conclusions

- Our model outperforms traditional models by orders of magnitude
- The best traditional approaches will remain slower, as speed-up is limited

Overall Conclusions

- We've developed and evaluated several new ways to parallelize constraint solving with global constraints
- Our model of distributed constraint programming vastly outperforms traditional approaches
- Parallelism in constraint programming can, with no understanding of parallel programming, give well-scaling performance for many kinds of problems

Lastly, a nice Quote

"We finish this review with a single paper, probably one that best represents the state of the art [31]."

[31] C.C. Rolf and K. Kuchcinski. *Combining parallel search and parallel consistency in constraint programming*. In TRICS workshop at CP2010, pages 38–52, 2010.

From:

I. P. Gent, C. Jefferson, I. Miguel, N. C. A. Moore, P. Nightingale, P. Prosser, C. Unsworth. *A Preliminary Review of Literature on Parallel Constraint Solving*. In Parallel Methods for Constraint Solving workshop at CP2011, pages 7–19, 2011.

Thank You!