CPLEX Optimization – IBM Germany
2015-10-06

# Using computing resources with IBM ILOG CPLEX

## CO@W2015

- Hardware resources
  - Multiple cores/threads
  - Multiple machines
  - No machines
- Software resources
  - Interfacing with CPLEX
  - Interacting with Python

# Using more than one core/thread to solve a problem

# Sequential branch and bound

Search tree

T

min $c^T x$
$Ax = b$
$x_j$ in {0, 1}

Pseudo code:

```
while !T.is_empty
  n = T.get_next()
  n.solve()
  if not (n.is_integer() ||
          n.is_infeasible())
    j = n.fractional_index()
    v = n.fractional_value(j)
    T.create(n + “x[j] <= floor(v)”)
    T.create(n + “x[j] >= ceil(v)”)
```

# Sequential branch and bound

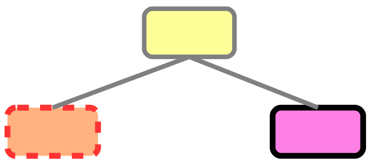
1.

```
while !T.is_empty
  n = T.get_next()
  n.solve()
  if not (n.is_integer() ||
          n.is_infeasible())
    j = n.fractional_index()
    v = n.fractional_value(j)
    T.create(n + “x[j] <= floor(v)”)
    T.create(n + “x[j] >= ceil(v)”)
```

# Sequential branch and bound

1.

2.



```
while !T.is_empty
  n = T.get_next()
  n.solve()
  if not (n.is_integer() ||
          n.is_infeasible())
    j = n.fractional_index()
    v = n.fractional_value(j)
    T.create(n + “x[j] <= floor(v)”)
    T.create(n + “x[j] >= ceil(v)”)
```

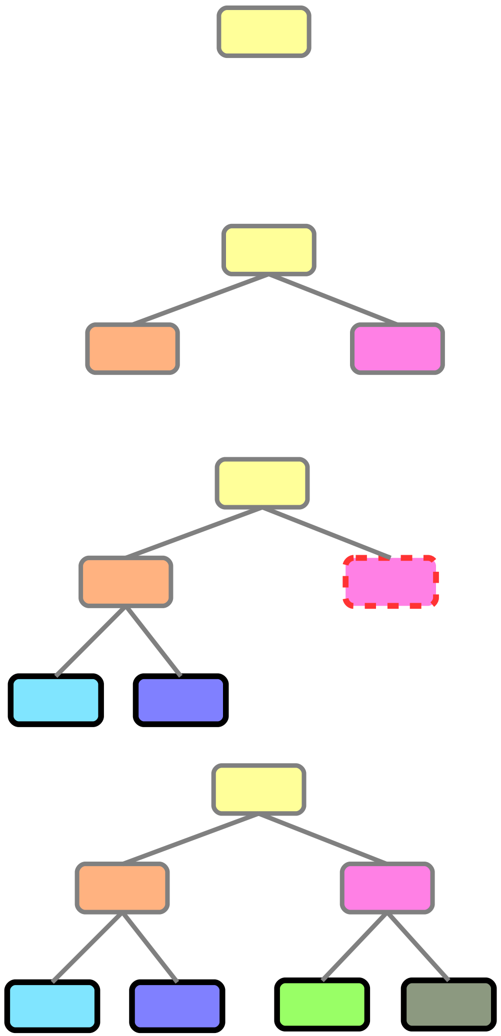
# Sequential branch and bound

1.

2.

3.

4.



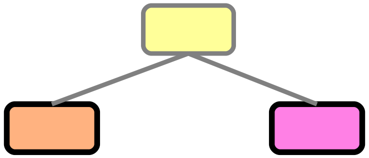
```
while !T.is_empty
  n = T.get_next()
  n.solve()
  if not (n.is_integer() ||
              n.is_infeasible())
    j = n.fractional_index()
    v = n.fractional_value(j)
    T.create(n + “x[j] <= floor(v)”)
    T.create(n + “x[j] >= ceil(v)”)
```

# Parallel branch and bound

```
while !T.is_empty
  T.lock()
  n = T.get_next()
  T.unlock()
  n.solve()
  if not (n.is_integer() ||
          n.is_infeasible())
    j = n.fractional_index()
    v = n.fractional_value(j)
    T.lock()
    T.create(n + “x[j] <= floor(v)”)
    T.create(n + “x[j] >= ceil(v)”)
    T.unlock()
```

# Parallel branch and bound

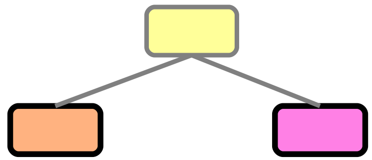
Thread 1



Thread 2

# Parallel branch and bound

Thread 1

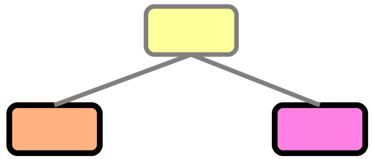


Thread 2

T.lock()

T.lock()

# Parallel branch and bound



## Thread 1

T.lock()

T.get_next()

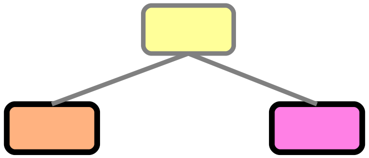
T.unlock()

## Thread 2

T.lock()

**suspended**

# Parallel branch and bound



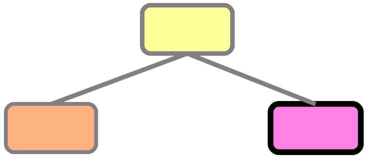
## Thread 1

T.lock()

T.get_next() 

T.unlock()

solve



## Thread 2

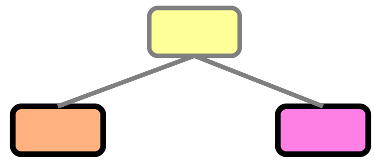
T.lock()

**suspended**

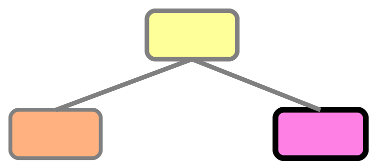
T.get_next() 

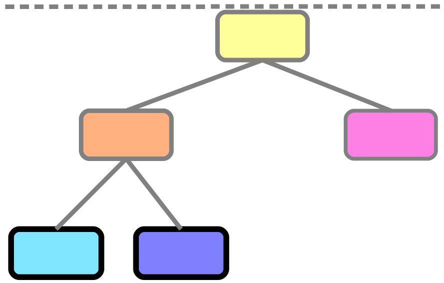
T.unlock()

# Parallel branch and bound



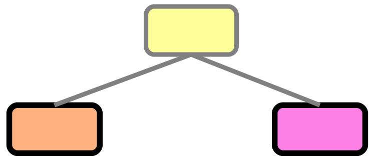
| Thread 1 | Thread 2 |
|---|---|
| T.lock() | T.lock() |
| T.get_next()<br>T.unlock() | **suspended** |
| solve | T.get_next()<br>T.unlock() |
| T.lock()<br>T.create()<br>T.unlock() | solve<br>T.lock()<br>**suspended** |

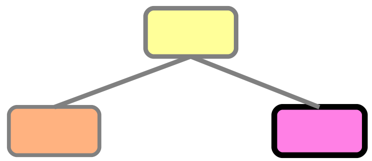


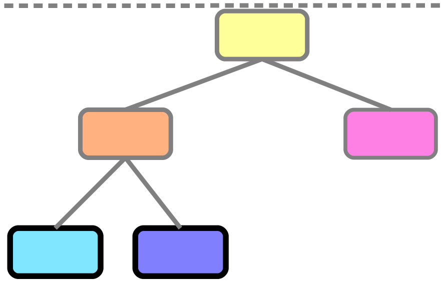


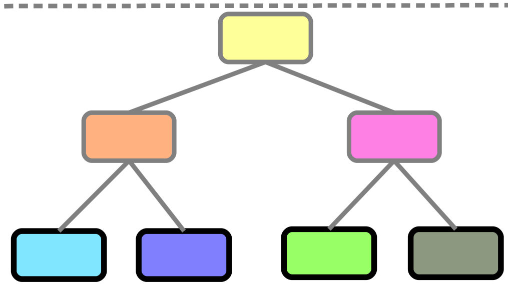
# Parallel branch and bound

| Thread 1 | Thread 2 |
| --- | --- |
| T.lock() | T.lock() |
| T.get_next()<br>T.unlock() | **suspended** |
| solve | T.get_next()<br>T.unlock() |
| T.lock()<br>T.create()<br>T.unlock() | solve<br>T.lock()<br>**suspended** |
| | T.create()<br>T.unlock() |

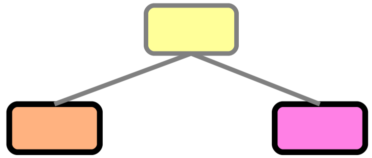




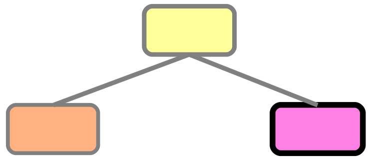


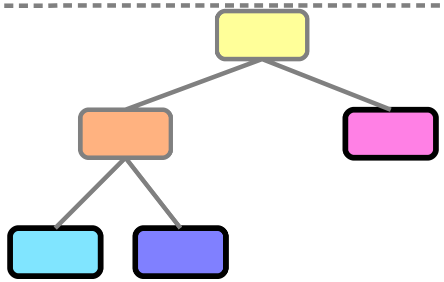


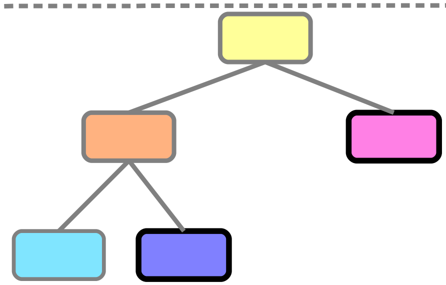
# Not deterministic!



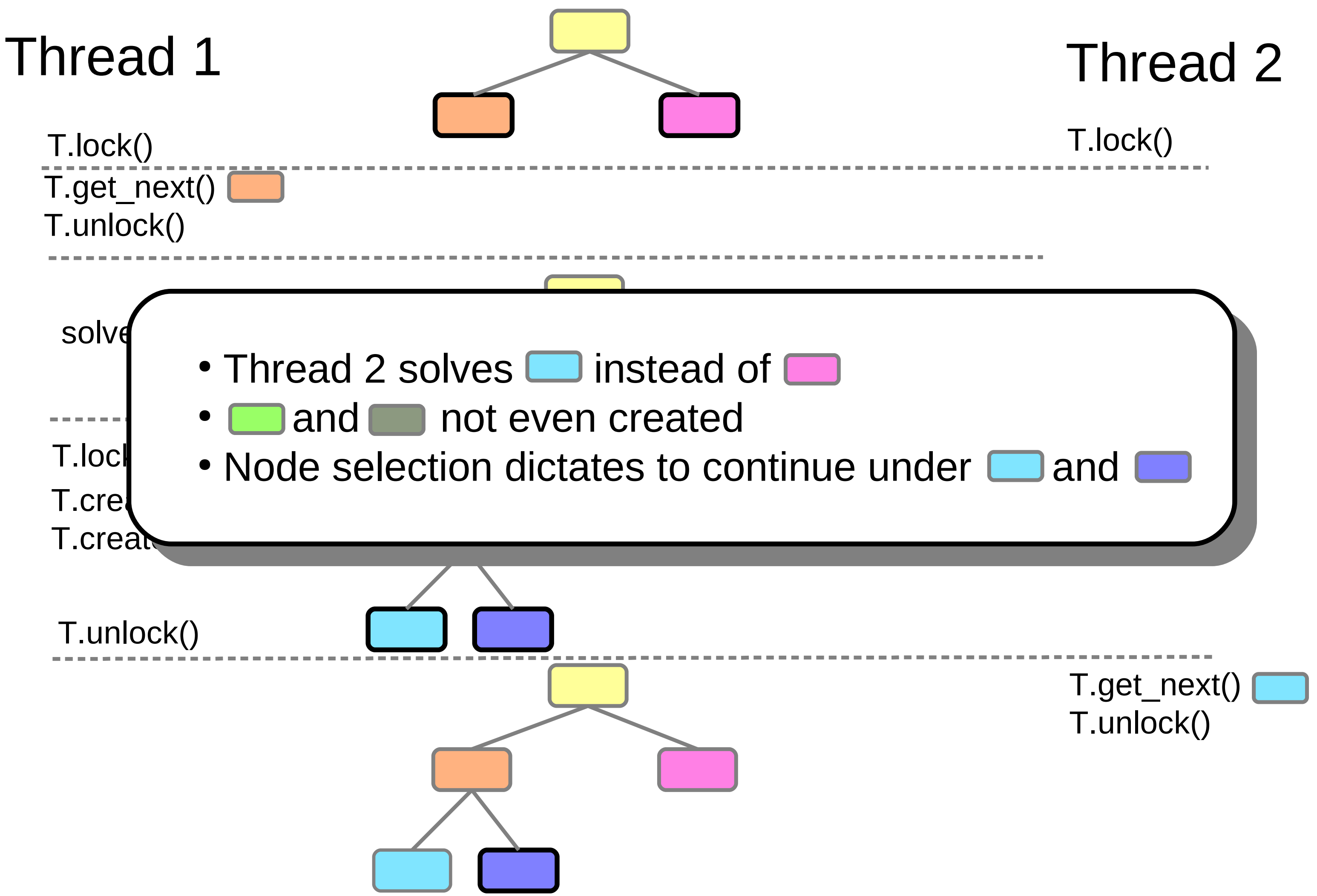
| Thread 1 | Thread 2 |
| --- | --- |
| T.lock() | T.lock() |
| T.get_next()<br>T.unlock() | |
| solve | **suspended** |
| T.lock()<br>T.create()<br>T.unlock() | |
| | T.get_next()<br>T.unlock() |







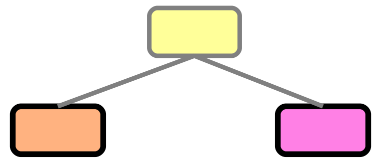
# Not deterministic!



# Not deterministic!

Thread 1

Thread 2



T.lock()

T.get_next()
T.unlock()

T.lock()

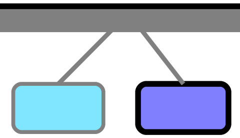
solve

T.lock
T.crea
T.crea

T.unl

- Thread 2 solves instead of
- and not even created
- Node selection dictates to continue under and

Assume must be processed to find optimal solution
Assume processing is enough to prove optimality
→ very different node counts, although same problem, code, hardware



# Issues with non-determinism

- opportunistic behavior of OS scheduler
- very minor side effects (cache misses, page swaps, …) can change order of threads

This is a problem in practice:

| gmu-35-40 | | |
|---|---|---|
| Time | Iterations | Nodes |
| 21.31 | 2236199 | 444838 |
| 18.46 | 2060789 | 322282 |
| **4.62** | **253871** | **51778** |
| 31.55 | 3300767 | 509641 |
| 11.89 | 1238162 | 177236 |
| **79.50** | **7212334** | **1307340** |
| 21.10 | 2955104 | 595939 |
| 16.27 | 1953450 | 401507 |
| 13.61 | 1410170 | 261508 |
| 17.10 | 1948228 | 410949 |

| iis-pima-cov | | |
|---|---|---|
| Time | Iterations | Nodes |
| **62.92** | **1678290** | **17744** |
| 139.72 | 5475373 | 47428 |
| 84.39 | 2075243 | 26460 |
| 80.37 | 2766163 | 22016 |
| 71.25 | 1753092 | 17513 |
| 69.18 | 1615592 | 21636 |
| 117.43 | 3002680 | 40299 |
| 72.41 | 1704471 | 21301 |
| 124.01 | 4863607 | 42298 |
| **191.31** | **7898071** | **71043** |

| glass4 | | |
|---|---|---|
| Time | Iterations | Nodes |
| 41.60 | 6577338 | 340192 |
| 70.57 | 11800428 | 637471 |
| **29.30** | **4075955** | **304913** |
| 34.31 | 6026804 | 208466 |
| 160.12 | 29207354 | 1082067 |
| **244.62** | **35189917** | **1893985** |
| 54.40 | 8700758 | 512053 |
| 74.64 | 11307243 | 698508 |
| 47.19 | 8600157 | 389436 |
| 54.42 | 8275014 | 544172 |

# Issues with non-determinism

- opportunistic behavior of OS scheduler
- very minor side effects (cache misses, page swaps, …) can change order of threads

This is a problem in practice:

| gmu-35-40 | | |
|---|---|---|
| Time | Iterations | Nodes |
| **4.62** | **253871** | **51778** |
| **79.50** | **7212334** | **1307340** |

| iis-pima-cov | | |
|---|---|---|
| Time | Iterations | Nodes |
| **62.92** | **1678290** | **17744** |
| **191.31** | **7898071** | **71043** |

| glass4 | | |
|---|---|---|
| Time | Iterations | Nodes |
| **29.30** | **4075955** | **304913** |
| **244.62** | **35189917** | **1893985** |

- → results are not repeatable
- → debugging becomes painful
- → benchmarking is complicated
- → assessment of algorithmic changes is difficult
  “Is my new cut really helpful or is this just a random change?”

**→ We need a solver that operates in a <u>deterministic</u> way!**

# Deterministic multi-threading

opportunistic — OS scheduler, cache misses, swaps, ...
→ order in which lock is granted to threads is not deterministic

deterministic — make this order deterministic:

- use **deterministic time** → same in any run
- grant lock to first thread to arrive in **deterministic time**

# Deterministic multi-threading

| | |
|---|---|
| opportunistic | OS scheduler, cache misses, swaps, ...<br>→ order in which lock is granted to threads is not deterministic |
| deterministic | make this order deterministic:<br>• use **deterministic time** → same in any run<br>• grant lock to first thread to arrive in **deterministic time** |
| Kendo | “Kendo: Efficient Deterministic Multithreading in Software”<br>http://people.csail.mit.edu/mareko/asplos073-olszewski.pdf<br>→ use number of retired store instructions as time (x86) |

# Deterministic multi-threading

opportunistic
OS scheduler, cache misses, swaps, ...
→ order in which lock is granted to threads is not deterministic

deterministic
make this order deterministic:
- use **deterministic time** → same in any run
- grant lock to first thread to arrive in **deterministic time**

Kendo
"Kendo: Efficient Deterministic Multithreading in Software"
http://people.csail.mit.edu/mareko/asplos073-olszewski.pdf
→ use number of retired store instructions as time (x86)

CPLEX
deterministic time = number of array accesses
- ➔ works the same on all hardware
- ➔ explicit instrumentation of source code

```
for (int i = 0; i < n; ++i)
   x[i] = y[i] * 0.5;
DETCLOCK_INDEX_ARRAY (2 * i);
```

# Deterministic multi-threading

- Deterministic threading is not for free
  - settle for a particular path through the search tree
  - increases wait/idle times in locks

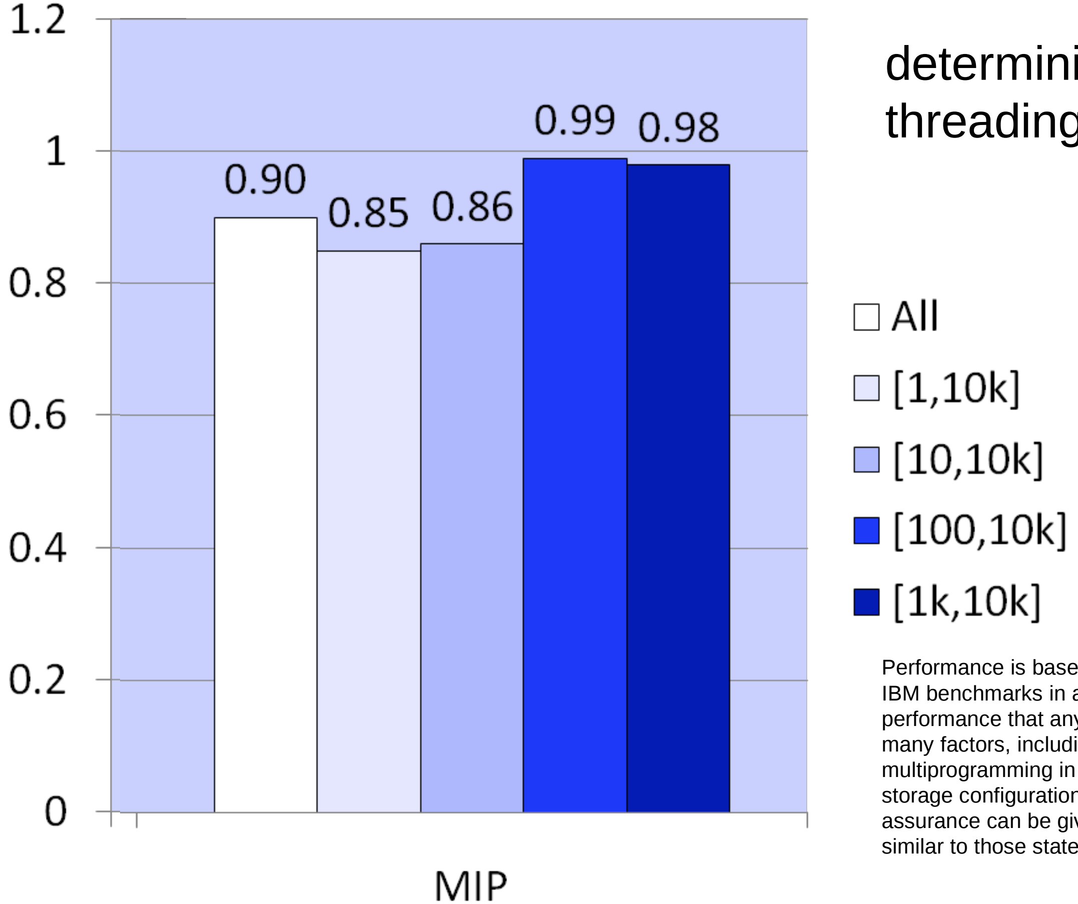
| gmu-35-40 | | |
|---|---|---|
| Time | Iterations | Nodes |
| 4.62 | 253871 | 51778 |
| 79.50 | 7212334 | 1307340 |
| **38.48** | **4124364** | **737994** |

| iis-pima-cov | | |
|---|---|---|
| Time | Iterations | Nodes |
| 62.92 | 1678290 | 17744 |
| 191.31 | 7898071 | 71043 |
| **72.84** | **1382605** | **20449** |

| glass4 | | |
|---|---|---|
| Time | Iterations | Nodes |
| 29.30 | 4075955 | 304913 |
| 244.62 | 35189917 | 1893985 |
| **91.68** | **11737409** | **528160** |

# Deterministic multi-threading

- Deterministic threading is not for free
  - settle for a particular path through the search tree
  - increases wait/idle times in locks




deterministic vs. opportunistic threading

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.

# Deterministic multi-threading

- Deterministic threading is not for free
  - settle for a particular path through the search tree
  - increases wait/idle times in locks

- Deterministic threading is unrelated to performance variability
  - parallel cut loop is still meaningful

- CPLEX extensions of Kendo framework allow efficient parallel cutloop
  - → opportunistic code, but results are deterministic

- All parallel algorithms in CPLEX are available as opportunistc/deterministic
  - → select at runtime with a parameter

# Using more than one machine to solve a problem

# Distributed parallel MIP

CPLEX implements distributed parallel MIP solving

- one master process to coordinate the search
- several worker processes to perform the heavy lifting
- usually one master/worker per machine (can configure otherwise)
- communication only between master and workers

# Distributed parallel MIP

CPLEX implements distributed parallel MIP solving

- one master process to coordinate the search
- several worker processes to perform the heavy lifting
- usually one master/worker per machine (can configure otherwise)
- communication only between master and workers
- 3 different ways of communication:

`ssh` → master starts workers via ssh
→ communication via pipes

`TCP/IP` → workers run a server-like process to which master connects
→ communication via sockets

`MPI` → all processes run within an MPI communicator
→ communication via MPI functions

# Distributed parallel MIP

CPLEX implements distributed parallel MIP solving

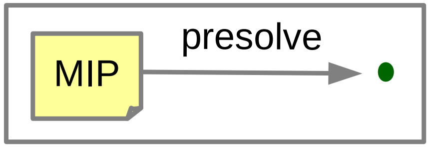
- one master process to coordinate the search
- several worker processes to perform the heavy lifting
- usually one master/worker per machine (can configure otherwise)
- communication only between master and workers
- 3 different ways of communication:

| | |
|---|---|
| `ssh` | → master starts workers via ssh<br>→ communication via pipes |
| `TCP/IP` | → workers run a server-like process to which master connects<br>→ communication via sockets |
| `MPI` | → all processes run within an MPI communicator<br>→ communication via MPI functions |

- 2 phases:
  1. “rampup” → create an initial set of open nodes
  2. “tree search” → perform distributed parallel b&b

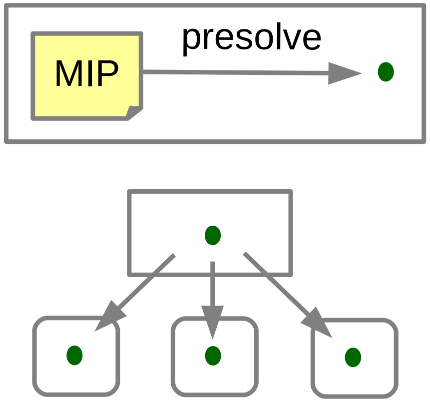
# Phase I: rampup

1. master runs presolve to create a root node



# Phase I: rampup

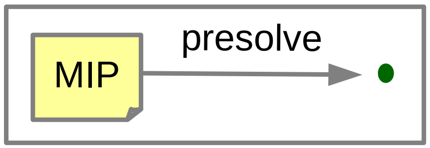
1. master runs presolve to create a root node



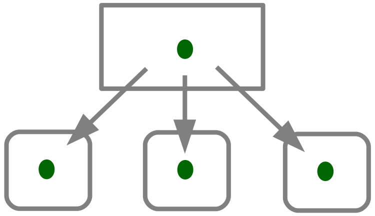

2. master sends root to workers

# Phase I: rampup

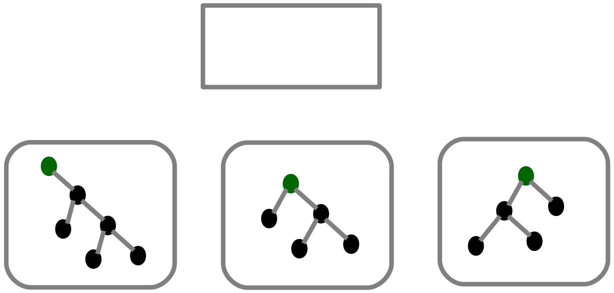
1. master runs presolve to create a root node




2. master sends root to workers

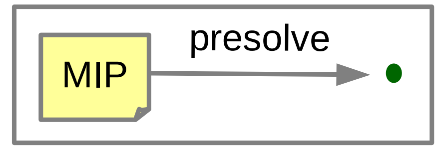


3. each worker starts b&b with different parameters
→ each worker produces a different tree

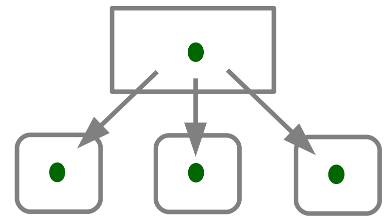


# Phase I: rampup

1. master runs presolve to create a root node

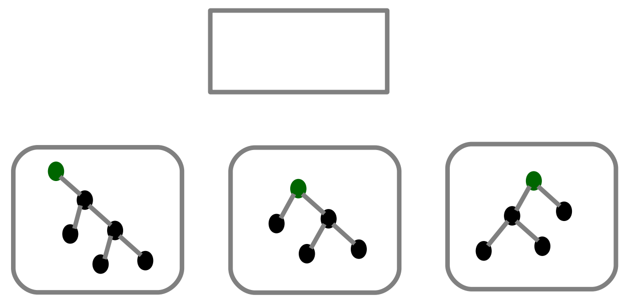



2. master sends root to workers



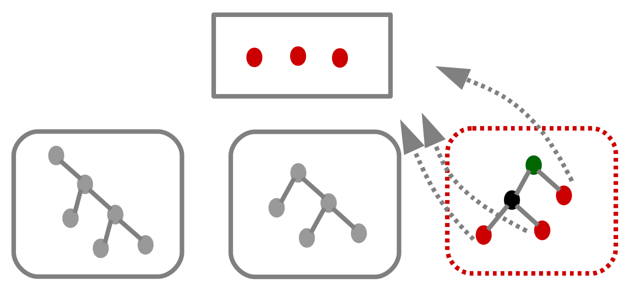
3. each worker starts b&b with different parameters

→ each worker produces a different tree



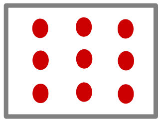
4. master eventually
   - stops workers
   - selects a winner
   - collects open nodes from winner

   → list of “supernodes”



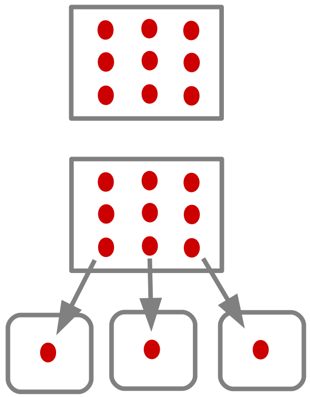
# Phase II: tree search

5. master starts with list of supernodes



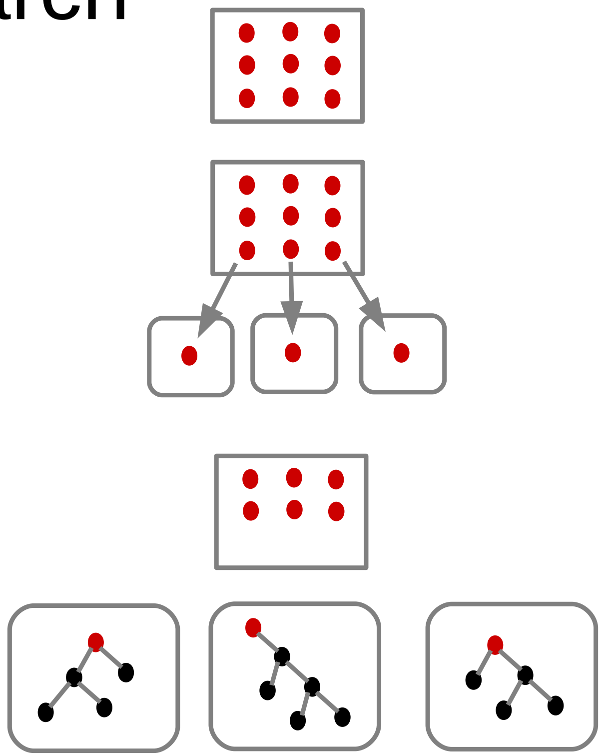
# Phase II: tree search

5. master starts with list of supernodes
6. master sends a supernode to each worker



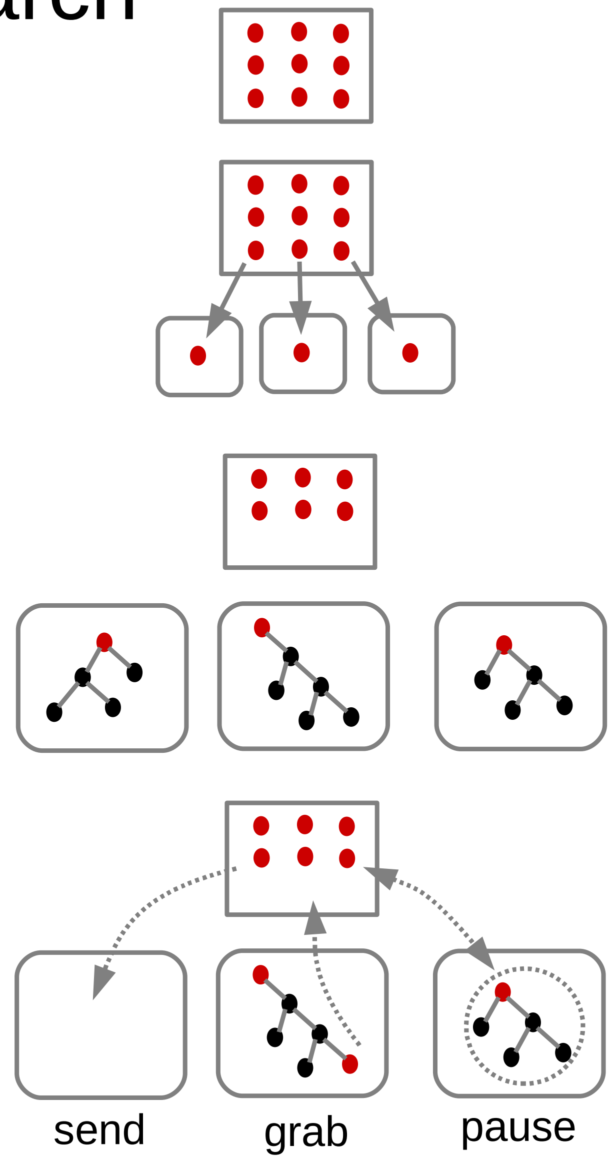
# Phase II: tree search

5. master starts with list of supernodes
6. master sends a supernode to each worker
7. workers solve supernode as MIP



# Phase II: tree search

5. master starts with list of supernodes
6. master sends a supernode to each worker
7. workers solve supernode as MIP
8. Master can
   - send new supernodes (if idle)
   - grab nodes to produce new supernodes
   - pause supernode (exchange)



# Distributed parallel MIP

Variants/improvements

1. exchange information (incumbents, bound tightenings, ...)

# Distributed parallel MIP

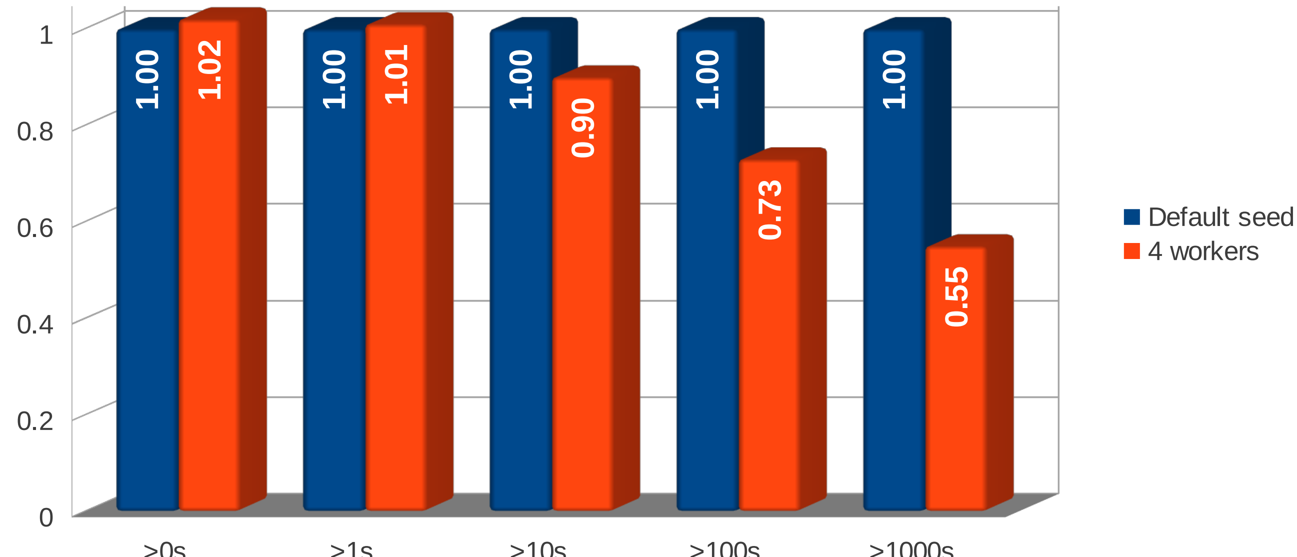
Variants/improvements

1. exchange information (incumbents, bound tightenings, ...)
2. in rampup, start some workers with special settings
   - aggressive heuristics → quickly find good solutions
   - aggressive cuts → quickly improve dual bound
   - ...

# Distributed parallel MIP

Variants/improvements

1. exchange information (incumbents, bound tightenings, ...)
2. in rampup, start some workers with special settings
   - aggressive heuristics → quickly find good solutions
   - aggressive cuts → quickly improve dual bound
   - ...
3. never stop the rampup phase → exploit performance variability




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.

# Solving a problem without a machine

# CPLEX in the cloud

What if no local resources to solve model?

- model too hard
- only need to solve once in a while
- ...

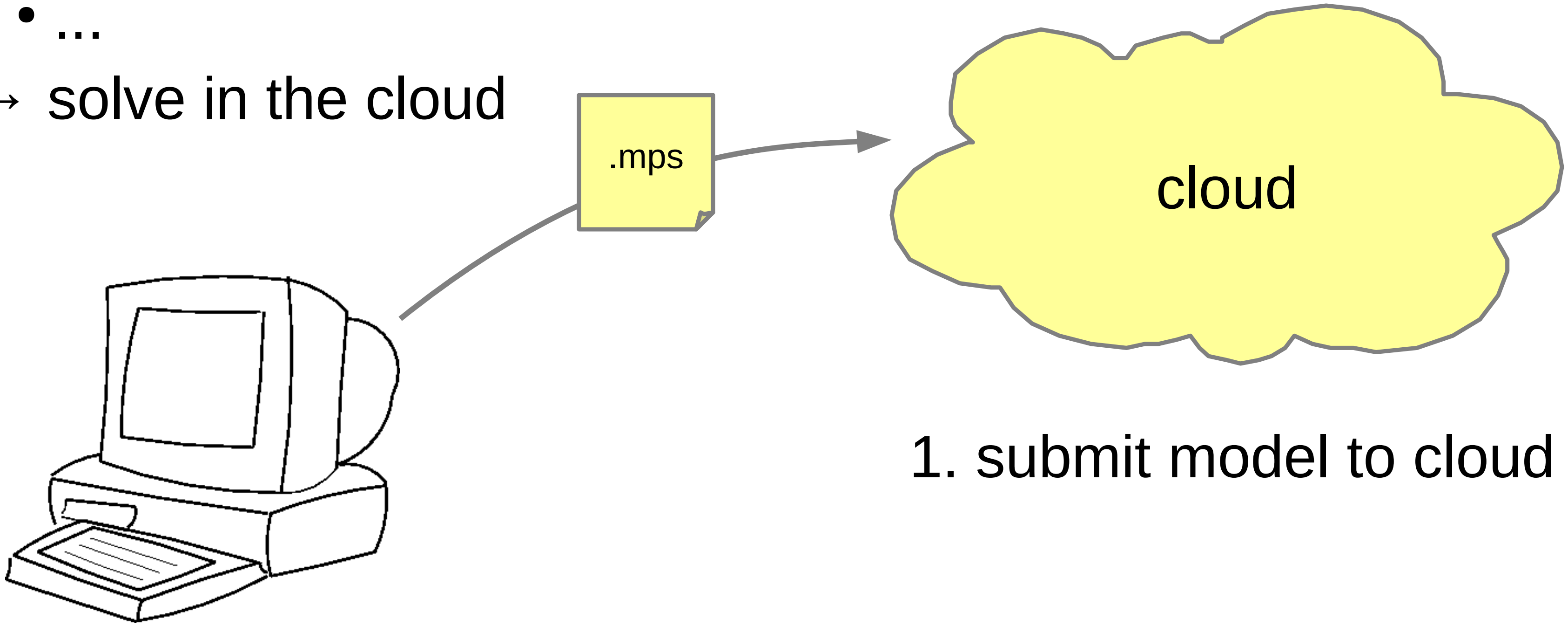
→ solve in the cloud

# CPLEX in the cloud

What if no local resources to solve model?

- model too hard
- only need to solve once in a while
- ...

→ solve in the cloud



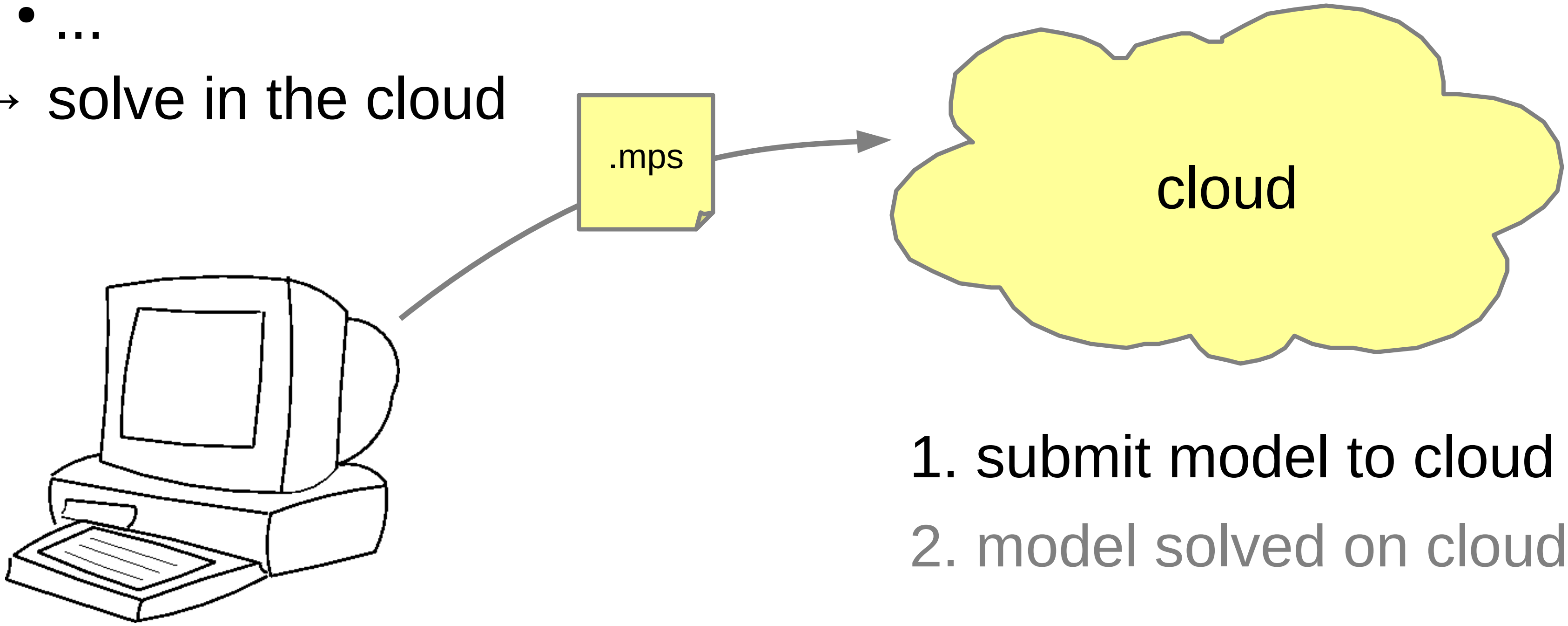

1. submit model to cloud

# CPLEX in the cloud

What if no local resources to solve model?

- model too hard
- only need to solve once in a while
- ...

→ solve in the cloud



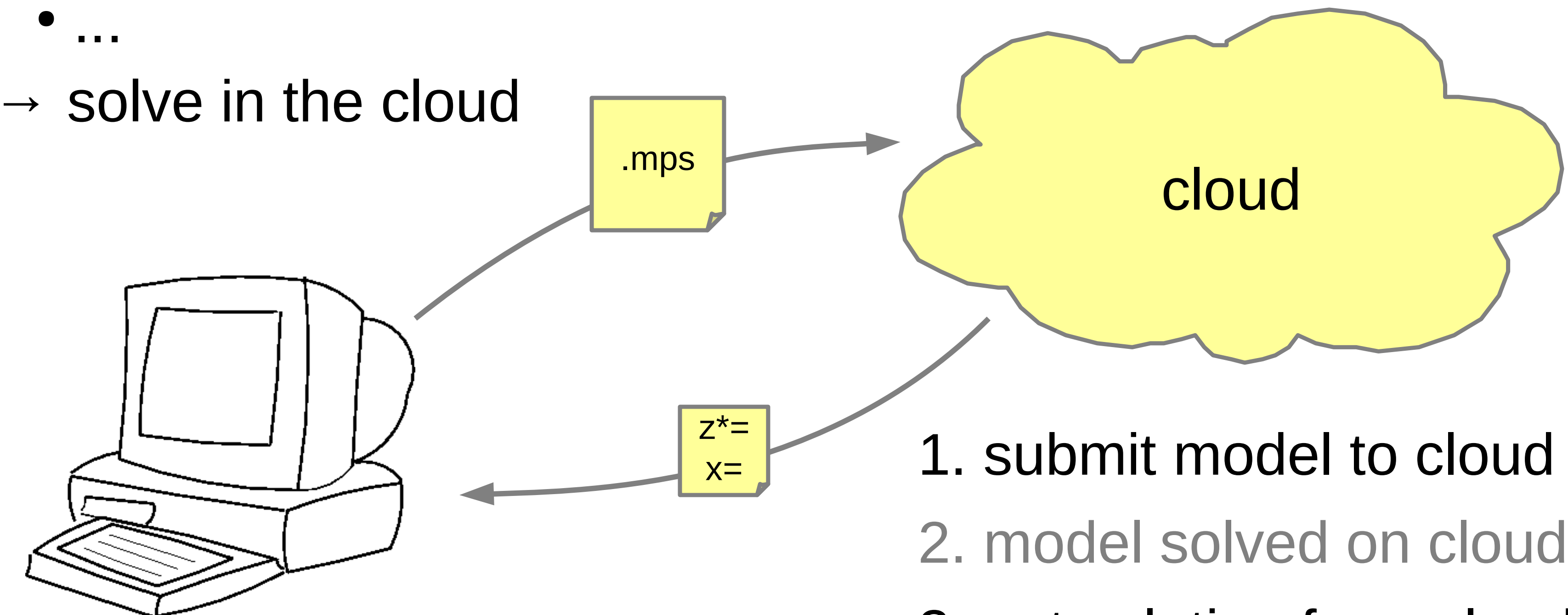

1. submit model to cloud
2. model solved on cloud

# CPLEX in the cloud

What if no local resources to solve model?

- model too hard
- only need to solve once in a while
- ...

→ solve in the cloud




1. submit model to cloud
2. model solved on cloud
3. get solution from cloud

# CPLEX in the cloud

Two ways to access CPLEX in the cloud

1. Dropsolve

`www.ibm.com/software/analytics/docloud`
`dropsolve-oaas.docloud.ibmcloud.com/software/analytics/docloud`

Drag and drop your model file from disk to web browser

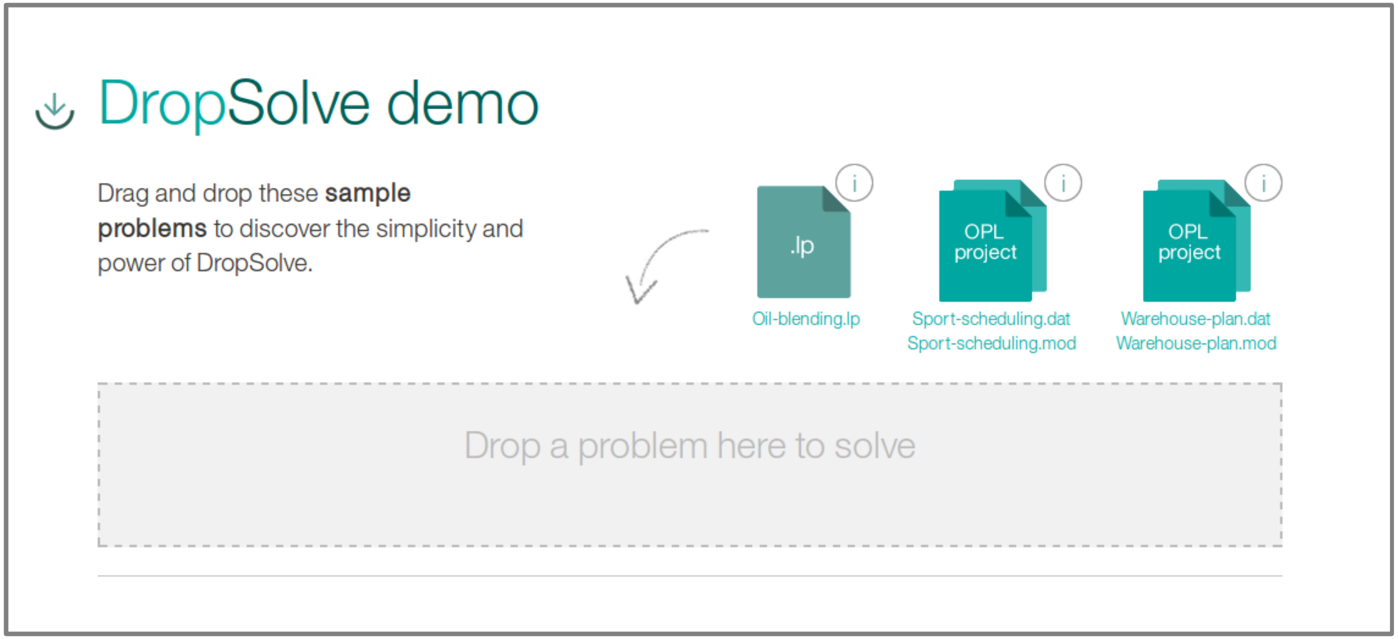
# CPLEX in the cloud

Two ways to access CPLEX in the cloud

1. Dropsolve

www.ibm.com/software/analytics/docloud
dropsolve-oaas.docloud.ibmcloud.com/software/analytics/docloud

Drag and drop your model file from disk to web browser



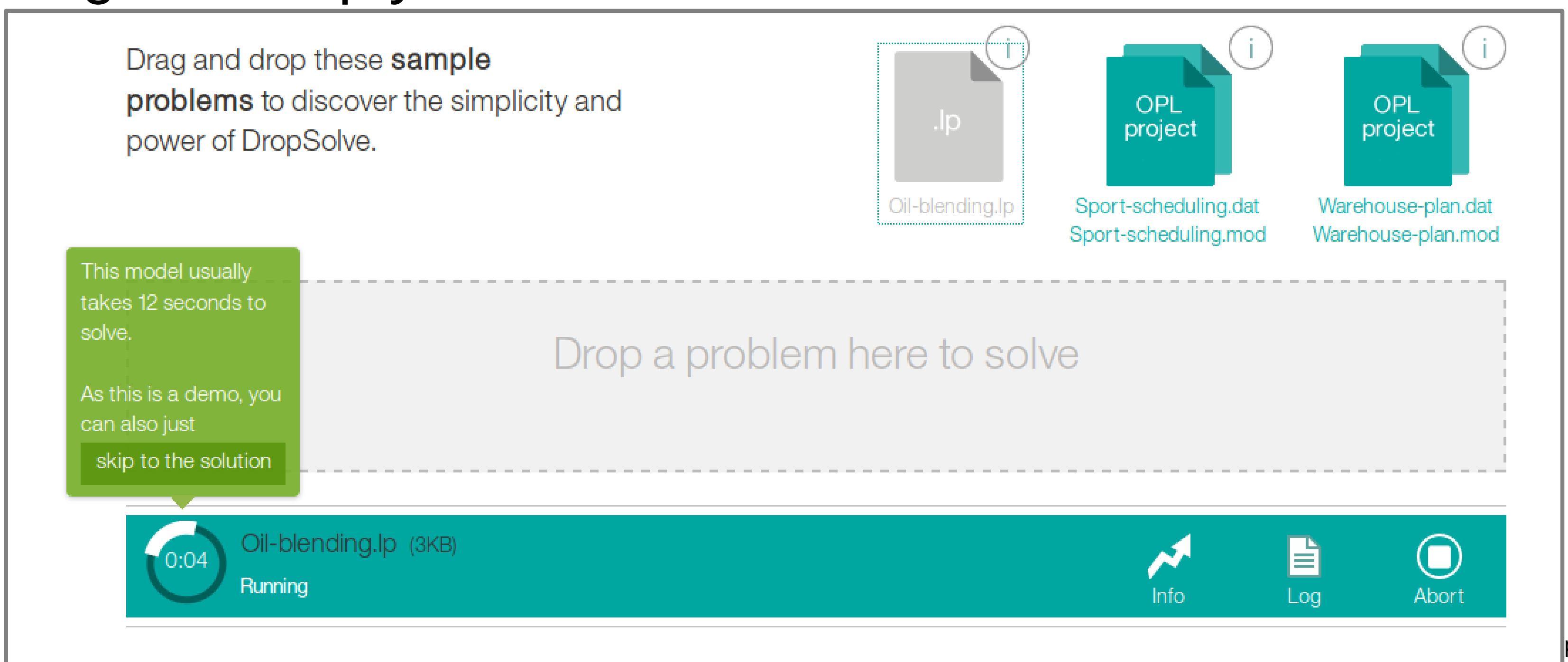
# CPLEX in the cloud

## Two ways to access CPLEX in the cloud

## 1. Dropsolve

www.ibm.com/software/analytics/docloud
dropsolve-oaas.docloud.ibmcloud.com/software/analytics/docloud

## Drag and drop your model file from disk to web browser



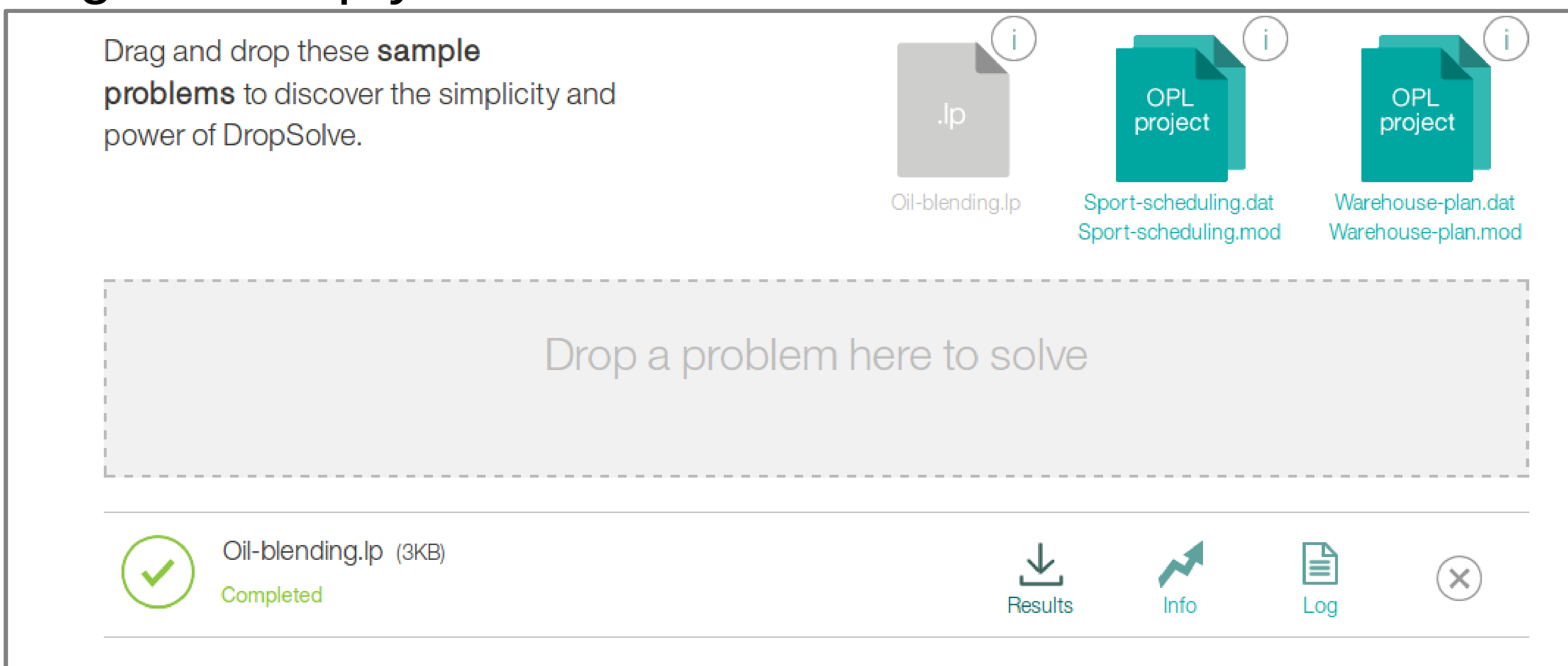
# CPLEX in the cloud

Two ways to access CPLEX in the cloud

1. Dropsolve

www.ibm.com/software/analytics/docloud
dropsolve-oaas.docloud.ibmcloud.com/software/analytics/docloud

Drag and drop your model file from disk to web browser



# CPLEX in the cloud

Two ways to access CPLEX in the cloud

2. REST API

https://developer.ibm.com/docloud/
https://developer.ibm.com/docloud/docs/welcome/

Access the solve service via its REST API

# CPLEX in the cloud

## Two ways to access CPLEX in the cloud

## 2. REST API

https://developer.ibm.com/docloud/
https://developer.ibm.com/docloud/docs/welcome/

Access the solve service via its REST API

- ready-to-use clients provided for Java™ and Python, e.g.

```
JobExecutor executor = JobExecutorFactory.createDefault();
JobClient jobclient = JobClientFactory.createDefault(BASE_URL,
                                                     APIKEY_CLIENTID);
jobclient.newRequest().input(new File("model.mps"))
                      .output(new File("x.sol"))
                      .execute(executor).get();
```

- With any HTTP client (cURL, ...)

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

# CPLEX in the cloud

Sign up for a free trial

www.ibm.com/software/analytics/docloud

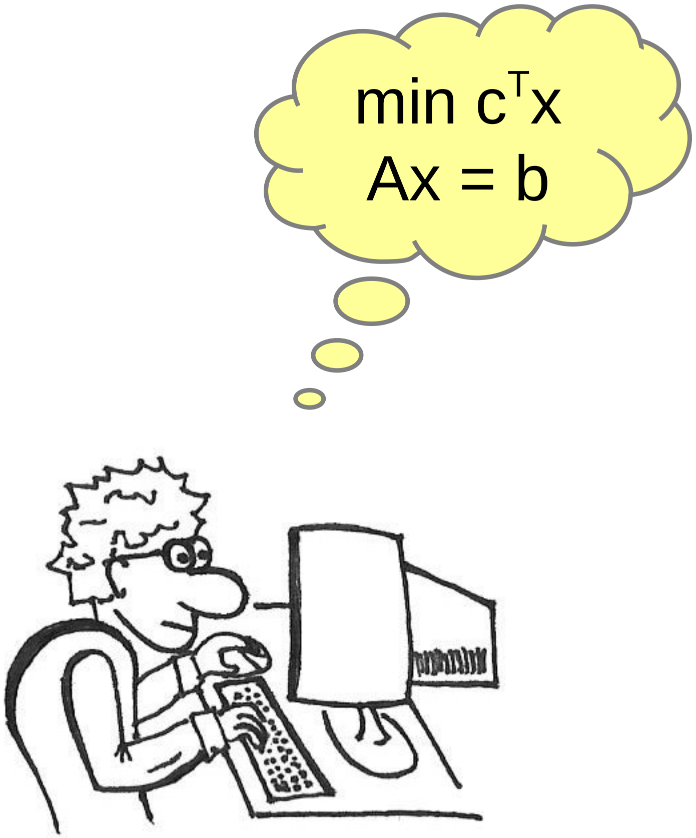
https://developer.ibm.com/docloud/

free CPLEX community edition

www.cplex.com

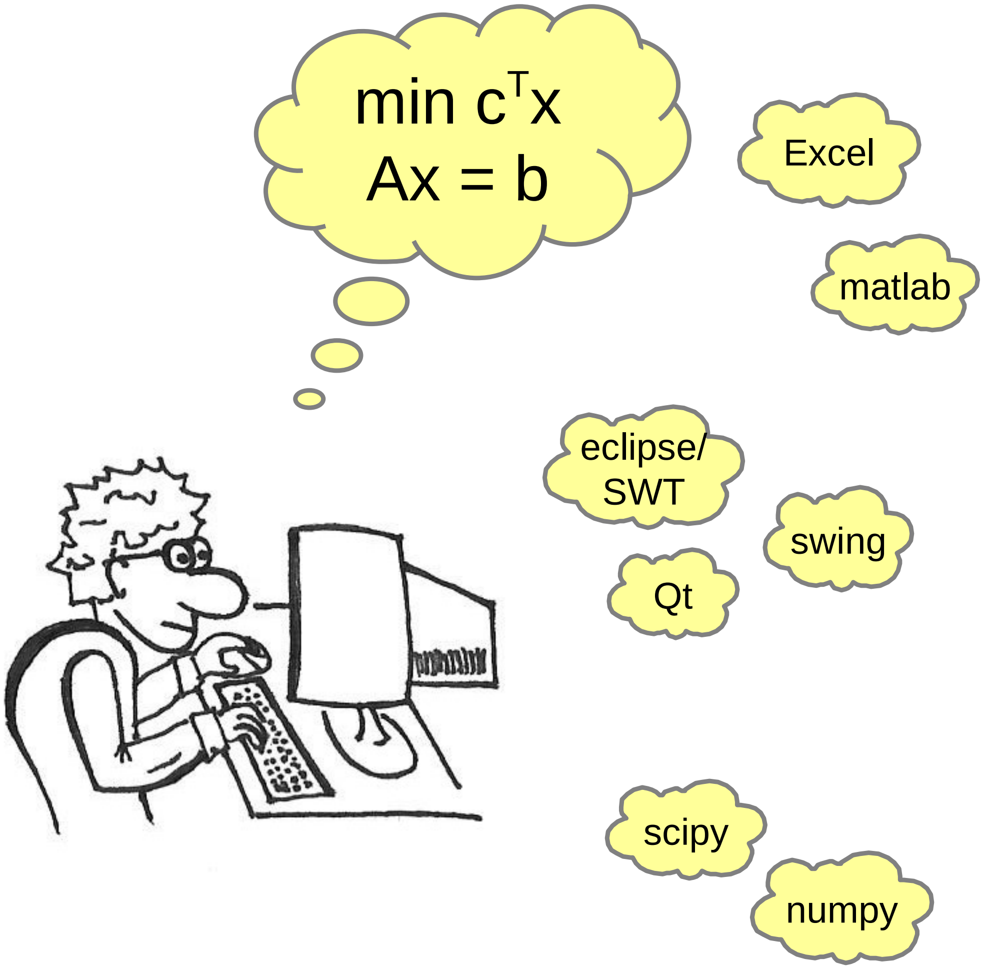
# Exploiting existing software resources

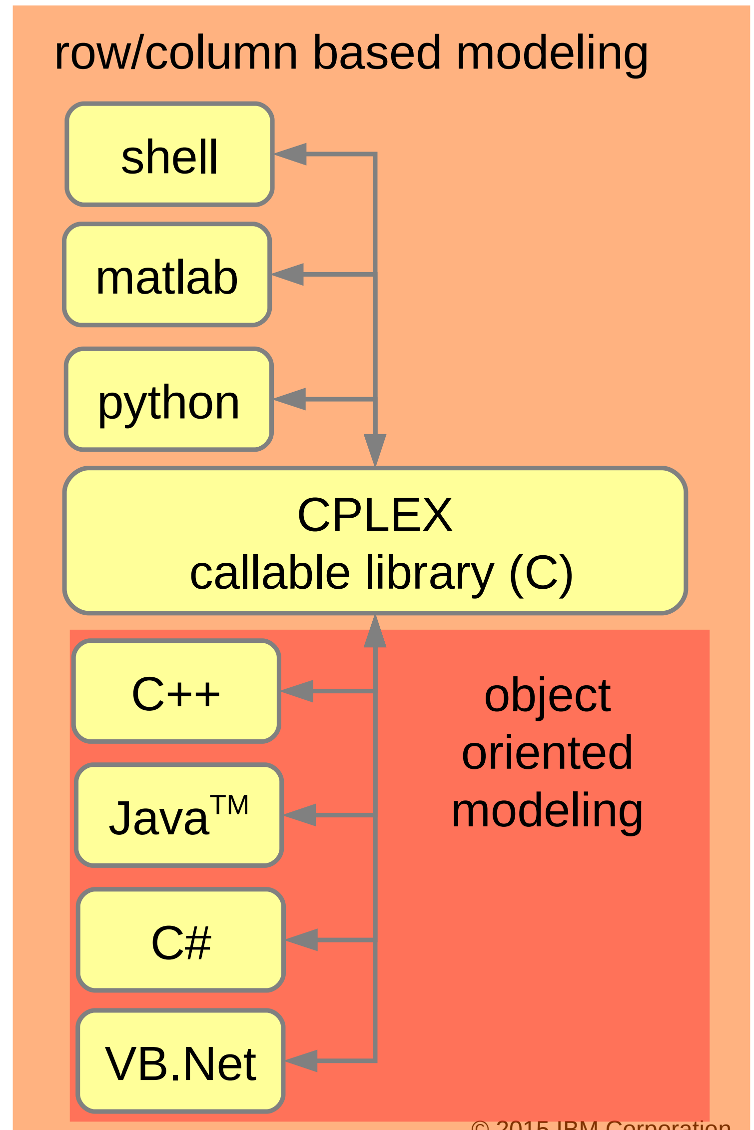
# Interfacing with CPLEX



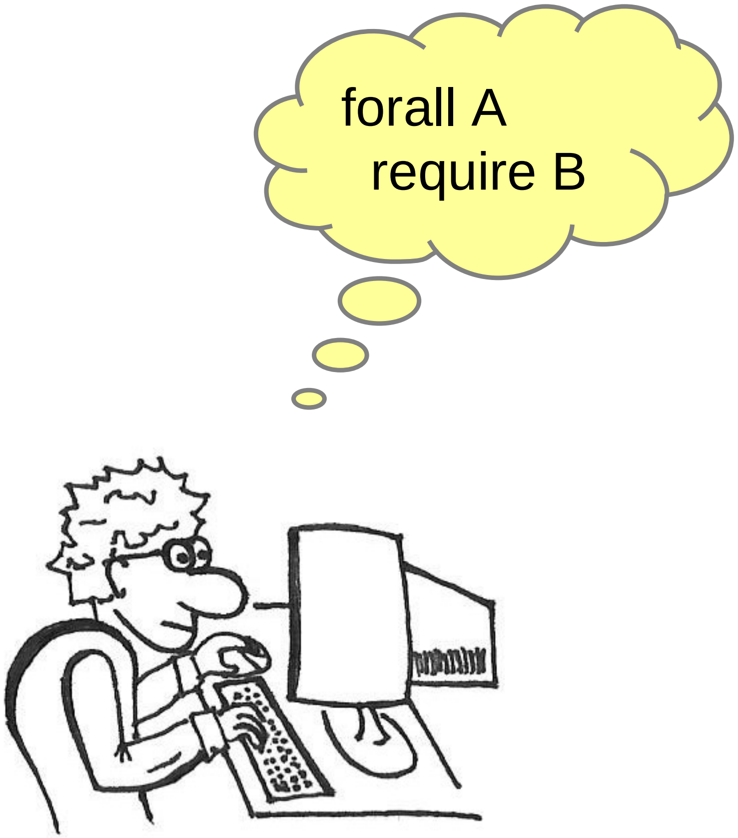

CPLEX
callable library (C)

# Interfacing with CPLEX



# Interfacing with CPLEX




## **O**ptimization **P**rogramming **L**anguage (OPL)

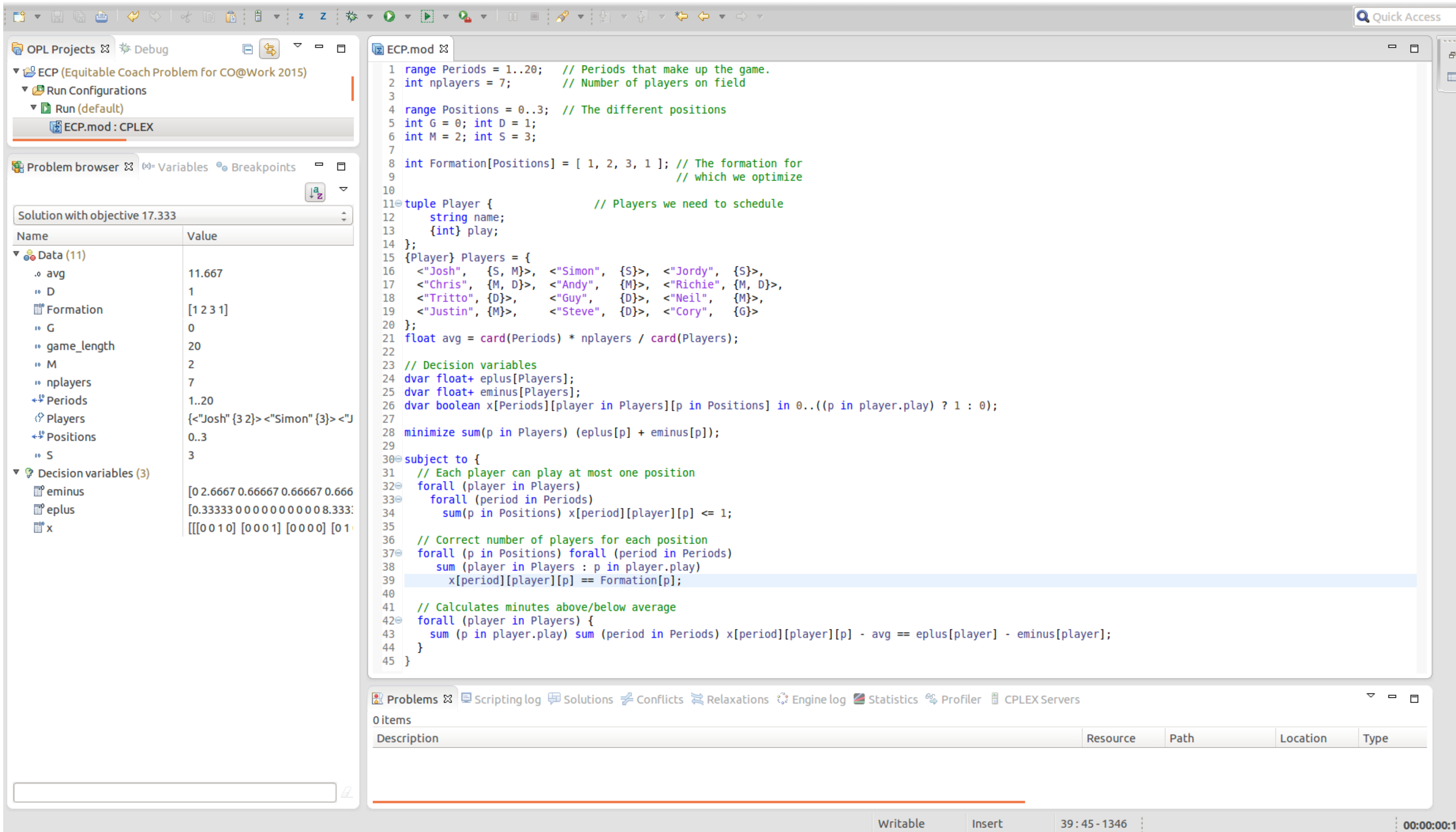
- write models in a more descriptive form
- write models in a more compact form
- faster prototyping, easier maintenance
- easier access to data (Excel, database, ...)

$\sum\sum x_{ijk} \leq 1$ for i in I

```
forall (i in I)
  sum (j in J)
    sum (k in K)
      x[i][j][k] <= 1;
```

- scriptable
- model editor
- IDE support (eclipse based)

# Interfacing with CPLEX



# Interfacing with CPLEX

ECP.mod

```
range Periods = 1..20;   // Periods that make up the game.
int nplayers = 7;        // Number of players on field

range Positions = 0..3;  // The different positions
int G = 0; int D = 1;
int M = 2; int S = 3;

int Formation[Positions] = [ 1, 2, 3, 1 ]; // The formation for
                                           // which we optimize

tuple Player {                // Players we need to schedule
    string name;
    {int} play;
};
{Player} Players = {
  <"Josh",   {S, M}>,  <"Simon",  {S}>,  <"Jordy",  {S}>,
  <"Chris",  {M, D}>,  <"Andy",   {M}>,  <"Richie", {M, D}>,
  <"Tritto", {D}>,     <"Guy",    {D}>,  <"Neil",   {M}>,
  <"Justin", {M}>,     <"Steve",  {D}>,  <"Cory",   {G}>
};
float avg = card(Periods) * nplayers / card(Players);

// Decision variables
dvar float+ eplus[Players];
dvar float+ eminus[Players];
dvar boolean x[Periods][player in Players][p in Positions] in 0..((p in player.play) ? 1 : 0);

minimize sum(p in Players) (eplus[p] + eminus[p]);

subject to {
  // Each player can play at most one position
  forall (player in Players)
    forall (period in Periods)
      sum(p in Positions) x[period][player][p] <= 1;

  // Correct number of players for each position
  forall (p in Positions) forall (period in Periods)
     sum (player in Players : p in player.play)
       x[period][player][p] == Formation[p];

  // Calculates minutes above/below average
  forall (player in Players) {
    sum (p in player.play) sum (period in Periods) x[period][player][p] - avg == eplus[player] - eminus[player];
  }
}
```

# `docplex`

## Most recent addition

- https://pypi.python.org/pypi/docplex
- pure Python modeling API (no native code)
- open source (`pypi`, `github`)
- prepared to connect to local or cloud CPLEX

→ write your model in Python
→ hook up with the whole Python software ecosystem

# docplex

## Most recent addition

- https://pypi.python.org/pypi/docplex
- pure Python modeling API (no native code)
- open source (`pypi`, `github`)
- prepared to connect to local or cloud CPLEX

→ write your model in Python
→ hook up with the whole Python software ecosystem

## For example

- Equitable Coach Problem
- list of players from the internet (web service)
- graphical display of solution

→ use `iPython/Jupyter` notebook and Python libraries

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.
- If the text contains performance statistics or references to benchmarks, insert the following language; otherwise delete:
  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.
- If the text includes any customer examples, please confirm we have prior written approval from such customer and insert the following language; otherwise delete:
  All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved.  Actual environmental costs and performance characteristics may vary by customer.
- Please review text for proper trademark attribution of IBM products.  At first use, each product name must be the full name and include appropriate trademark symbols (e.g., IBM Lotus® Sametime® Unyte™).  Subsequent references can drop "IBM" but should include the proper branding (e.g., Lotus Sametime Gateway, or WebSphere Application Server). Please refer to http://www.ibm.com/legal/copytrade.shtml for guidance on which trademarks require the ® or ™ symbol.  Do not use abbreviations for IBM product names in your presentation. All product names must be used as adjectives rather than nouns.  Please list all of the trademarks that you use in your presentation as follows; delete any not included in your presentation. IBM, the IBM logo, Lotus, Lotus Notes, Notes, Domino, Quickr, Sametime, WebSphere, UC2,  PartnerWorld and Lotusphere are trademarks of International Business Machines Corporation in the United States, other countries, or both.   Unyte is a trademark of WebDialogs, Inc., in the United States, other countries, or both.
- If you reference Adobe® in the text, please mark the first use and include the following; otherwise delete:
  Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- If you reference Java™ in the text, please mark the first use and include the following; otherwise delete:
  Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- If you reference Microsoft® and/or Windows® in the text, please mark the first use and include the following, as applicable; otherwise delete:
  Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- If you reference Intel® and/or any of the following Intel products in the text, please mark the first use and include those that you use as follows; otherwise delete:
  Intel, Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- If you reference UNIX® in the text, please mark the first use and include the following; otherwise delete:
  UNIX is a registered trademark of The Open Group in the United States and other countries.
- If you reference Linux® in your presentation, please mark the first use and include the following; otherwise delete:
  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.
- If the text/graphics include screenshots, no actual IBM employee names may be used (even your own), if your screenshots include fictitious company names (e.g., Renovations, Zeta Bank, Acme) please update and insert the following; otherwise delete: All references to [insert fictitious company name] refer to a fictitious company and are used for illustration purposes only.