Domenico Salvagnin – CPLEX Optimization – IBM Italy
06/10/2015

Recent MILP advances
CO@W2015

CPLEX TEAM

Andrea Lodi
MIP/MINLP

Andrea Tramontani
MIP

Bo Jensen
LP

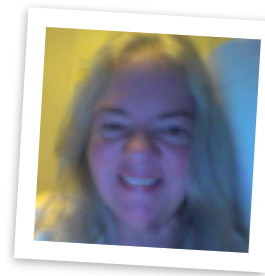
Christian Bliek
Barrier/(MI)Q(C)P

Daniel Junglas
MIP

Domenico Salvagnin
MIP

Ed Klotz
Product Expert

Jonas Schweiger
MINLP



Kathleen Callaway
Documentation

Laszlo Ladanyi
MIP

Philippe Charman
QA

Pierre Bonami
MIP/MINLP

Roland Wunderling
LP/MIP

Ryan Kersh
Python

Xavier Nodet
Product Manager

CPLEX Solver

- Problem types that can be solved:
  - LP: linear programs
  - QP: quadratic programs (convex objective)
  - QCP: quadratically constrained programs (convex constraints)
  - MIP: mixed integer linear programs
  - MIQP: mixed integer quadratic programs
  - MIQCP: mixed integer quadratically constrained programs
  - MINLP: mixed integer nonconvex quadratic objective programs
- Continuous algorithms (LP, QP, QCP):
  - primal simplex
  - dual simplex
  - barrier

CPLEX Features

- Tools to improve solving MIP models:
  - parameter tuning tool
    - find settings that work best for your problem class
  - polishing
    - improve primal bound
  - solution pool
    - get multiple solutions out of a solve
  - populate
    - get more and diverse solutions
  - rich callback API
    - try your research ideas inside state-of-the-art environment

- Tools to analyze MIP model issues:
  - FeasOpt (infeasible problems)
    - find minimal relaxation of problem to make it feasible
  - Conflict Refiner (infeasible problems)
    - extract a minimal infeasible subproblem to show the issue
  - Condition Number and MIP kappa (numerical issues)
    - estimate the ill-conditioning and numerics of an LP/MIP solve

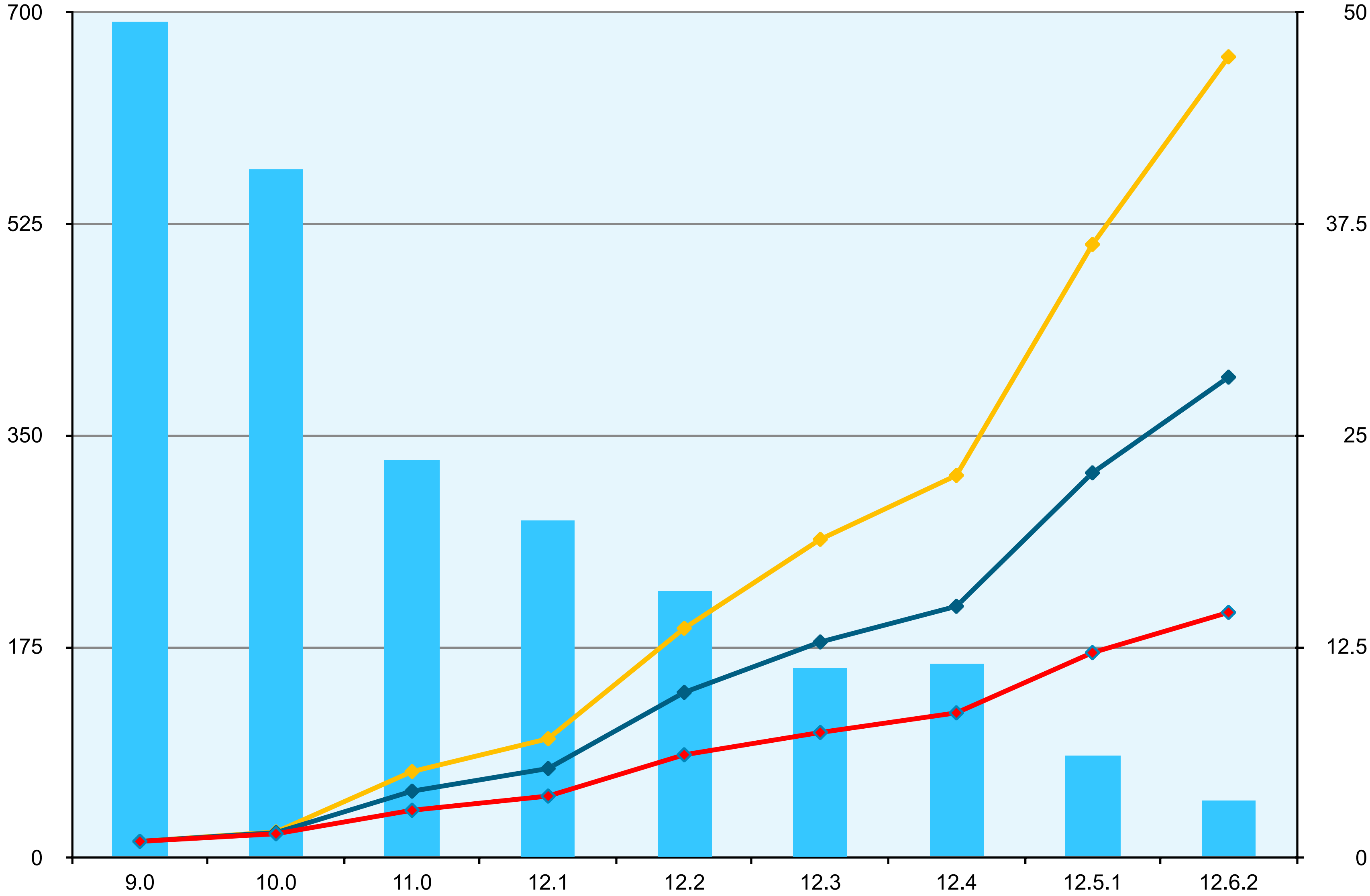
CPLEX Everywhere

- Available on many platforms:
  - Windows
  - Linux
  - Mac
  - zOS/AIX/Solaris/Sparc/HPux

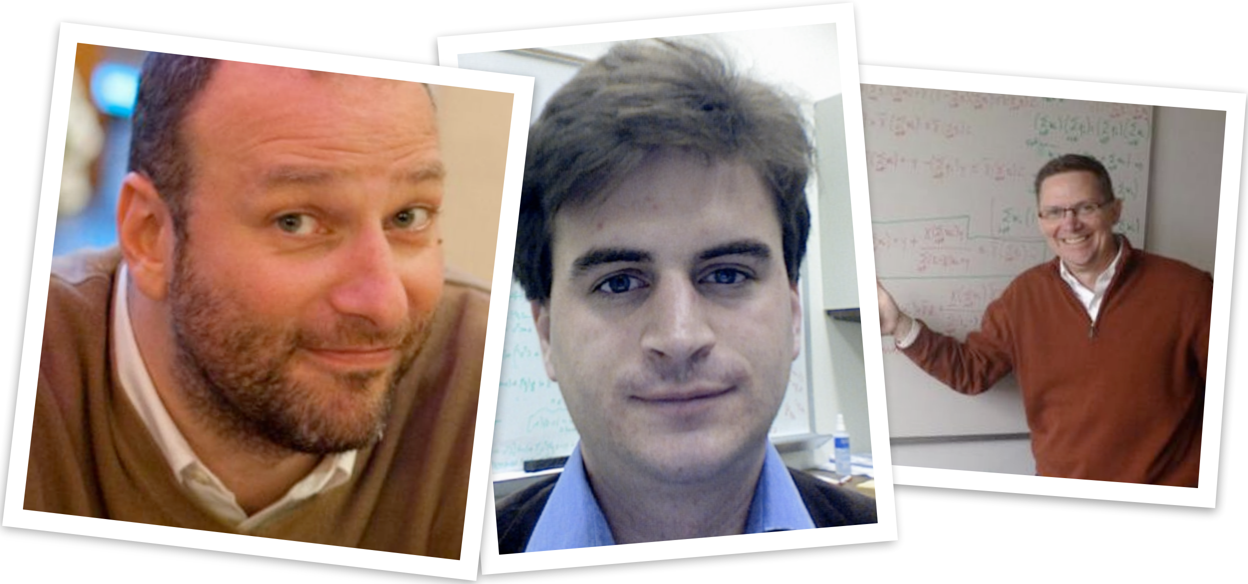
- Scalable to many scenarios:
  - Sequential
  - Shared-memory parallel
  - Distributed-memory parallel
  - Remote Object API
  - Cloud

see Daniel's talk

CPLEX keeps getting better everyday ☺



All the cool kids do MINLP these days…



…but this is supposed to be a MILP session!

(don't worry, let's just redefine "recent"…)

Recent advances in MILP

1) Parallel cut loop

2) Local implied bound cuts

3) Lift-and-project cuts

PARALLEL CUT LOOP

exploiting performance variability at root node

Performance Variability

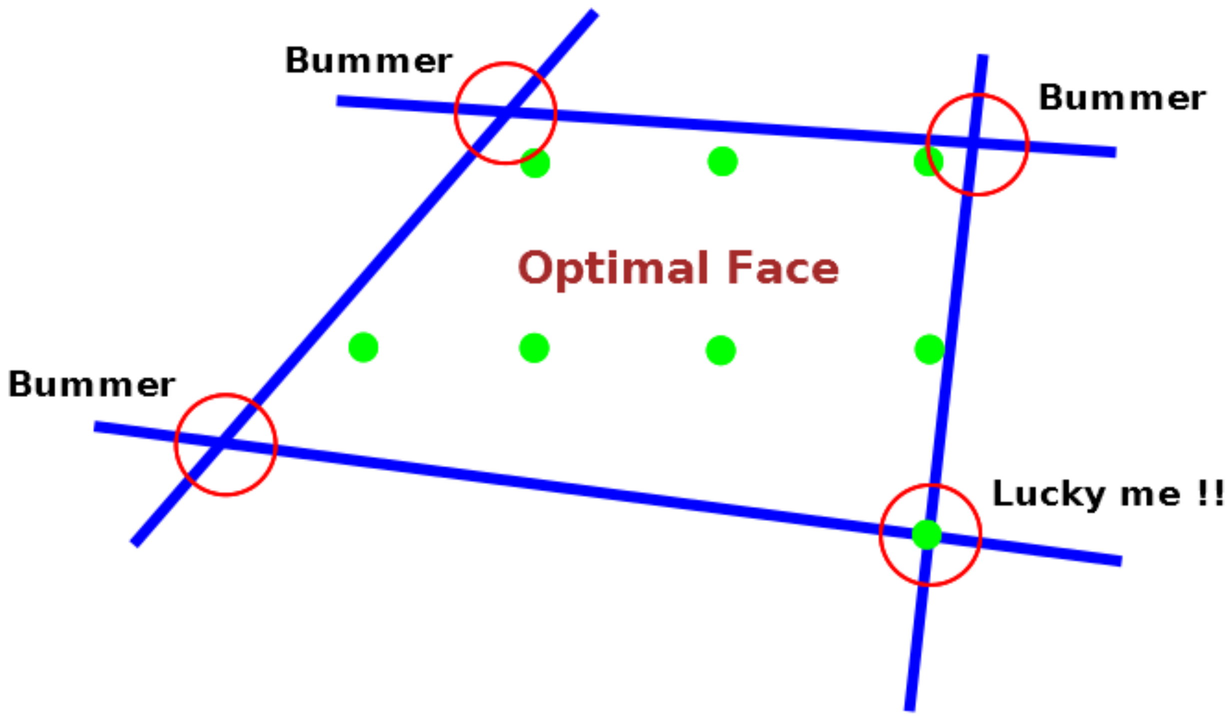
- Performance variability is a well known phenomenon **intrinsic** to MIP:
  - Danna (MIP Workshop, 2008)
  - Koch et al. (Math. Prog. Comp., 2011)
  - Achterberg and Wunderling (Facets of Comb. Optimization, 2013)
  - Lodi and Tramontani (INFORMS 2013 Tutorial)
  - Fischetti and Monaci (Op. Res., 2014)

- MIP Solvers contain various ingredients:
  - Heuristics
  - Cutting planes
  - Criteria for branching variable selection
  - ...

- Seemingly performance neutral changes (in the platform, in the code, or in the parameter setting) may have a big impact
  - on all those ingredients (separated cuts, heuristic solution found, picked variable to branch on)
  - and therefore on the whole solution process!

- A natural source of performance variability is the choice of the **initial LP basis**

- Because of **dual degeneracy**, there may be tons of equivalent (but **different**!) optimal solution/bases of the first LP relaxation



- Equivalent (but different) optimal solution/bases may lead to different:
  - cutting planes
  - heuristic solutions
  - ...
- Reoptimizing with different cutting planes amplifies the diversification
- The final root node could be very different in terms of:
  - Fractional solution
  - Dual and primal bounds
  - Cuts active in the LP
- Branching rules will take different decisions
- Branch-and-cut will then amplify the diversification again

Simulating Performance Variability: CPLEX random seed

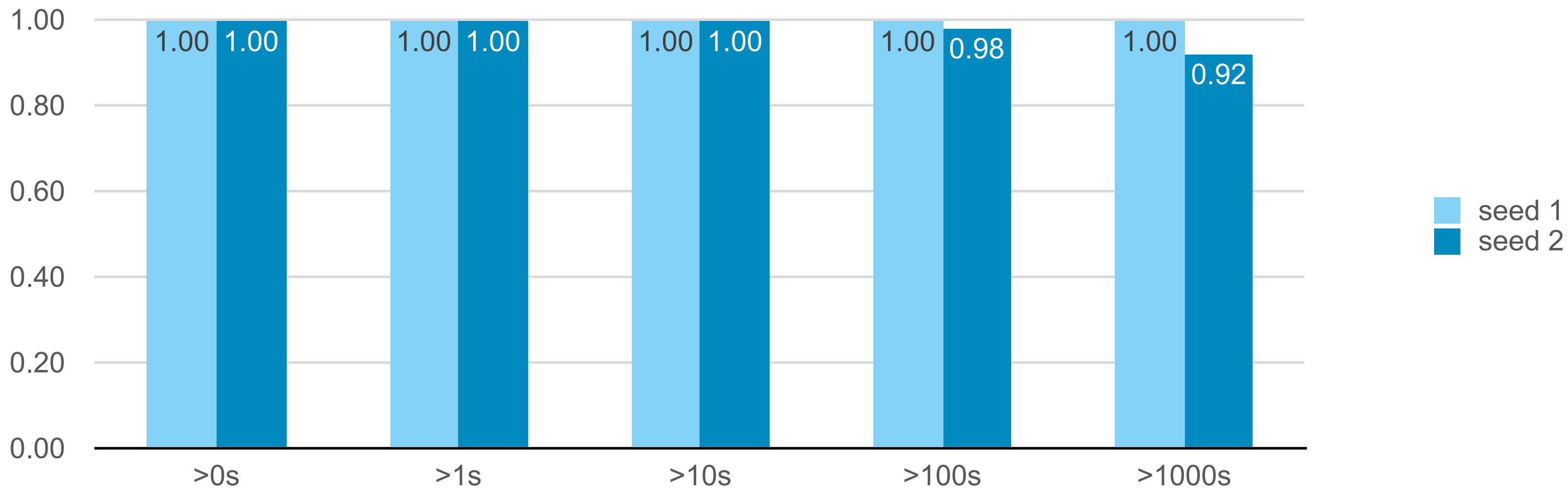
- Use CPLEX random seed parameter `CPXPARAM_RandomSeed` to simulate a performance neutral change in the environment (very convenient!)

- Compare:
  - Solver A: CPLEX 12.6.0 with random seed 1
  - Solver B: CPLEX 12.6.0 with random seed 2

- Test set: 3224 models classified according to their difficulty:
  - [0, 10k]: all models but the ones for which the solvers provide an inconsistent answer
  - [*T*, 10k]: all models for which one of the solver takes at least *T* seconds

- The comparison is fair because the difficulty of a given model is defined by all the compared solver: there is no bias against any of the compared solvers

- Comparison: (shifted) geometric mean of computing times



3084 models (75% affected)

1877 models (91% affected)

1112 models (95% affected)

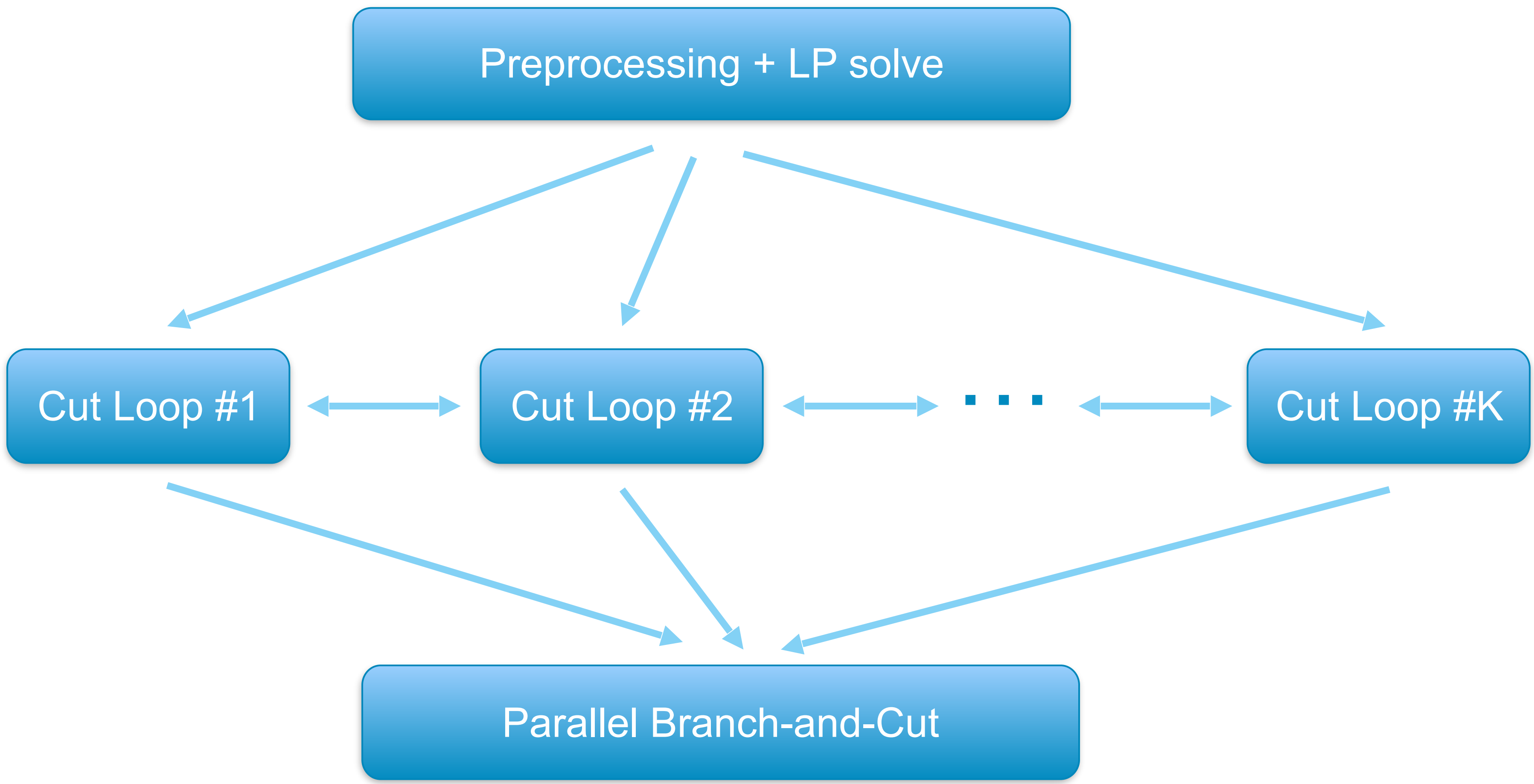
572 models (98% affected)

218 models (100% affected)

- 91% of the models in [1,10k] affected:
  - the random seed is effective to simulate performance variability.
- ~200 models in [1k,10k] **cannot** measure performance difference of less than 10%:
  - Small test sets are less robust to outliers,
  - Harder models are typically the ones exhibiting the larger variability.

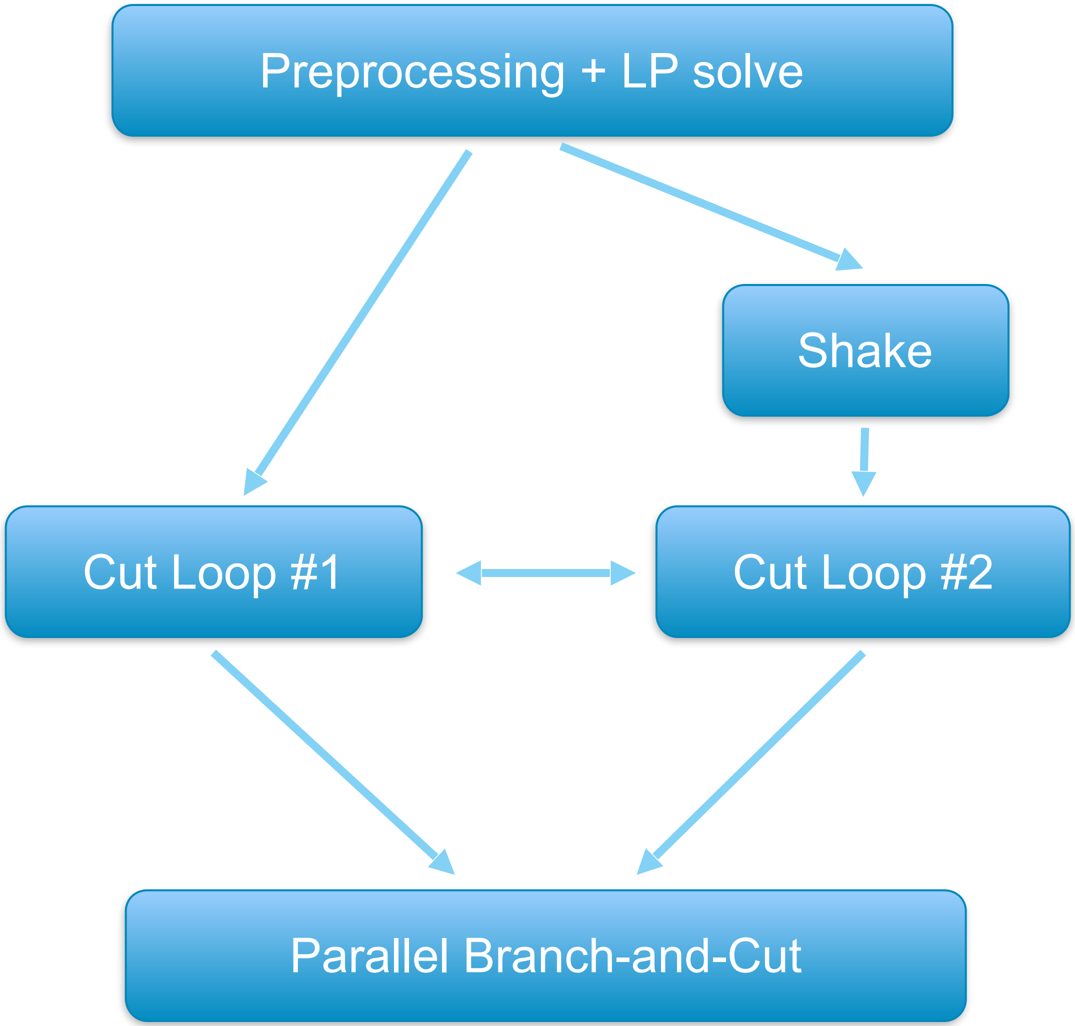
Can we take advantage of that?

Main Concept

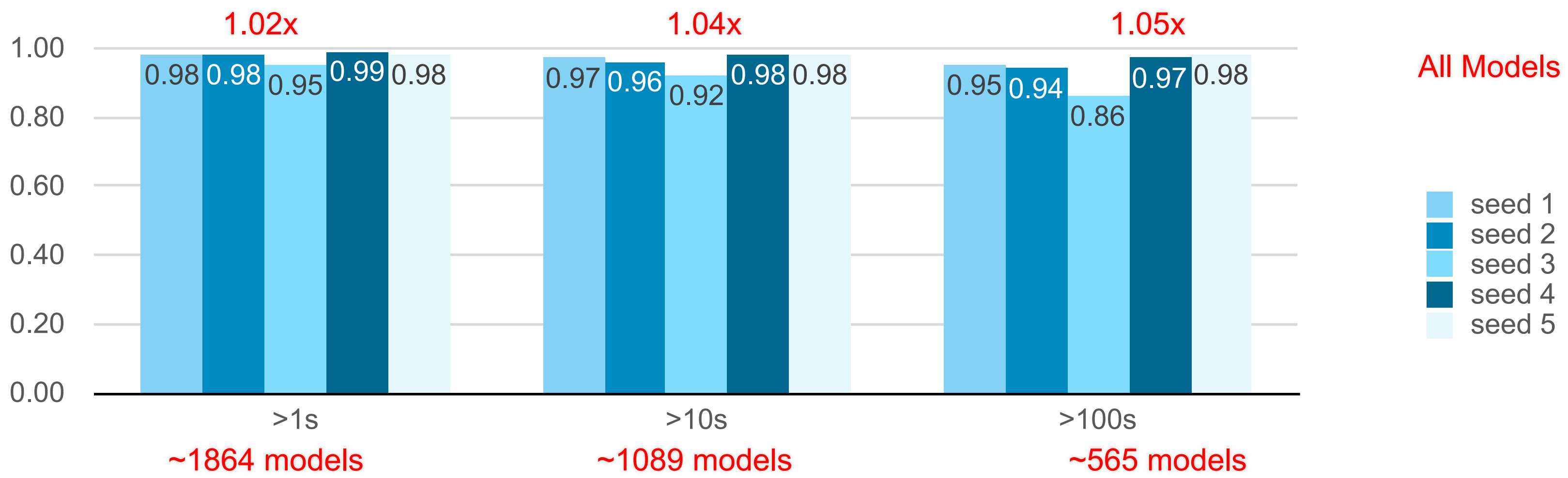


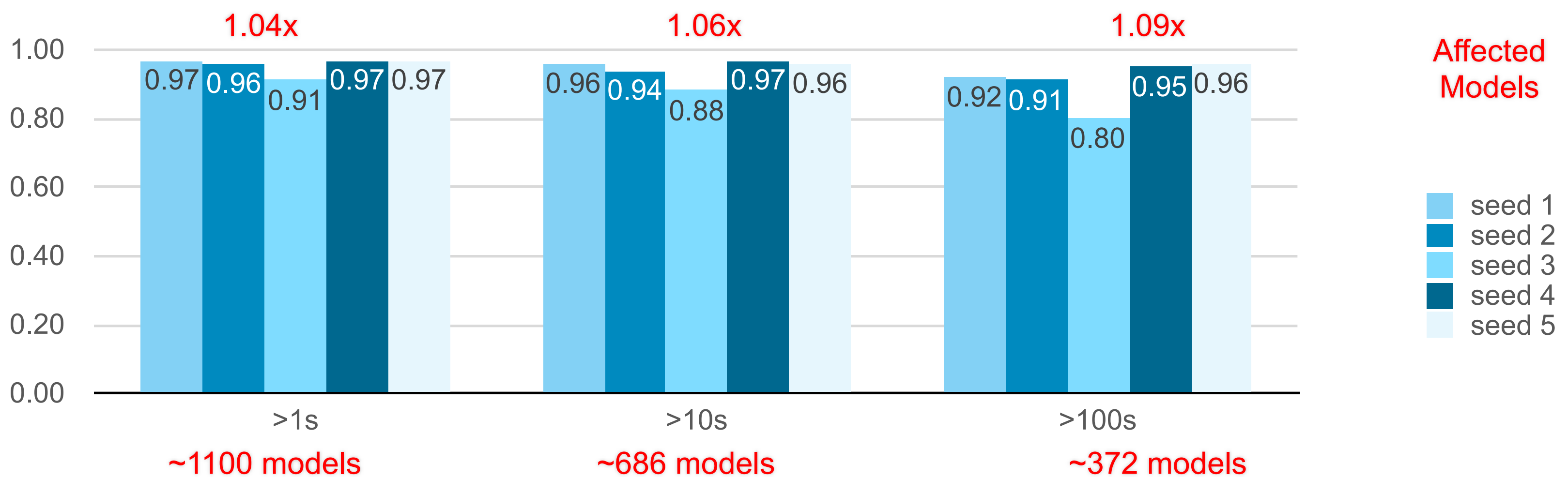
Implementation within CPLEX 12.5.1

- Forget about large K (we just use K = 2)
  - multi-threading is not for free
  - we also want to apply other methods at the root node
  - more cuts typically means better dual bound, but better dual bound does not mean less time to solve...
- Create diversification in the parallel cut loop by changing
  - the basis
  - the random seed
  - other parameters
- Be conservative with the cuts:
  - share cuts between the two cut loops
  - but be clever when selecting the cuts to be shared!


Preprocessing + LP solve
Shake
Cut Loop #1
Cut Loop #2
Parallel Branch-and-Cut

Parallel cut loop results (CPLEX 12.5.1)





LOCAL IMPLIED BOUND CUTS

Implications and implied bound cuts

$$z = 1 \rightarrow y \leq D$$

$$z \in \{0, 1\}, L \leq y \leq U, D < U$$

$$\Downarrow$$

$$y \leq U + (D - U)z$$

Discovered during presolve and probing

- **Global implied bound cuts**:
  - Globally valid implications
  - **Globally valid bounds** on implied non-binary variables
  - Separated at the root node and at branch-and-bound nodes

- **Local implied bound cuts (new in CPLEX 12.6.1):**
  - Globally valid implications
  - **Locally valid bounds** on implied non-binary variables
  - Separated at branch-and-bound **nodes only**

- **New parameter** `CPXPARAM_MIP_Cuts_LocalImplied` to control local implied bound cuts:
  - -1 = disabled
  - 0 = automatic (let CPLEX choose, currently equivalent to -1)
  - 1 = moderate: separated at starts of new dives, under conservative restrictions
  - 2 = aggressive: separated at starts of new dives, more often
  - 3 = very aggressive: potentially separated at every node

- Enabling moderate separation of local implied bound cuts:
  - 1-2% degradation (neutral?) on our test bed

- **Key points to be investigated**:
  - Cut strength versus cut applicability
    - Local cuts are stronger, but global cuts applies to the whole tree
  - Interaction with other cut types
  - Cut filtering and cut selection
    - Local cuts are violated more often than global cuts: special filtering rules?

- **Disabled by default, but fundamental in some cases** (e.g., to handle “big-M constraints”)

MIQP models from classification problems (Brooks, OR 2011):

$$
\begin{cases}
\min \frac{1}{2} w^T Q w + c^T y + 2c^T z \\
z_i = 0 \rightarrow K_i(w^T a_i + \alpha) + y_i \geq 1 \\
z_i \in \{0, 1\} \\
0 \leq y_i \leq 2 \\
-M \leq w_j \leq M \\
-M \leq \alpha \leq M
\end{cases}
$$

Key points:

- Implications are the **core of the problem**
- Very **large bounds** on α and w make any global cut completely ineffective.

Did you notice something odd in the previous model?

$$z_i = 0 \rightarrow K_i(w^T a_i + \alpha) + y_i \geq 1$$

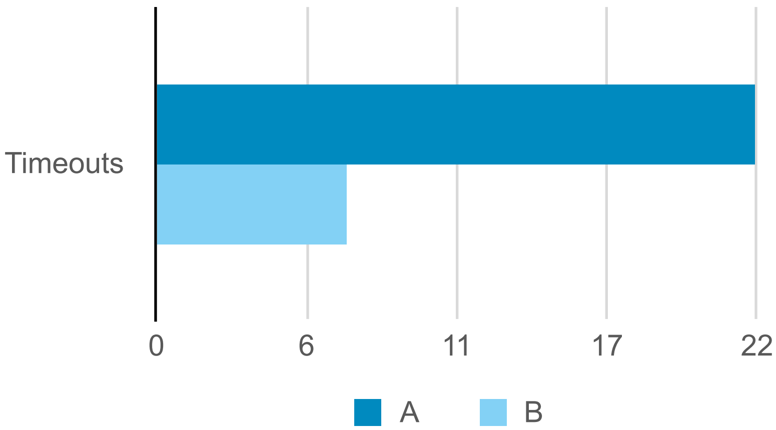
- Are these linear constraints?
- No, they are indicator constraints:
  - Advanced modeling feature natively supported by the CPLEX solver
  - Not just a modeling convenience
  - More numerically robust than their big-M counterpart and possibly more efficient as well

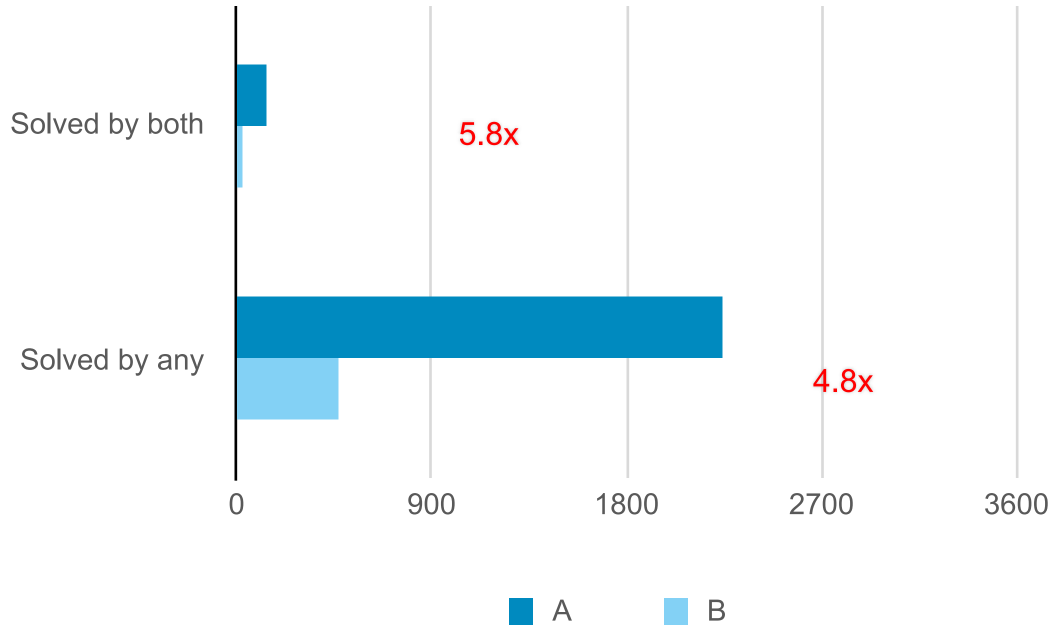
$$z_i = 0 \rightarrow K_i(w^T a_i + \alpha) + y_i \geq 1$$

$$\begin{cases} K_i(w^T a_i + \alpha) + y_i - s_i = 1 \\ s_i \geq -L_i z_i \end{cases}$$

- Bounds L_i are **just too big** and global implied bound cuts are totally ineffective
- Local implied bound cuts:
  - Two levels of branching immediately yield **reasonable local bounds** on α and ω_j variables
  - Local bounds L_i on s_i variables become tight
  - **Local implied bound cuts are effective**

Results on classification models





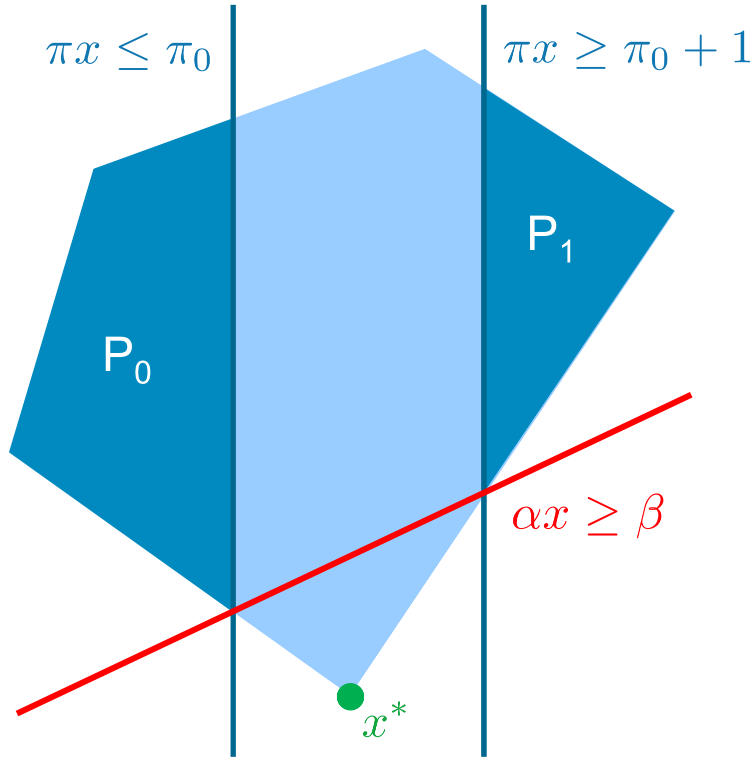
A: CPLEX 12.6.1 default
B: CPLEX 12.6.1 + aggressive local implied bound cuts

LIFT & PROJECT CUTS

CGLP magic

CGLP basics

$$(u) Ax = b$$
$$(u_0) - \pi x \geq -\pi_0$$
$$x \geq 0$$



$$(v) Ax = b$$
$$(v_0) \pi x \geq \pi_0 + 1$$
$$x \geq 0$$

$$\begin{cases} \min \boxed{\alpha x^* - \beta} \\ \alpha \geq uA - u_0\pi \\ \beta \leq ub - u_0\pi_0 \\ \alpha \geq vA + v_0\pi \\ \beta \leq vb + v_0(\pi_0 + 1) \\ u_0, v_0 \geq 0 \end{cases}$$

maximize violation

cut valid for P_0

cut valid for P_1

- All violated inequalities make the CGLP unbounded.
- By construction can only distinguish between violated and not-violated inequalities…
- Normalization is crucial!!!
- Twice the size of the original LP…

Lift and Project Cuts à la Pierre (Math. Prog. Comp., 2012)

$$\begin{cases} \min \alpha x^* - \beta \\ \alpha \geq uA - u_0\pi \\ \beta \leq ub - u_0\pi_0 \\ \alpha \geq vA + v_0\pi \\ \beta \leq vb + v_0(\pi_0 + 1) \\ u_0, v_0 \geq 0 \end{cases}$$

$$u_0 + v_0 = 1$$

$$d^* = \pi x^* - \pi_0$$

duality

$$\begin{cases} \max \pi y - (\pi_0 + 1) \\ \boxed{Ay = b} \\ 0 \leq \boxed{y \leq x^*/d^*} \end{cases}$$

Same size and structure of original problem

*Automatically works on the support of x**

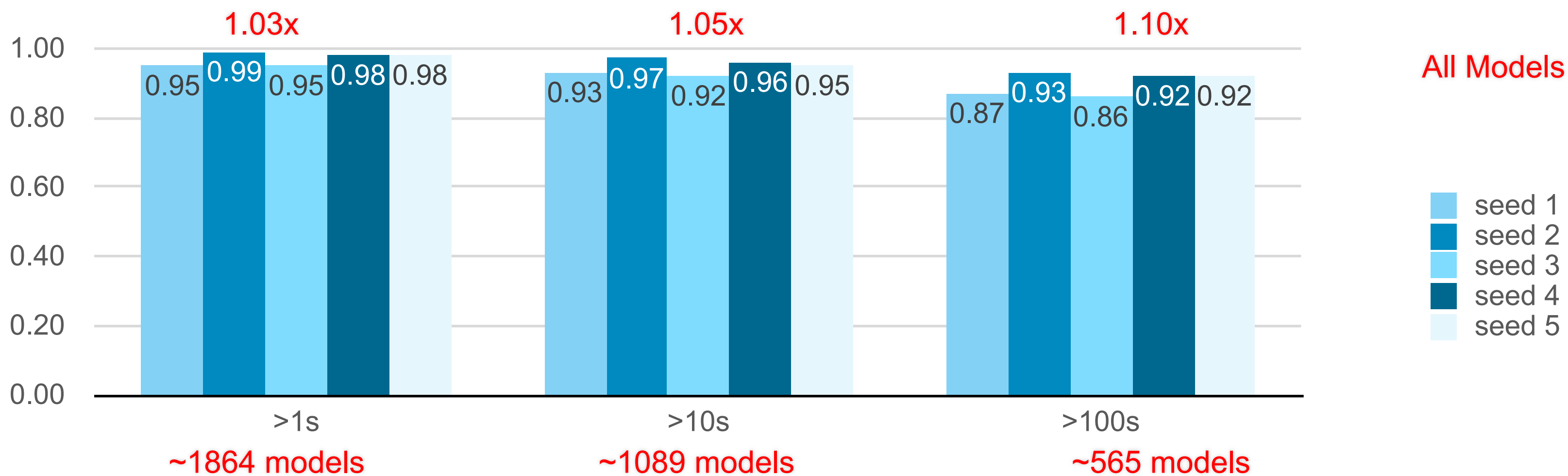
Lift and Project Cuts in CPLEX 12.5.1

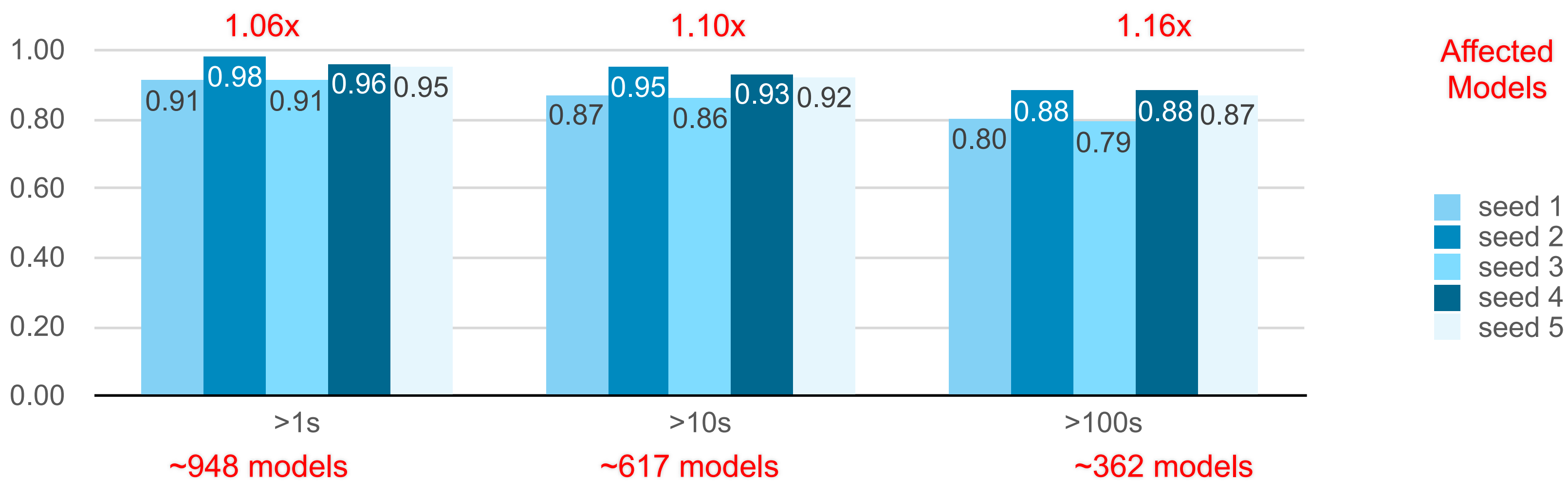
- Pierre's CGLP to separate **rank-1 cuts** only. Advantages:
  - Tends to produce cleaner and sparser cuts
  - Enforces the separation of low-rank inequalities (not a big surprise)
  - Breaks the correlation between optimization and separation

- Plus **some tricks**:
  - Heuristic reduction of CGLP to
    - Speed up the separation
    - Get rid of constraints you don't want to use to produce the cut
  - Clever selection of the disjunctions to use

- Not explored but worthy to mention:
  - Rank-1 heuristics: Dash and Goycoolea (Math. Prog. Comp., 2010)
  - Relax-and-cut: Fischetti and S. (Math. Prog. Comp., 2011)

L&P results (CPLEX 12.5.1)





Questions?

IBM®

Legal Disclaimer

- The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
- References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
- Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.