International Ruhr Energy Conference (INREC)
Essen, Germany

GAMS

Solving Large-Scale Energy System Models

Frederik Fiand
Operations Research Analyst
GAMS Software GmbH

Agenda

1. GAMS – System Overview
2. BEAM-ME – Background
3. BEAM-ME – Lessons Learned
4. BEAM-ME – High-Performance-Computing
5. Summary/Outlook

GAMS

System Overview

Algebraic Modeling Language

Facilitates to formulate mathematical optimization problems similar to algebraic notation

→Simplified model building:

Model is executable algebraic description of optimization problem.

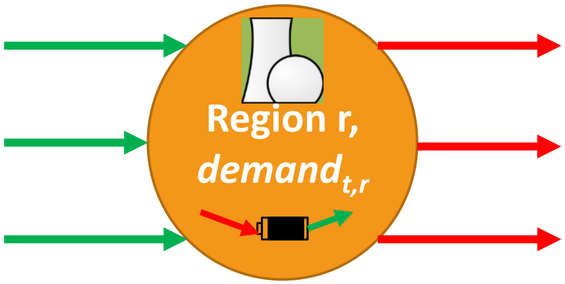
Algebraic Modeling Language

Facilitates to formulate mathematical optimization problems similar to algebraic notation

→Simplified model building:

Model is executable algebraic description of optimization problem.

$$
\begin{aligned}
& \sum_{p \in P: rp_{r,p}} \mathrm{POWER}_{t,r,p} \\
+ & \sum_{r2 \in R: net_{r2,r}} (\mathrm{FLOW}_{t,r2,r}) - \sum_{r2: net_{r,r2}} \mathrm{FLOW}_{t,r,r2} \\
+ & \sum_{s \in S: rs_{r,s}} (\mathrm{STORAGE\_OUTFLOW}_{t,r,s} - \mathrm{STORAGE\_INFLOW}_{t,r,s}) \quad \geq \mathrm{demand}_{t,r} \quad \forall t \in T, r \in R
\end{aligned}
$$



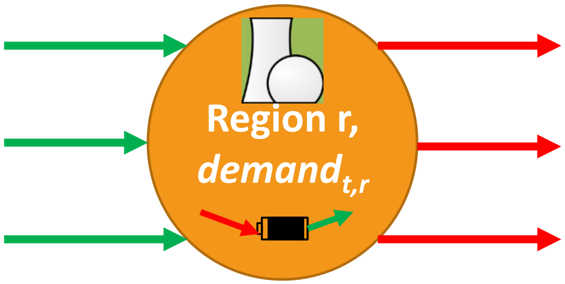
Algebraic Modeling Language

Facilitates to formulate mathematical optimization problems similar to algebraic notation

→Simplified model building:

Model is executable algebraic description of optimization problem.

$$\sum_{p \in P: rp_{r,p}} \mathrm{POWER}_{t,r,p} + \sum_{r2 \in R: net_{r2,r}} (\mathrm{FLOW}_{t,r2,r}) - \sum_{r2: net_{r,r2}} \mathrm{FLOW}_{t,r,r2} + \sum_{s \in S: rs_{r,s}} (\mathrm{STORAGE\_OUTFLOW}_{t,r,s} - \mathrm{STORAGE\_INFLOW}_{t,r,s}) \geq \mathrm{demand}_{t,r} \quad \forall t \in T, r \in R$$




```
eq_power_balance(t,r)..
  sum(rp(r,p),    POWER(t,r,p))
+ sum(net(r2,r), FLOW(t,net)) - sum(net(r,r2), FLOW(t,net))
+ sum(rs(r,s),    STORAGE_OUTFLOW(t,r,s) - STORAGE_INFLOW(t,r,s))
=g= demand(t,r);
```

Algebraic Modeling Language

Facilitates to formulate mathematical optimization problems similar to algebraic notation

→Simplified model building:

→Switching solvers with one line of code!

All major commercial LP/MIP solver

Open Source Solver (COIN)

Also solver for NLP, MINLP, global, and stochastic optimization

…

Algebraic Modeling Language

Facilitates to formulate mathematical optimization problems similar to algebraic notation

→Simplified model building:

Declarative elements

- Similar to mathematical notation
- Easy to learn - few basic language elements: sets, parameters, variables, equations, models
- Model is executable (algebraic) description of the problem

Procedural elements

- Control Flow Statements (e.g. loops, for, if,...),
- Build complex problem algorithms within GAMS
- Simplified interaction with other systems
  - Data exchange
  - GAMS process control

Fields of Application

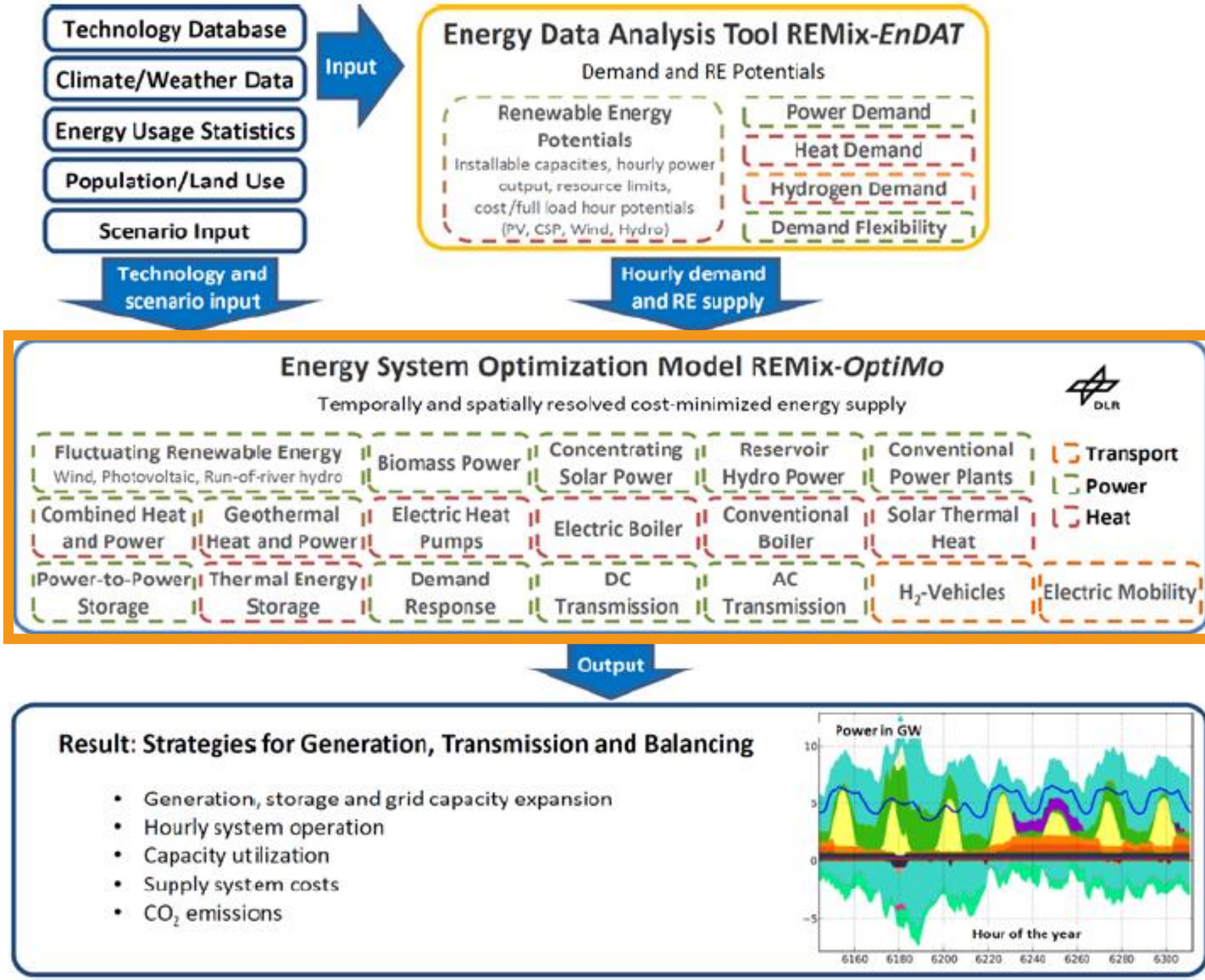
Agricultural Economics	Applied General Equilibrium
Chemical Engineering	Economic Development
Econometrics	**Energy**
Environmental Economics	Engineering
Finance	Forestry
International Trade	Logistics
Macro Economics	Military
Management Science/OR	Mathematics
Micro Economics	Physics

GAMS is widespread in the Energy community:
http://www.energyplan.eu/othertools/

BEAM-ME

Project Background

Energy System Models



Optimization Problem

Focus on **large-scale Linear Programs** (LPs)

http://www.dlr.de/Portaldata/41/Resources/dokumente/institut/system/Modellbeschreibungen/DLR_Energy_System_Model_REMix_short_description_2016.pdf

The BEAM-ME Project

Supported by:

Federal Ministry for Economic Affairs and Energy

on the basis of a decision by the German Bundestag

What exactly is BEAM-ME about?

Implementation of acceleration strategies from mathematics and computational sciences for optimizing energy system models

An Interdisciplinary Approach:

Energy System Modeling → Modeling Language → Solver Development → High Performance Computing

Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

GAMS

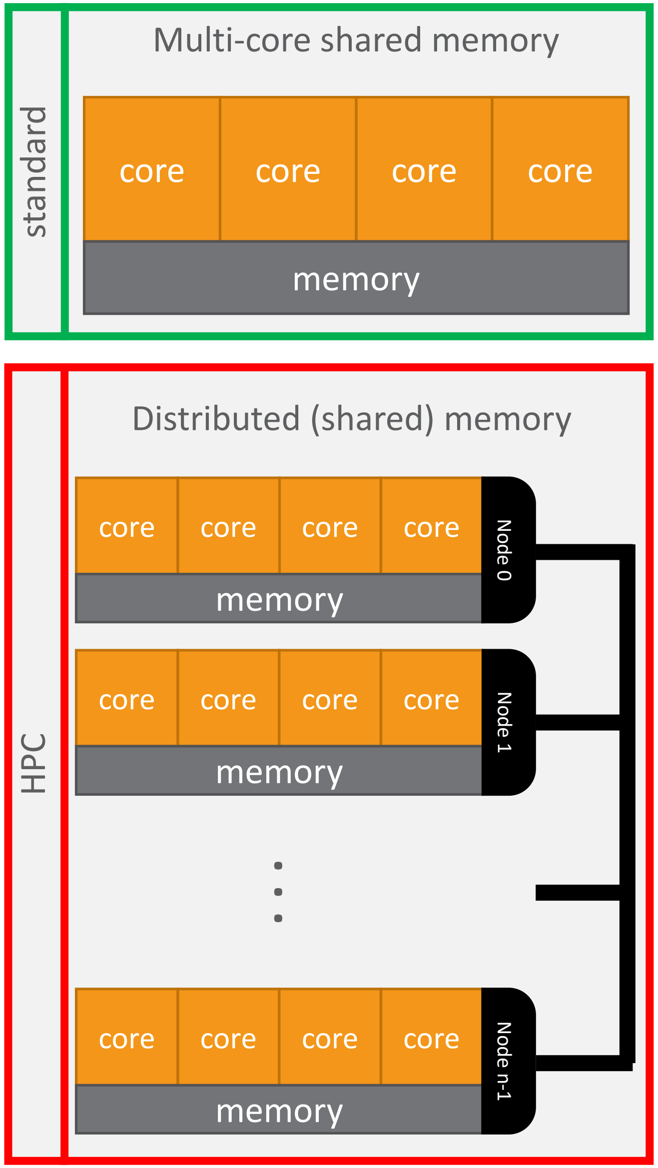
ZIB

TU berlin

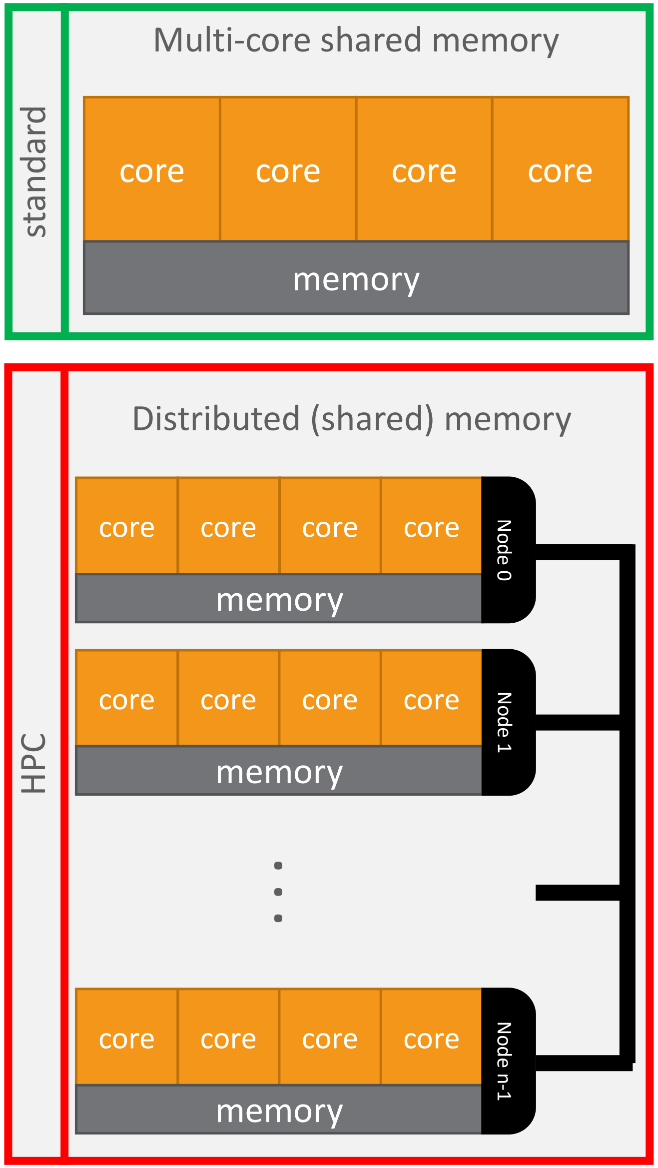
HLRS

JÜLICH
FORSCHUNGSZENTRUM

Available Computing Resources



Available Computing Resources

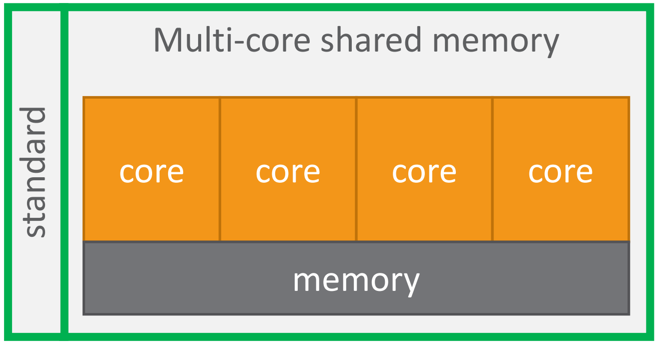


Convenient to use.

Model should be brought "in shape" and capabilities of standard hardware should be exploited first.

Section 3

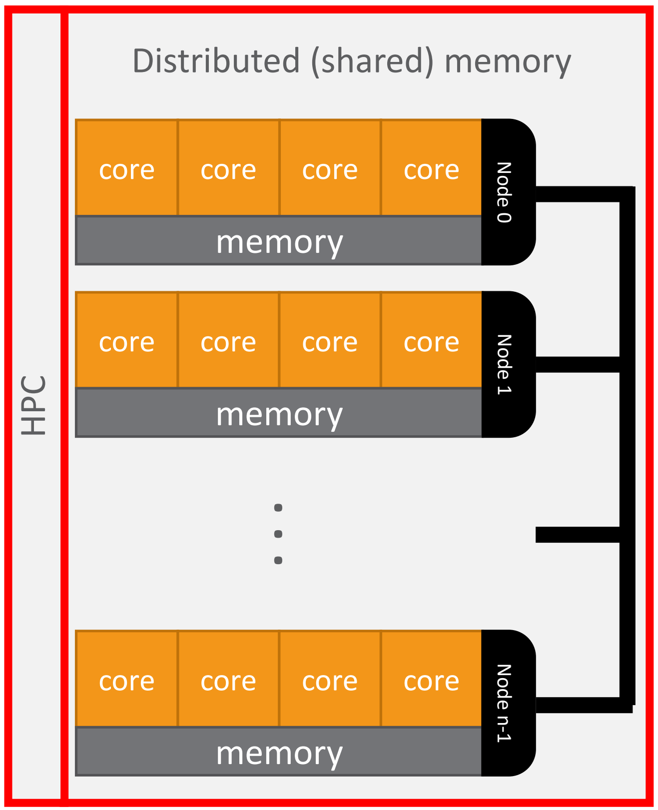
Available Computing Resources



Convenient to use.

Model should be brought “in shape” and capabilities of standard hardware should be exploited first.

Section 3



Complex to use.

But huge speedup potential for *certain* models/methods.

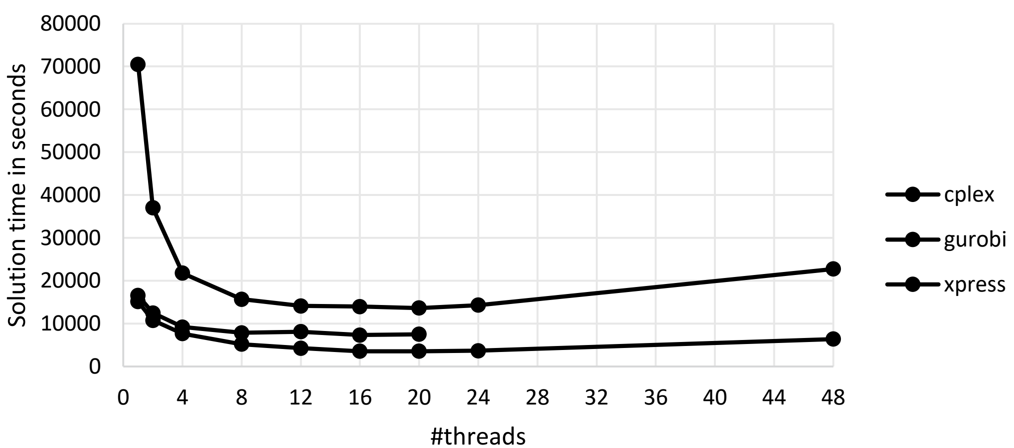
Section 4

BEAM-ME

Lessons Learned from Solving Large-Scale ***Linear Programs***

1. Choice of Solver/Algorithm

- Try different solvers
- Try different algorithms and choose the superior explicitly
  - Simplex
  - Barrier (**often much faster on large-scale LPs**)
- What type of solution is needed?
  - Basic solution (Simplex or Barrier+Crossover)
  - Interior point (**Crossover time often dominates barrier time**)
- Barrier Algorithm benefits from multiple threads (on shared memory machines)



Solution time comparison for a large-scale LP

- 27,212,755 rows
- 29,701,364 columns
- 93,938,356 non-zeroes

solved on a 24 core shared memory machine.

2. Check Where Time is Consumed

- Usually `solver time >> GAMS time`

- If GAMS execution is suspiciously slow, vast amount of time is often consumed in few lines

- Profiling gives detailed feedback on time consumption
  (https://www.gams.com/latest/docs/UG_ExecErrPerformance.html#UG_ExecErrPerformance_ExecutionProfile)

```
...
x(a,b,c) = sum(d, p(a,b,c,d));
y(c,b,a) = sum(d, p(a,b,c,d));
...
```

Profiling Output:

```
----     13 Assignment x        3.875    11.922 SECS   3,260 MB   1000000
----     14 Assignment y        9.500    21.422 SECS   3,292 MB   1000000
```

- Ordering indices consistently can make a huge difference
  (https://www.gams.com/latest/docs/UG_ExecErrPerformance.html#UG_ExecErrPerformance_ExecutionProfile)

3. Scaling is Important

- Rules for good scaling are exclusively based on algorithmic needs.
- Rules of thumb:
  - Ideally, constants should have values "around 1", e.g. `0.1 < |A(i,j)| < 10`
  - Ratio of max/min non-zero coefficient in row/column should be < 1e6,
  - https://www.gams.com/latest/docs/UG_NLP_GoodFormulations.html#UG_NLP_GoodFormulations_Scaling
  - https://www.gams.com/latest/docs/UG_LanguageFeatures.html#UG_LanguageFeatures_ModelScaling-TheScaleOption
  - https://www.gams.com/fileadmin/community/mccarlarchive/news41.pdf
- Solvers give warnings if numerical difficulties occur
  - `… Solution available but not proven optimal due to numerical difficulties. …`
  - `… Warning: Model contains large rhs …`
- GAMS ships a tool GAMSCHK to examine a problem's structure (https://www.gams.com/latest/docs/S_GAMSCHK.html)

4. Look at Solver Output

- Solvers provide useful tools and give helpful messages, e.g.
  - CPLEX option datacheck (https://www.gams.com/latest/docs/S_CPLEX.html#CPLEXdatacheck)

```
… Detected nonzero <= the maximum value of either CPX_PARAM_EPRHS or
CPX_PARAM_EPOPT at constraint 188802, variable 7741216 …
… Detected constraint with wide range of coefficients. In constraint 'e3'
the ratio of largest and smallest (in absolute value) coefficients is
1.63452e+11. …
```

  - CPLEX option quality (https://www.gams.com/latest/docs/S_CPLEX.html#CPLEXquality)

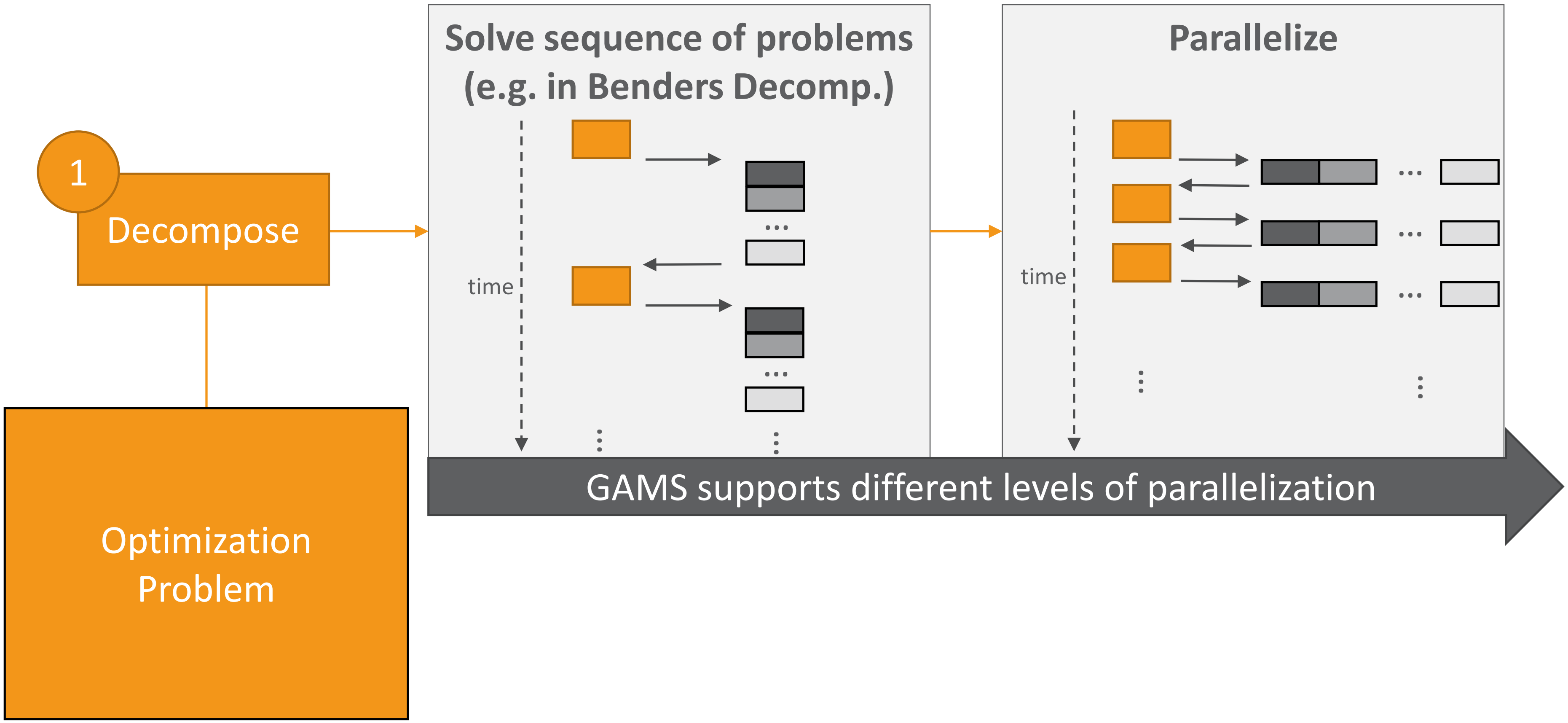
```
… Detected 100.00% (1) unstable condition number(s) >= 1e+10.
…
Solution Quality Statistics:
                                       unscaled                 scaled
                                   max         sum         max         sum
primal infeasibility           2.274e-13   6.579e-13   4.441e-16   7.804e-16
dual infeasibility             8.959e-10   7.484e-08   9.999e-10   1.810e-07
…
Condition number of the scaled basis matrix =    8.994e+11
```

  - GUROBI coefficient statistics

```
Coefficient statistics:
  Matrix range     [1e-03, 1e+05]
  Objective range  [6e-03, 1e-02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [9e-01, 2e+09]
Warning: Model contains large rhs
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
```

BEAM-ME

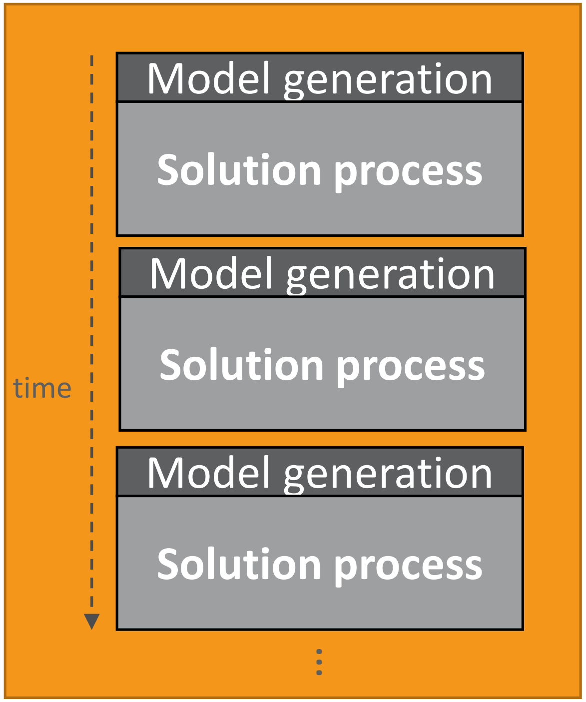
High-Performance-Computing: Two Examples


Solve sequence of problems
(e.g. in Benders Decomp.)
Parallelize
1
Decompose
time
time
Optimization Problem
GAMS supports different levels of parallelization

Parallelization with GAMS

From Sequential to Parallel Solve Statements

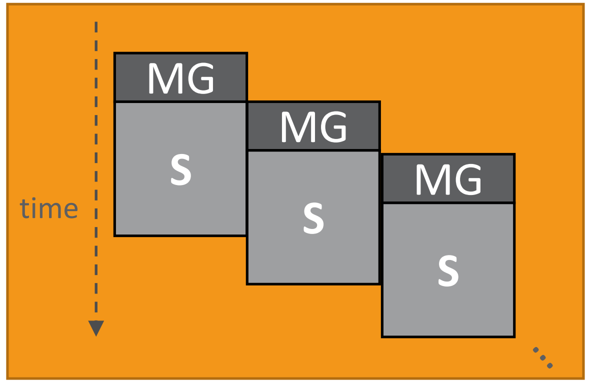
Sequential Solve Statements



- Simple sequential loop body
- Limited to **shared memory**

```
... //preparatory work
loop(scen,
  ... //model setup
  solve mymodel min obj use lp;
  ... //process results
);
... //reporting
```

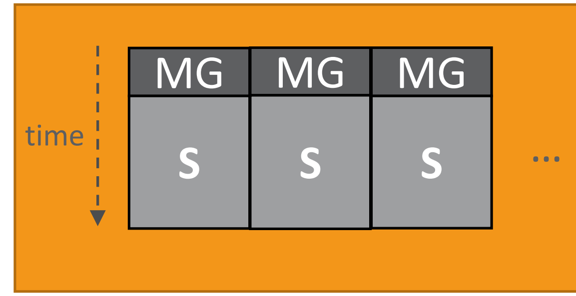
Asynchronous Solve Statements



- GAMS Grid Facility
- SolveLink option specifies the solver linking conventions
- Split loop in submission & collection loop
- Limited to **shared memory** or file based I/O

https://www.gams.com/latest/docs/UG_GridComputing.html

Parallel Solve Statements



- Run GAMS as an MPI Program on **distributed memory**
- Efficient network based inter-process communication via embedded Python code and mpi4py
- Requires reorganization of the code

https://www.gams.com/latest/docs/UG_EmbeddedCode.html

Example: Sequential Benders Decomposition

```
set k       'benders iterations' / k1*k1000 /
    scen    'scenario set' / scen1*scen100 /
singleton set s(scen) 'active scenario';

... // preparatory work
loop(k$( NOT done ),
  ... // setup model for master-problem
  solve master min obj_master use lp;
  ... // fix first stage variables
  loop(scen,
    ... // setup model for sub-problem
    s(scen) = yes;
    solve sub min obj_sub use lp;
    ... // process results
  );
  ... // compute cuts for next master
  ... // free fixed first stage variables
  ... // set done=1 if convergence criterion is met
);
... // reporting
```

Example: Parallel Benders with mpi4py

PMI_RANK=0

```
set k        'benders iterations' / k1*k1000 /
    scen     'scenario set' / scen1*scen100 /
singleton set s(scen) 'active scenario';

...
embeddedCode Python:
  from mpi4py import *
  comm = MPI.COMM_WORLD
  ...
pauseEmbeddedCode
... // preparatory work
$ifthen.MPI 0==%sysenv.PMI_RANK%
loop(k$( NOT done ),
  ... // setup model for master-problem
  solve master min obj_master use lp;
  ... // fix first stage variables
  continueEmbeddedCode:
    comm.bcast([[done]] + <data for sub>, root=0)
    cut = comm.gather(None, root=0)[1:]
    ... // gathered data → GAMS data struct.
  pauseEmbeddedCode <load GAMS data struct.>
  ... // compute cuts
  ... // free fixed first stage variables
  ... // set done=1 if convergence criterion is met
);
continueEmbeddedCode:
  comm.bcast([[done],<empty>], root=0)
endEmbeddedCode
... // reporting
$else.MPI
```

PMI_RANK>=1

```
set k        'benders iterations' / k1*k1000 /
    scen     'scenario set' / scen1*scen100 /
singleton set s(scen) 'active scenario';

...
embeddedCode Python:
  from mpi4py import *
  comm = MPI.COMM_WORLD
  ...
pauseEmbeddedCode
... // preparatory work
$else.MPI
  s(scen) = ord(scen)=%sysenv.PMI_RANK%;
  while(1,
    continueEmbeddedCode:
      primal_solution = comm.bcast(None, root=0)
      // broadcasted data → GAMS data struct.
    pauseEmbeddedCode <GAMS data struct.>
    abort.noerror$done 'terminating subprocess';
    solve sub min obj_sub use lp;
    ... // process results
    continueEmbeddedCode:
      comm.gather(<subproblem results>), root=0 )
    pauseEmbeddedCode
  );
$endif.MPI
```

Example: Parallel Benders with mpi4py

PMI_RANK=0

```
set k       'benders iterations' / k1*k1000 /
    scen    'scenario set' / scen1*scen100 /
singleton set s(scen) 'active scenario';

...
embeddedCode Python:
  from mpi4py import *
  comm = MPI.COMM_WORLD
  ...
pauseEmbeddedCode
... // preparatory work
$ifthen.MPI 0==%sysenv.PMI_RANK%
loop(k$( NOT done ),
  ... // setup model for master-problem
  solve master min obj_master use lp;
  ... // fix first stage variables
  continueEmbeddedCode:
    comm.bcast([[done]] + <data for sub>, root=0)
    cut = comm.gather(None, root=0)[1:]
    ... // gathered data → GAMS data struct.
  pauseEmbeddedCode <load GAMS data struct.>
  ... // compute cuts
  ... // free fixed first stage variables
  ... // set done=1 if convergence criterion is met
);
continueEmbeddedCode:
  comm.bcast([[done],<empty>], root=0)
endEmbeddedCode
... // reporting
$else.MPI
```

PMI_RANK>=1

```
set k       'benders iterations' / k1*k1000 /
    scen    'scenario set' / scen1*scen100 /
singleton set s(scen) 'active scenario';

...
embeddedCode Python:
  from mpi4py import *
  comm = MPI.COMM_WORLD
  ...
pauseEmbeddedCode
... // preparatory work
$else.MPI
  s(scen) = ord(scen)=%sysenv.PMI_RANK%;
  while(1,
    continueEmbeddedCode:
      primal_solution = comm.bcast(None, root=0)
      // broadcasted data → GAMS data struct.
    pauseEmbeddedCode <GAMS data struct.>
    abort.noerror$done 'terminating subprocess';
    solve sub min obj_sub use lp;
    ... // process results
    continueEmbeddedCode:
      comm.gather(<subproblem results>), root=0 )
    pauseEmbeddedCode
  );
$endif.MPI
```

- **Code refactorization**
- **19 lines of embedded Python**

Computational Result(s)

- Two-stage stochastic problem emerged from energy system model
- 100 scenarios
- Deterministic Equivalent:
  21,029,101 rows, 23,217,077 columns, 85,721,477 non-zeroes
- Benders:
  - Master: up to 553 rows, 177 columns, 24,911 non-zeroes
  - Sub: 210,282 rows 232,161 columns 696,461 non-zeroes
  - 19 lines of Python Code + some refactorization of GAMS code for MPI version

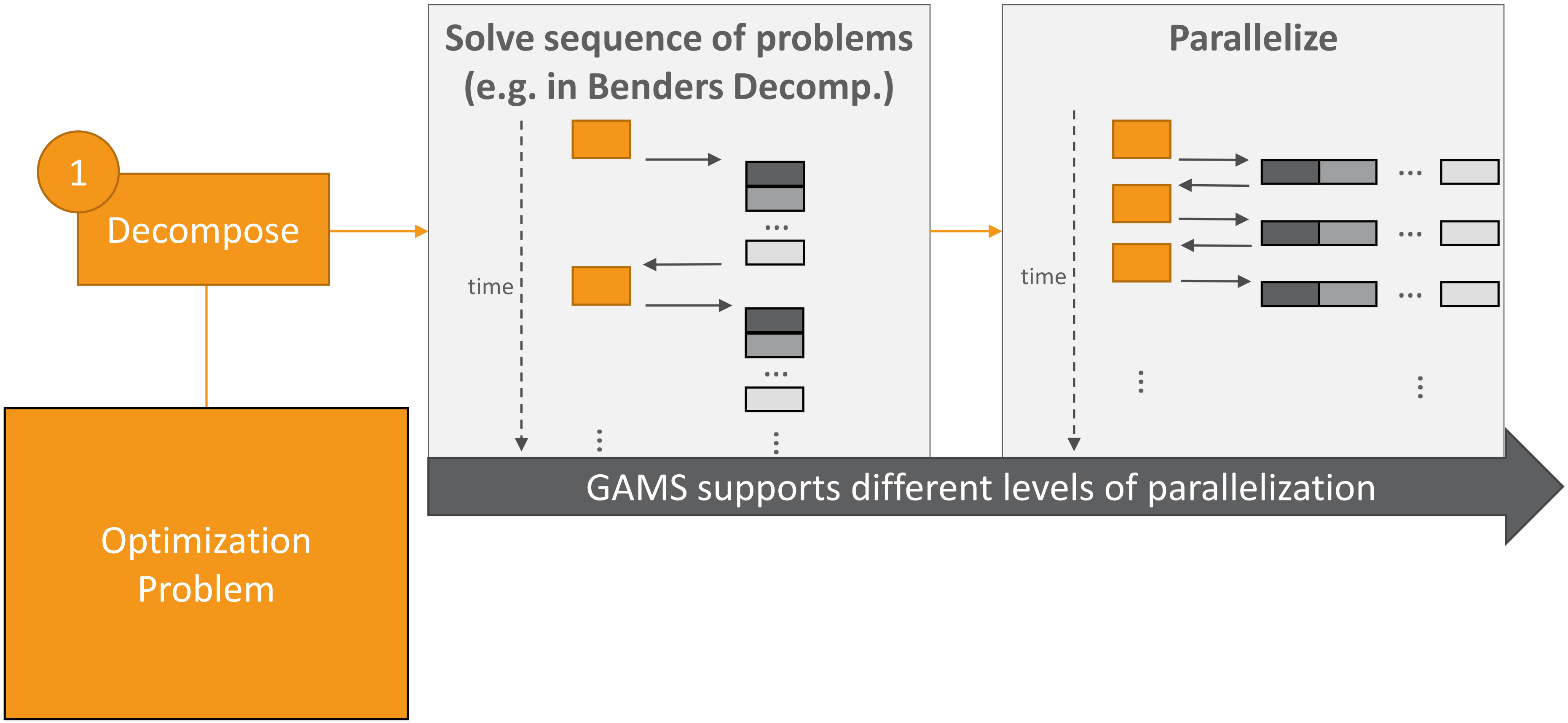
	TIME [sec]		
Method	**sub-problems**	**master-problem**	**total**
Deterministic Equivalent[1]			4059.00
Seq. Benders[2]	2394.92	0.18	2395.10
MPI Benders[3]	28.35	0.16	28.51

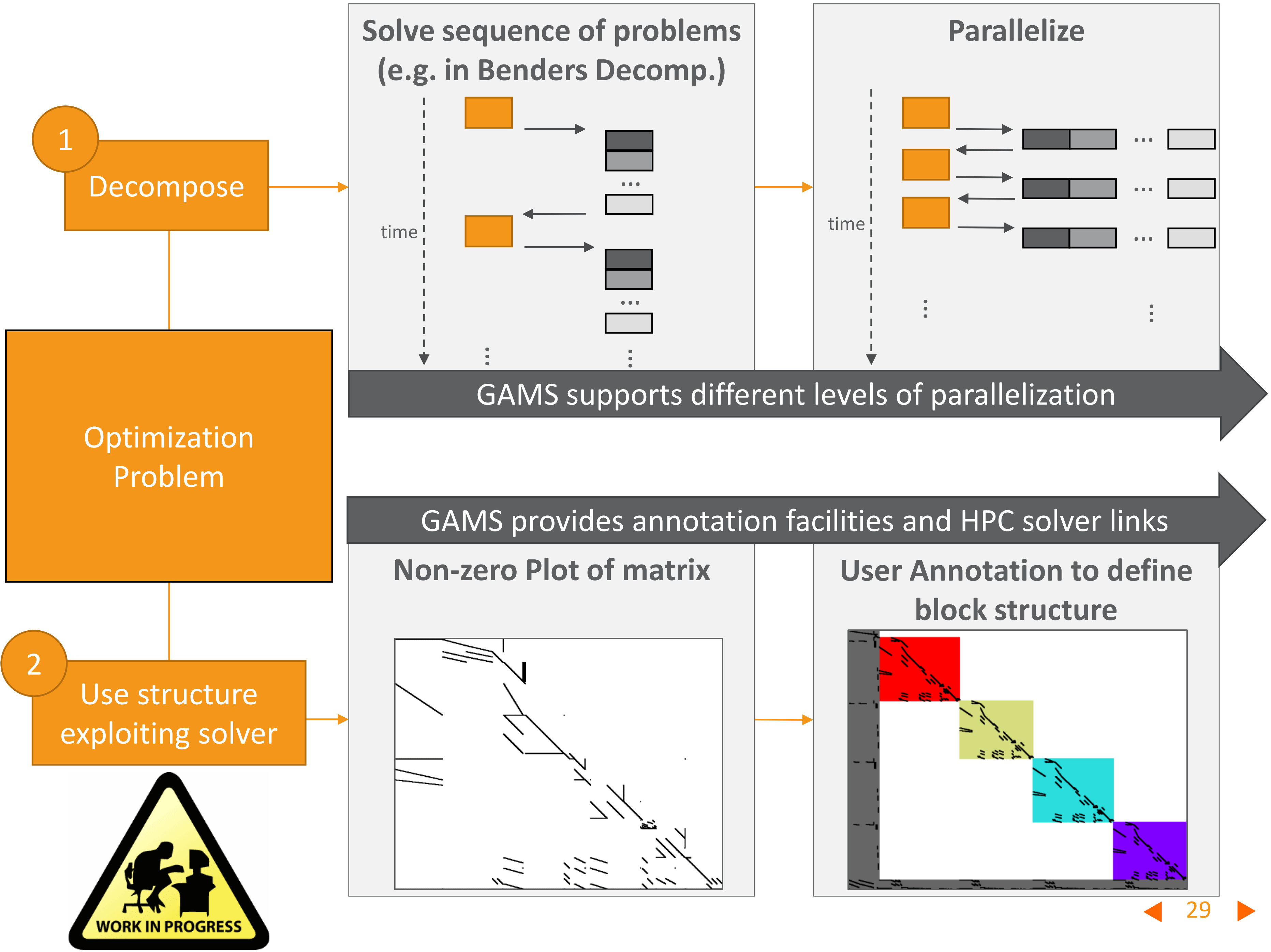
All runs were made with GAMS 25.1.2 on JURECA@JSC with 24 cores per node, 2.5 GHz, (Intel Xeon E5-2680 v3 Haswell), 128 GB RAM

1: single node, 16 cores, CPLEX barrier, no crossover

2: single node, 4 cores per solve statement, CPLEX barrier, advind 0

3: 17 nodes, 404 cores in total, 4 cores per solve statement, CPLEX barrier, advind 0


Solve sequence of problems
(e.g. in Benders Decomp.)
Parallelize
1
Decompose
time
time
Optimization
Problem
GAMS supports different levels of parallelization


1
Decompose
Optimization Problem
2
Use structure exploiting solver
WORK IN PROGRESS
Solve sequence of problems (e.g. in Benders Decomp.)
time
Parallelize
time
GAMS supports different levels of parallelization
GAMS provides annotation facilities and HPC solver links
Non-zero Plot of matrix
User Annotation to define block structure

PIPS-IPM[1,2]

Consider LP with block-diagonal structure, linking constraints, and linking variables (the kind of problem we want to solve):

$$
\begin{array}{llllll}
\min & \sum_{i=0}^{N} c_i^T x_i & & & & \\
\text{s.t.} & T_0 x_0 & & & & = b \\
& T_1 x_0 + W_1 x_1 & & & & = h_1 \\
& T_2 x_0 + & W_2 x_2 & & & = h_2 \\
& \vdots & & \ddots & & \vdots \\
& T_N x_0 + & & & W_N x_N & = h_N \\
& F_0 x_0 + F_1 x_1 & + \quad F_2 x_2 & \cdots & F_N x_N & = g
\end{array}
$$

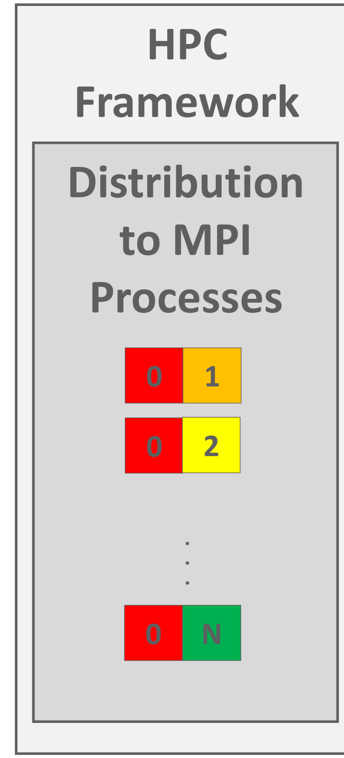
[1] Petra et al. 2014: *"Real-Time Stochastic Optimization of Complex Energy Systems on High-Performance Computers"*

[2] Breuer et al. 2017: *"Optimizing Large-Scale Linear Energy System Problems with Block Diagonal Structure by Using Parallel Interior-Point Methods."*

PIPS-IPM[1,2]

Consider LP with block-diagonal structure, linking constraints, and linking variables (the kind of problem we want to solve):

$$
\begin{array}{lllllllll}
\min & \sum_{i=0}^{N} c_i^T x_i & & & & & & & \\
\text{s.t.} & T_0 x_0 & & & & & & & = b \\
& T_1 x_0 & + & W_1 x_1 & & & & & = h_1 \\
& T_2 x_0 & + & & & W_2 x_2 & & & = h_2 \\
& \vdots & & & & & \ddots & & \vdots \\
& T_N x_0 & + & & & & & W_N x_N & = h_N \\
& F_0 x_0 & + & F_1 x_1 & + & F_2 x_2 & \cdots & F_N x_N & = g
\end{array}
$$

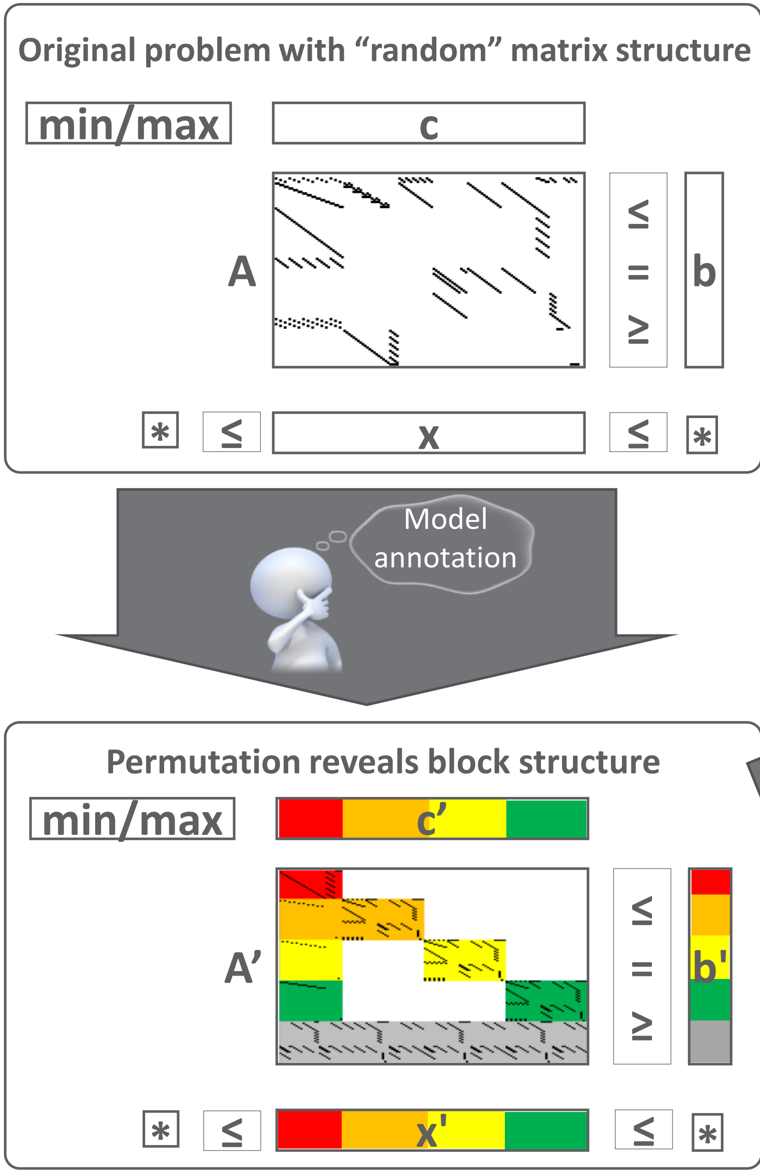


- Block diagonal structure allows parallelization of linear algebra within PIPS-IPM
- Solve N systems of linear equations in parallel instead of one huge system

[1] Petra et al. 2014: *"Real-Time Stochastic Optimization of Complex Energy Systems on High-Performance Computers"*

[2] Breuer et al. 2017: *"Optimizing Large-Scale Linear Energy System Problems with Block Diagonal Structure by Using Parallel Interior-Point Methods."*

GAMS/PIPS-IPM Solver Link - Overview

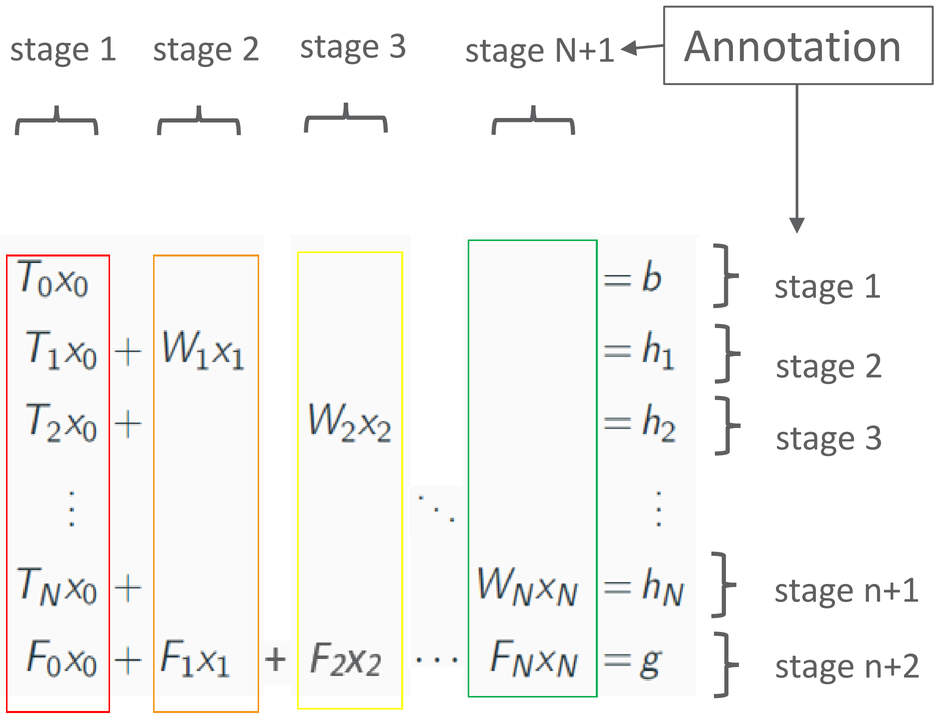


$$
\begin{array}{lllllllll}
\min & \sum_{i=0}^{N} c_i^T x_i & & & & & & & \\
\text{s.t.} & T_0 x_0 & & & & & & & = b \\
 & T_1 x_0 & + & W_1 x_1 & & & & & = h_1 \\
 & T_2 x_0 & + & & & W_2 x_2 & & & = h_2 \\
 & \vdots & & & & & \ddots & & \vdots \\
 & T_N x_0 & + & & & & & W_N x_N & = h_N \\
 & F_0 x_0 & + & F_1 x_1 & + & F_2 x_2 & \cdots & F_N x_N & = g
\end{array}
$$

Model Annotation

Model Annotation by .stage attribute

Matrix structure required by PIPS API

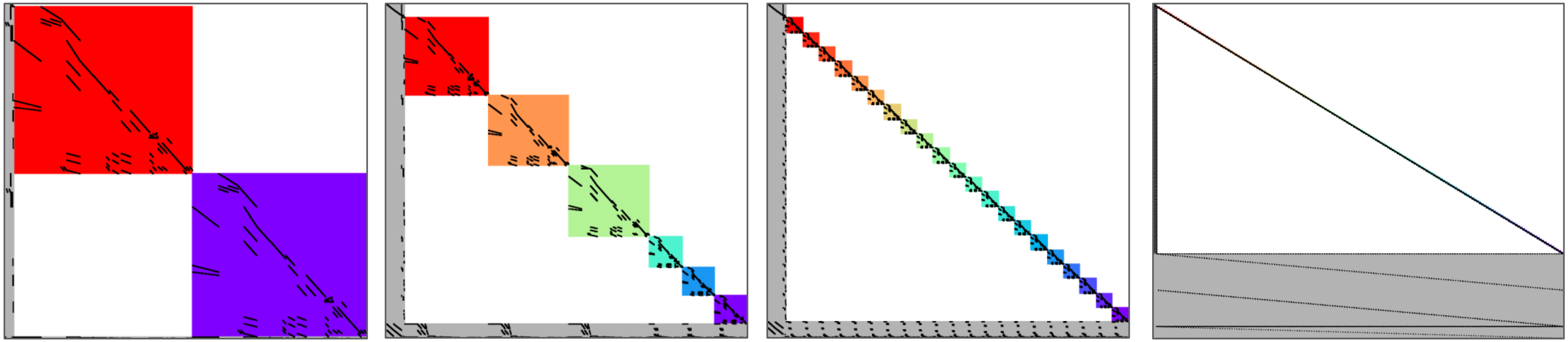


Exemplary Annotation for simple energy system model (regional decomposition)

```
[...]
* Master variables and equation
FLOW.stage(t,net(rr1,rr2))         = 1;
LINK_ADD_CAP.stage(net(rr1,rr2))  = 1;
[...]
* Block variables and equations
POWER.stage(t,rp(rr,p))      = ord(rr)+1;
EMISSION_SPLIT.stage(rr,e)   = ord(rr)+1;
[...]
eq_power_balance.stage(t,rr)     = ord(rr)+1;
eq_emission_region.stage(rr,e)   = ord(rr)+1;
eq_emission_cost.stage(rr,e)     = ord(rr)+1;
[...]
* Linking Equation
eq_emission_cap.stage(e)  = card(rr)+2;
```

Model Annotation cont.

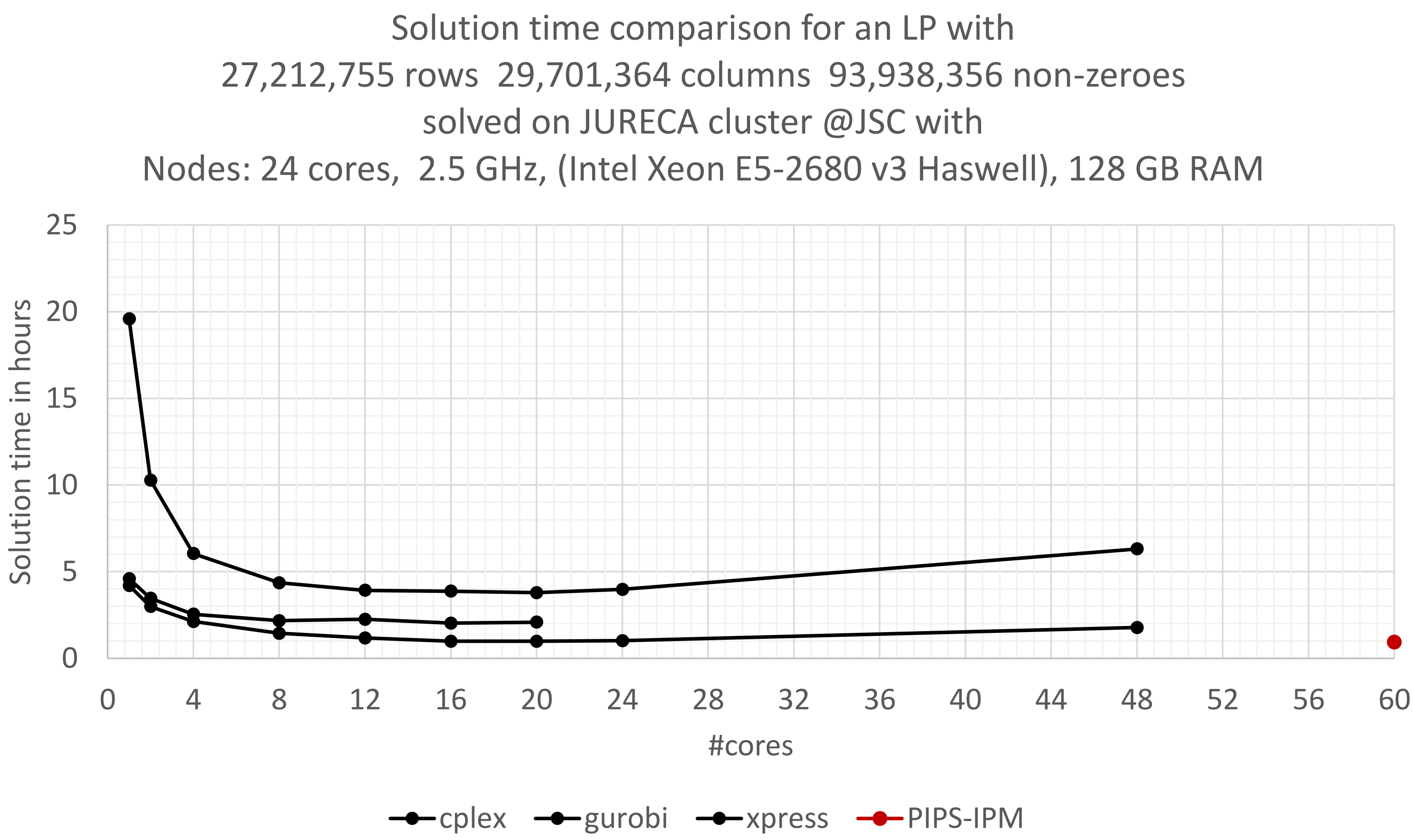
- How to annotate Model depends on how the model should be “decomposed” (by region, time,...)



Plots show four different annotations of identical model

- Blocks of equal size are beneficial

Computational Result(s)



Summary / Outlook

Summary

- Before thinking of HPC, model should be brought “in shape” and capabilities of “standard” hardware should be exploited
- GAMS & Solvers provide useful hints on efficient model formulation
- GAMS provides broad set of parallelization facilities
- HPC Capabilities of GAMS can be easily extended via embedded Python Code (Parallel Benders with mpi4py)
- Annotation Facilities to allow users the definition of block structures are available
- Link to HPC solver PIPS-IPM available

Outlook

- Embedded Python Code in combination with GAMS Python OO API allows to further increase efficiency (e.g. via GAMS ModelInstances, Warmstarts, ...)
- Parallelization can be extended to Model Generation
  - Usual Model": model generation time << solver time
  - For LARGE-scale models the model generation may become significant:
    - due to time/memory consumption
    - due to hard coded limitations of model size (# non-zeroes < ~2.1e9)
  - Generation of separate model blocks as required by solver
    - Fully implemented by user: possible (significant refactorization of code)
    - Annotation provided by user → block sharp generation by GAMS: work in progress
- Additional HPC Solver Link to OOPS[1,2] is currently under development

[1]: **J. Gondzio and R. Sarkissian**, Parallel Interior Point Solver for Structured Linear Programs, *Mathematical Programming* **96** (2003) No 3, 561-584.
[2]: **J. Gondzio and A. Grothey**, Reoptimization with the Primal-Dual Interior Point Method, *SIAM Journal on Optimization* **13** (2003) No 3, pp. 842-864.

GAMS

Frederik Fiand

Operations Research Analyst

GAMS Software GmbH

ffiand@gams.com