Internationale Energiewirtschaftstagung (IEWT)
Vienna, Austria

GAMS

Solving Large-Scale Energy System Models

Hermann von Westerholt
Technical Sales Engineer
GAMS Software GmbH

Agenda

1. GAMS – Company Overview
2. BEAM-ME – Project Background
3. BEAM-ME – High-Performance-Computing
4. Summary

GAMS

Company Overview

Company History

- Roots at World Bank (1976)
- went commercial in 1987
- Locations:
  - GAMS Development Corp. (Fairfax, USA)
  - GAMS Software GmbH (Germany)
- Product: The **G**eneral **A**lgebraic **M**odeling **S**ystem

GAMS at a Glance

- High-level algebraic modeling language
- Focus lies on modeler
- All major solvers available (30+ integrated)
- Used in more than 120 countries (research and production)

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

BEAM-ME

Project Background

The BEAM-ME Project

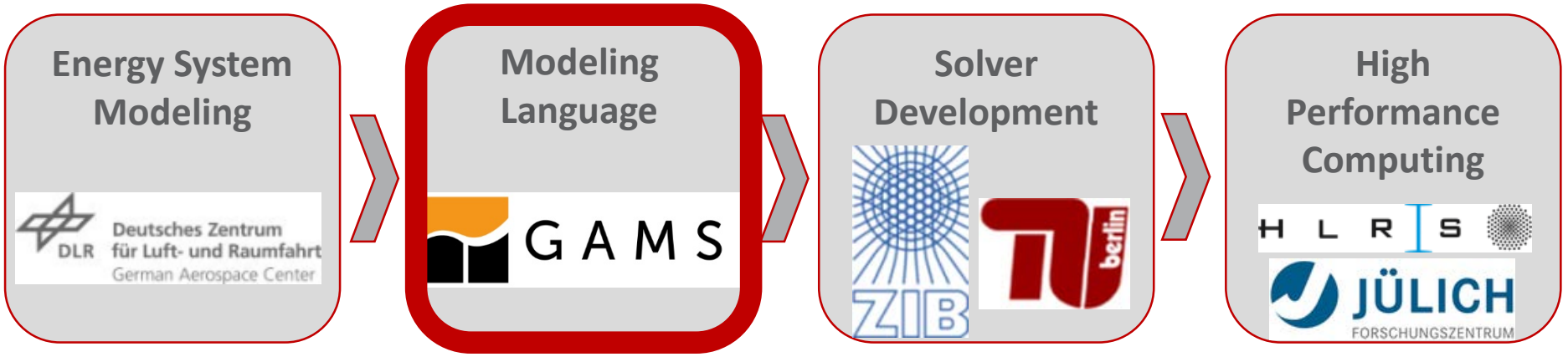
Supported by:
Federal Ministry for Economic Affairs and Energy
on the basis of a decision by the German Bundestag

GAMS

What exactly is BEAM-ME about?

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

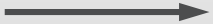
An Interdisciplinary Approach:



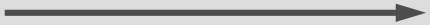
Model Parameters that Drive Complexity

Time

Planning Horizon



short term



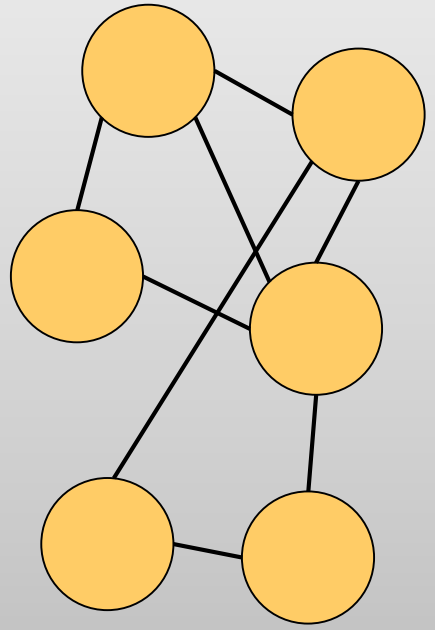
long term

Discretization

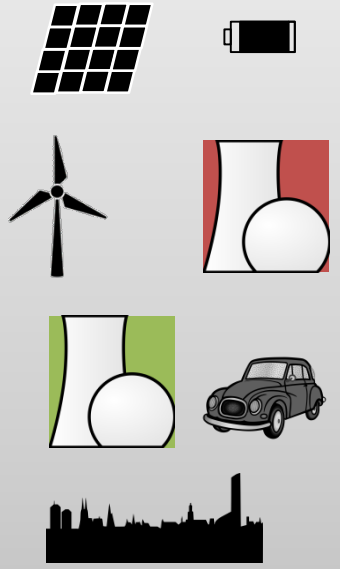
coarse

fine

Regional Aggregation



Technology Parameters



(Very-) Large-scale LP

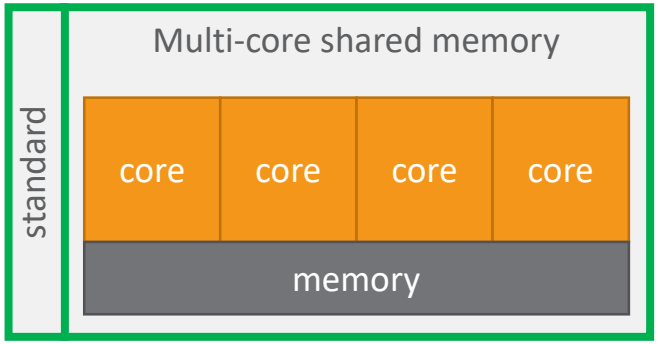
- Scalable (resolution time, space, and technology)
- Block structure

BEAM-ME

High-Performance-Computing: An Example

Available Computing Resources

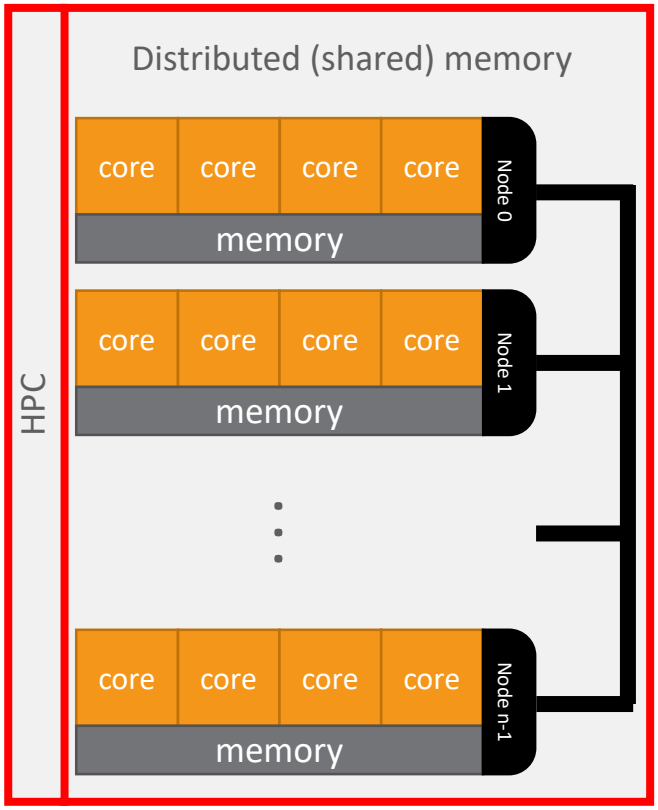
GAMS



Convenient to use.

Capabilities of standard hardware should be exploited first.

CHEAP



Complex to use.

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

EXPENSIVE

JUWELS at Jülich Supercomputing Centre

Hardware characteristics

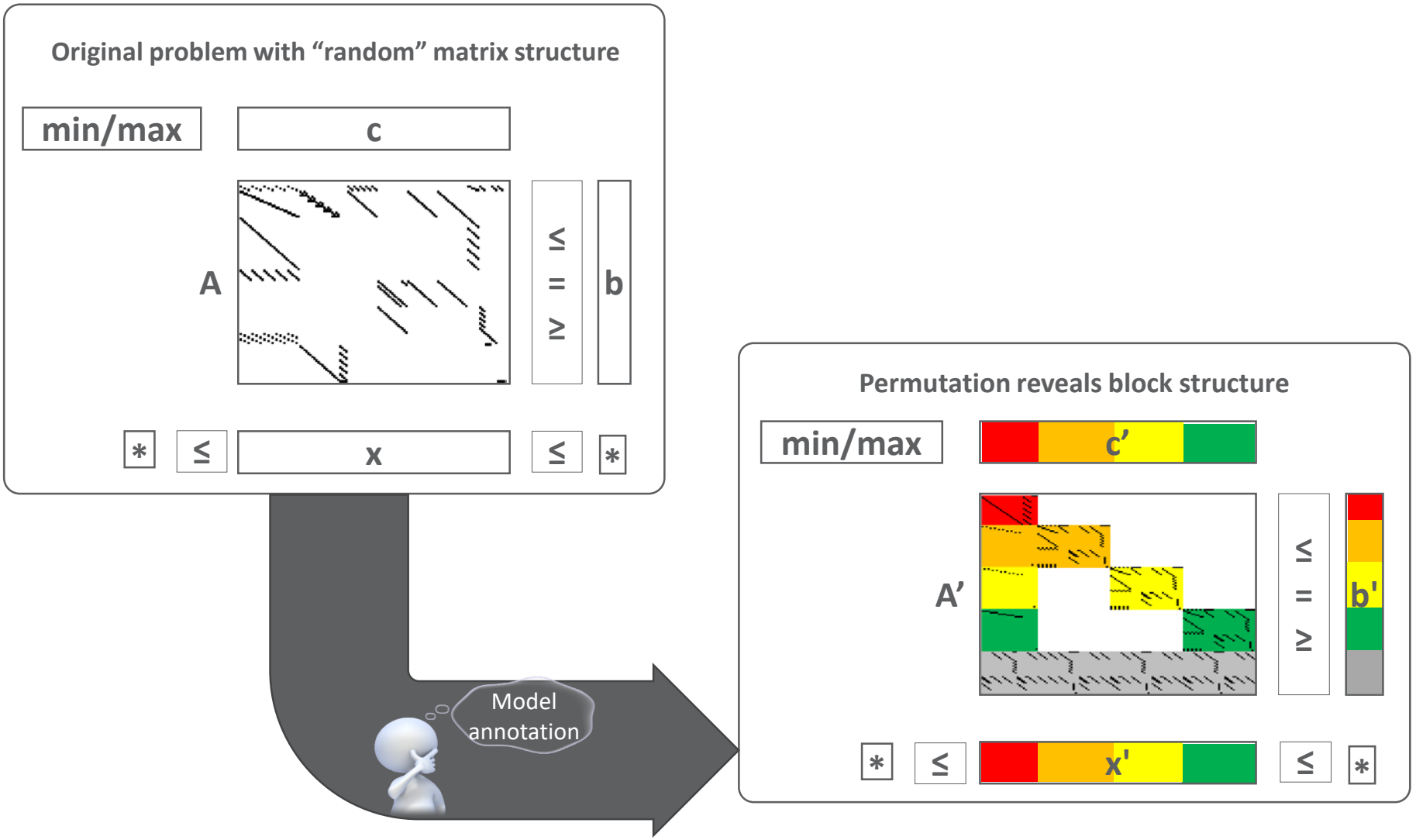
- 2271 standard compute nodes
  - 2x24 cores, 2.7 GHz
  - 12x8 GB, 2666 MHz
- …

Also tested on other target platforms at JSC Jülich and HPC center Stuttgart

GAMS/PIPS-IPM Solver Link Overview

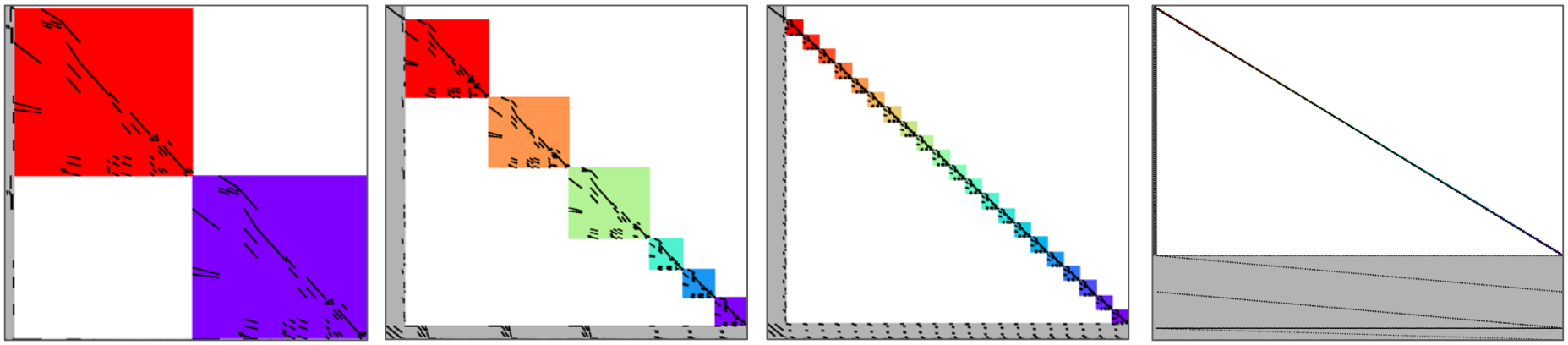
- **P**arallel **I**nterior-**P**oint **S**olver for LPs (and QPs), designed for high-performance computing platforms
- Originally developed for stochastic problems by Cosmin Petra (Argonne National Lab)
- Had already been applied to very-large-scale problems
- extension to support linking constraints implemented by ZIB

GAMS/PIPS-IPM Solver Link
How it Works



Model Annotation cont.

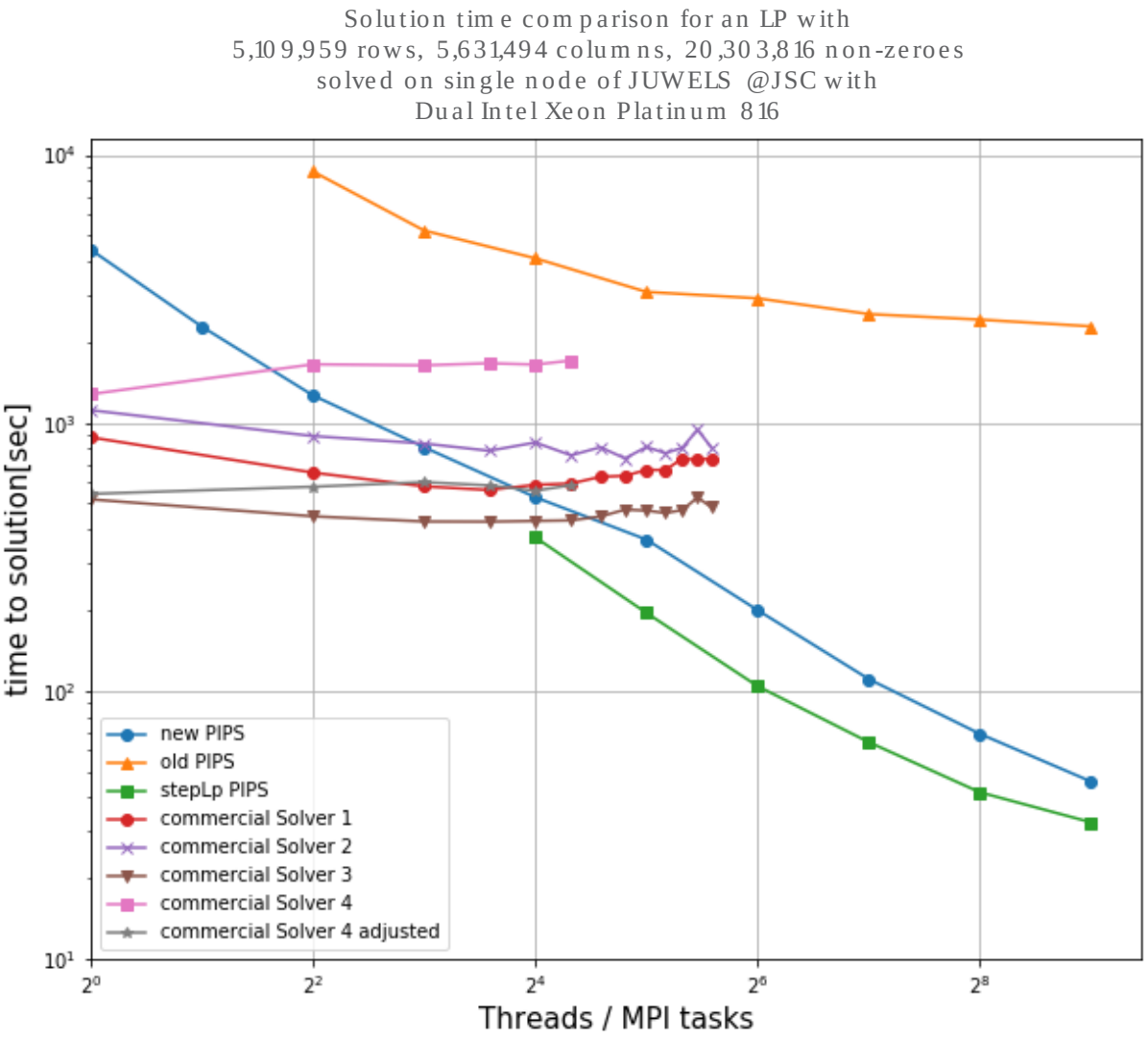
- How to annotate Model depends on how the model should be "decomposed" (by region, time,..)



Plots show four different annotations of identical model

- Blocks of equal size are beneficial

Computational Result(s)



Summary

Summary

- Increasing complexity makes solving ESM more difficult
- Conventional solution strategies at their limits, new approaches needed
- Before thinking of HPC, model should be brought “in shape” and capabilities of “standard” hardware should be exploited
- Annotation Facilities to allow users the definition of block structures are available
- PIPS-IPM is open source, but hardware is expensive
- Currently: user knowledge required in order to fully exploit HPC capabilities

GAMS

Thank you for your kind attention

Hermann von Westerholt

Technical Sales Engineer

GAMS Software GmbH

hwesterholt@gams.com

Backup Slides

Motivation

- Energy system models (ESM) have to increase in complexity to provide valuable quantitative insights for policy makers and industry:
  - Uncertainty
  - Large shares of renewable energies
  - Complex underlying electricity systems

- **Challenge:**
  - Increasing complexity makes solving ESM more and more difficult

→Need for new solution approaches

→Exploit the parallel computing of modern HPC systems

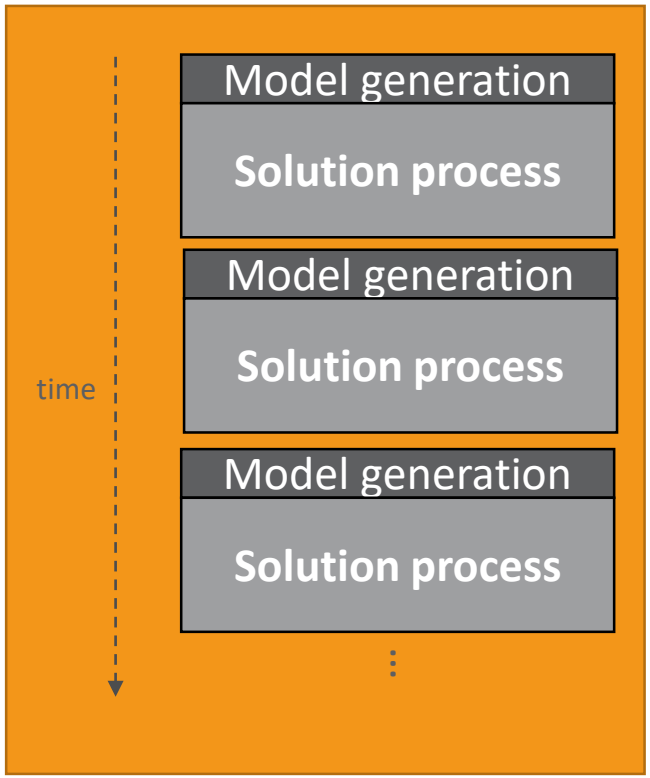
Parallelization with GAMS

From simple sequential to highly parallel solve statements

Sequential Solve Statements in Loops

- loop body code in sequence, often with an expensive solve statement:

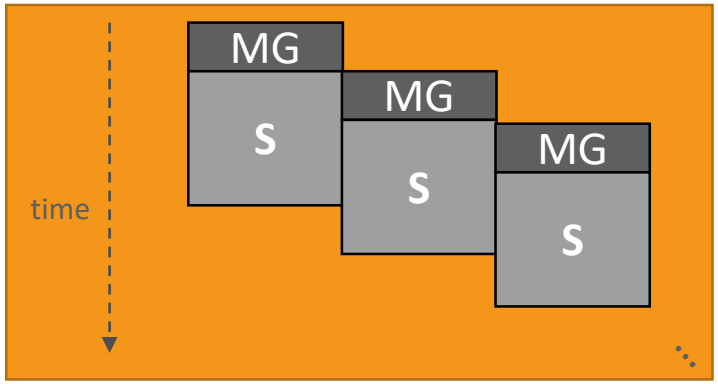
```
... // preparatory work
loop(scen,
    ... // setup model
    option clear=s; s(scen) = yes;
    solve mymodel min obj using minlp;
    ... // store results
);
... // reporting
```



Parallel Solves – GAMS Grid Facility

- SolveLink option specifies the solver linking conventions
- Split loop in submission & collection loop:

```
... // preparatory work
parameter h(scen); mymodel.solvelink=%solvelink.async...%;
loop(scen,
    ... // setup model
    option clear=s; s(scen) = yes;
    solve mymodel min obj using minlp;
    h(scen) = mymodel.handle;
);
repeat
    loop(scen$handlecollect(h(scen)),
        ... // store results
        h(scen) = 0;
    );
until card(h)=0;
... // reporting
```



- Model generation and loop body code in sequence
- Either file based IO or limited to shared memory

Excursus: Embedded Code Facility

GAMS

24.9.1 Major release (August 30, 2017)

Acknowledgments

We would like to thank all of our users who have reported problems and made suggestions for improving this release. In particular, we thank Etienne Ayotte-Sauvé, Wolfgang Britz, Florian Habermacher, Florian Häberlein, Maximilian Held, Ignacio Herrero, Hanspeter Höschle, Erwin Kalvelagen, Toni Lastusilta, John Ross, and Tom

GAMS System

GAMS

- New feature, the **Embedded Code Facility**: This extends the connectivity of GAMS to other programming languages. It allows the use of Python code during compile and execution time. GAMS symbols are shared with the external code, so no communication via disk is necessary.
  The embedded code feature is available on Linux, MacOS X, and Windows. For these platforms, a Python 3.6 installation is included with the GAMS distribution. If the user wants to work with a different Python 3.6, installed separately, for models with embedded code the new command line option **pySetup** needs to be set to 0.

  Note
  This feature is currently in beta status. Any feedback to support@gams.com is appreciated.
- New command line option **procDirPath**: Specifies the directory where the process directory should be created.

Example: Sequential Benders Decomposition

```
set scen    'scenario set' / scen1*scen100 /
    s(scen) 'dynamic secenario subset'
    k       'benders iterations' / k1*k1000 /;

... // 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
    option clear=s; 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

GAMS

PMI_RANK=0

```
set scen    'scenario set' / scen1*scen100 /
    s(scen) 'dynamic secenario subset'
    k       'benders iterations' / k1*k1000 /;

...
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 scen    'scenario set' / scen1*scen100 /
    s(scen) 'dynamic secenario subset'
    k       'benders iterations' / k1*k1000 /;

...
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
```

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