6th International Conference
"Distributed Computing and Grid-technologies in Science and Education"

High-level optimization modeling software in distributed computing environment

Vladimir V. Voloshinov, Smirnov S.A.

Supported by the Russian Foundation for Basic Research (grant # 13-07-00987)

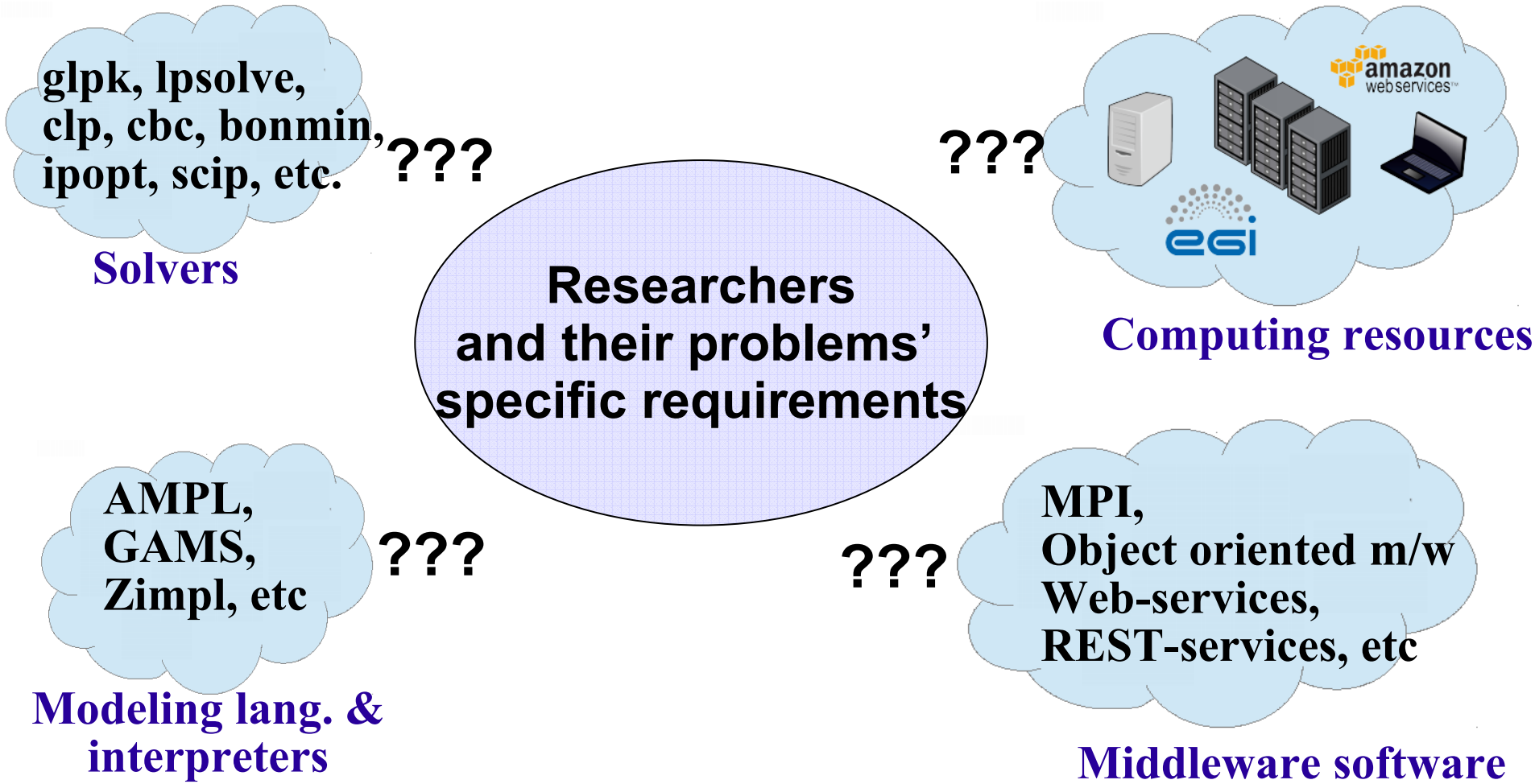
Center of Grid-technologies & Distributed Computing,
Institute for Information Transmission Problems RAS, Moscow

JINR, Dubna, 2014

Optimization & distributed computing

Typical problems:

- **to increase computing power of available and/or emerging solvers by available computing environment**
- **to provide convenient interface for researchers**



Solvers for optimization problems (open source)

GLPK (LP, MILP), A. Makhorin, RU, since ~2002

LP_SOLVE, (LP, MILP) Eindhoven University of Technology, NL, since ~2000

COmputational INfrastructure for Operations Research, www.coin-or.org ("IBM's aegis"), more than 40 solvers&libs: since ~2005

CLP (LP), CBC (MILP),
Ipopt (NLP),
Bonmin/CBC/Ipopt (MINLP, convex on all variables)

SCIP (LP, MILP, MIQP, MINLP of some types, e.g. polynomial), Zuse Institute Berlin, DE, ver. 1.0 at 2007

BnB-solver (MILP, global optimization), M. Posypkin, IITP RAS, RU

Incomplete list of those solvers we used in our researches

Existing approach - usage of AML-system

AML - Algebraic Model Languages (AMPL, GAMS, Zimpl, etc).

Common features:

- Convenient (symbolic "TeX-like") description of object & constraints functions
- Separation of "symbolic/abstract" models and numerical data for multivariate computation (parameter sweeping)
- Automatic differention (Jacobian & Hessian)
- Support of "Lagrangian formalism" - access to optimal variables and duals found by solver
- Unified open-source (even for "commercial" AMLs) API for solvers' and applications' developers

Usage of AMLs is crucial at preliminary phases of R&D

There are a number of AMLs

Incomplete list:

AMPL - A Modeling Language for Mathematical Programming,
AT&T Bell Laboratories, D.M. Gay, Brian W. Kernighan,
since 1980-x, http://www.ampl.com

GAMS - General Algebraic Modeling System,
International Bank for Reconstruction and Development,
since 1980-x, http://www.gams.com

OPL - Optimization Programing Lang., IBM,
ILOG CPLEX (LP, QP, ...), CP Optimizer, http://www-01.ibm.com/

GNU MathProg - "subset" of AMPL for GLPK, GNU LP Kit,
Andrey Makhorin, MAI,
since 2000, http://www.gnu.org/software/glpk/

Zimpl - since 2004, http://zimpl.zib.de/ (LP, MILP, NLP ?)
Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)

General scheme of AMLs usage

"Symbolic" opt. model

$$f_o(p,x) \to \min_x,$$
$$f_i(p,x) \leqslant 0 \quad (i \in I),$$
$$g_j(p,x) = 0 \quad (j \in J)$$
$$x \in M$$

file *.mod

Parameters' values

```
param p : ...;
set I : ...;
set J : ...;
```

file *.dat

AML-script

```
model *.mod;
data     *.dat;
option solver ipopt;
solve;
display _var, _dvar;
printf ...
```

Problem's data as a stub file

AMPL "stub", *.nl, *.sol

GAMS data exch., *.gdx

ZIMPL for SCIP

AML-translator

ampl.exe
gams.exe
SCIP

AML API

AMPL/
Solver
interface
Library

GAMS

....

Solvers

CPLEX
Lpsolve
Minos
Knitro
Snopt
Gurobi
Mosek
...

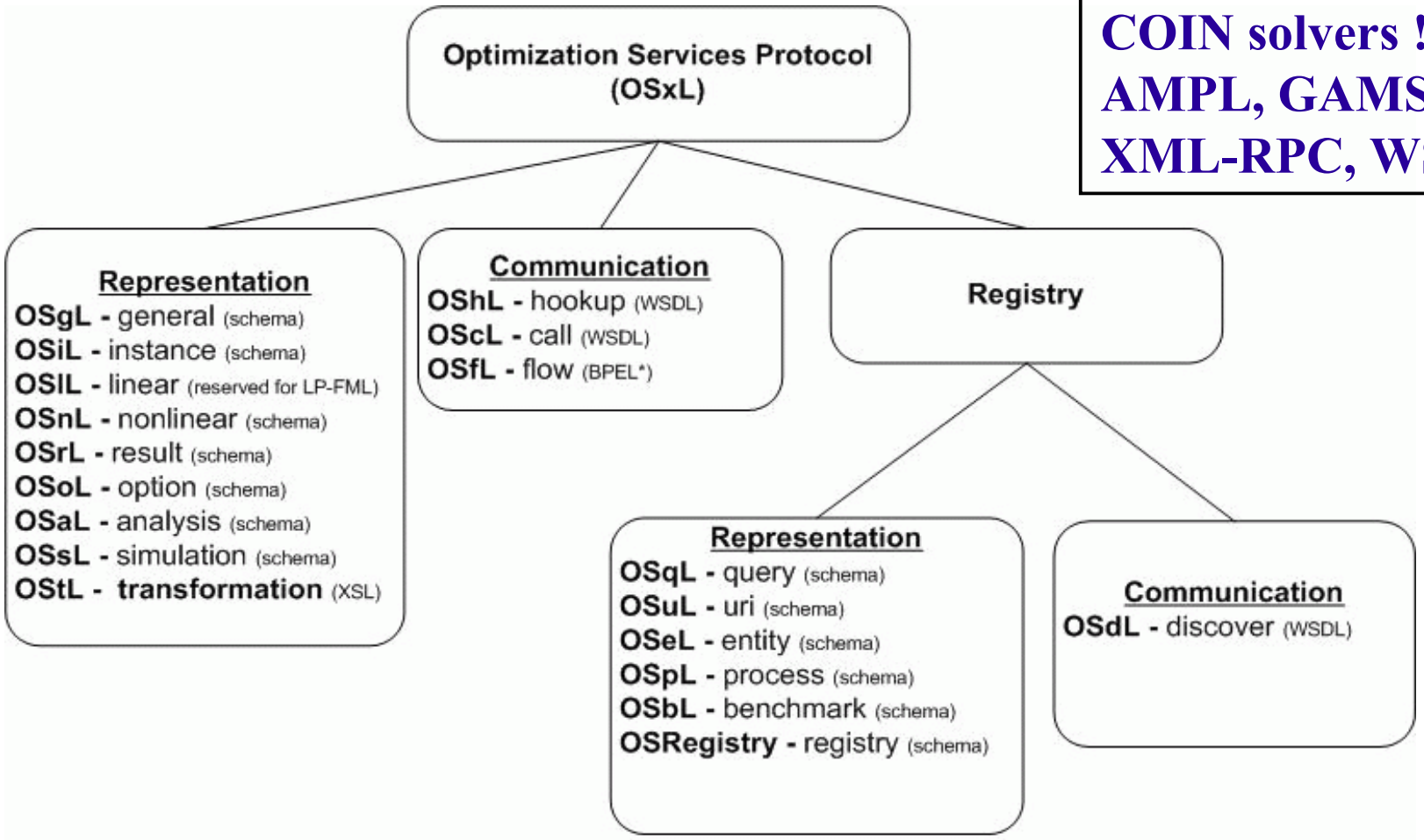
COIN-OP
CLP, CBC,
Ipopt
Bonmin
...

SCIP

AMPL and GAMS -
most popular de-facto standards

Optimization & distributed computing (service-oriented)

Since 2004, project Optimization Services, www.optimizationservices.org, under the aegis of COIN-OR (IBM) www.COIN-OR.org/projects/OS.xml



COIN solvers !!!
AMPL, GAMS - !!!
XML-RPC, WSDL, BPEL - ???

***OSmL:** a modeling language and NOT an Optimization Services Protocol
***BPEL:** Business Process Execution Language for flow orchestration.

Our approach

https://gitlab.com/u/sol, https://mc2.distcomp.org

http://dcs.isa.ru/drupal/ru/development/mathcloud/optimizationServices

REST - as an architectural style

MathCloud - as a middleware and software toolkit

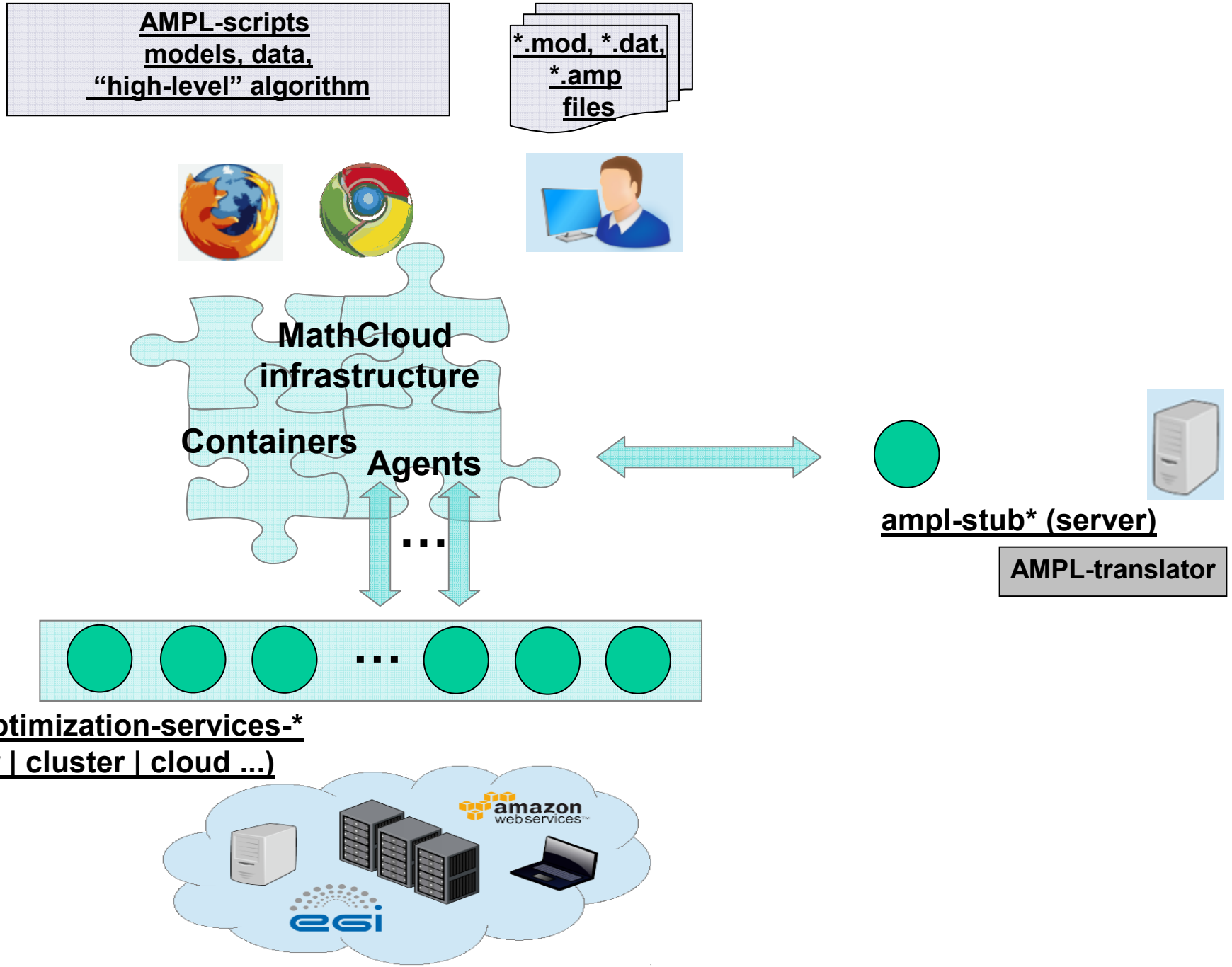
HTTP, JSON (JavaScript Object Notation) as a messages format (plain text), HTML+JavaScript for Web User Interface (WUI)

AMPL - optimization modeling and algorithms (high-level) description

AMPL-compatible solvers CLP, CBC, Ipopt, Bonmin, SCIP (LP/MILP, NLP, MINLP), BnB (MILP, global opt)

Mathcloud Python API, MPI, Erlang (experimental) – for low-level data exchange (solver⇔solver, ampl⇔solver)

Our approach. Use case: solve “separate” problems



Our approach. Use case: run “remote” AMPL-script

AMPL-scripts
models, data,
“high-level” algorithm

*.mod, *.dat,
*.amp
files



MathCloud
infrastructure

Containers

Agents

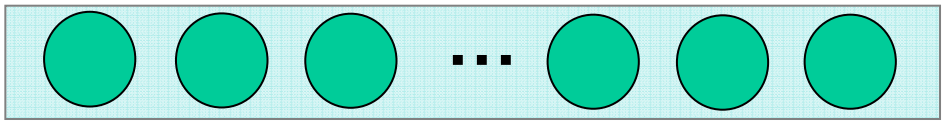
...



ampl-stub*, ampl-exec* (server)

AMPL-translator

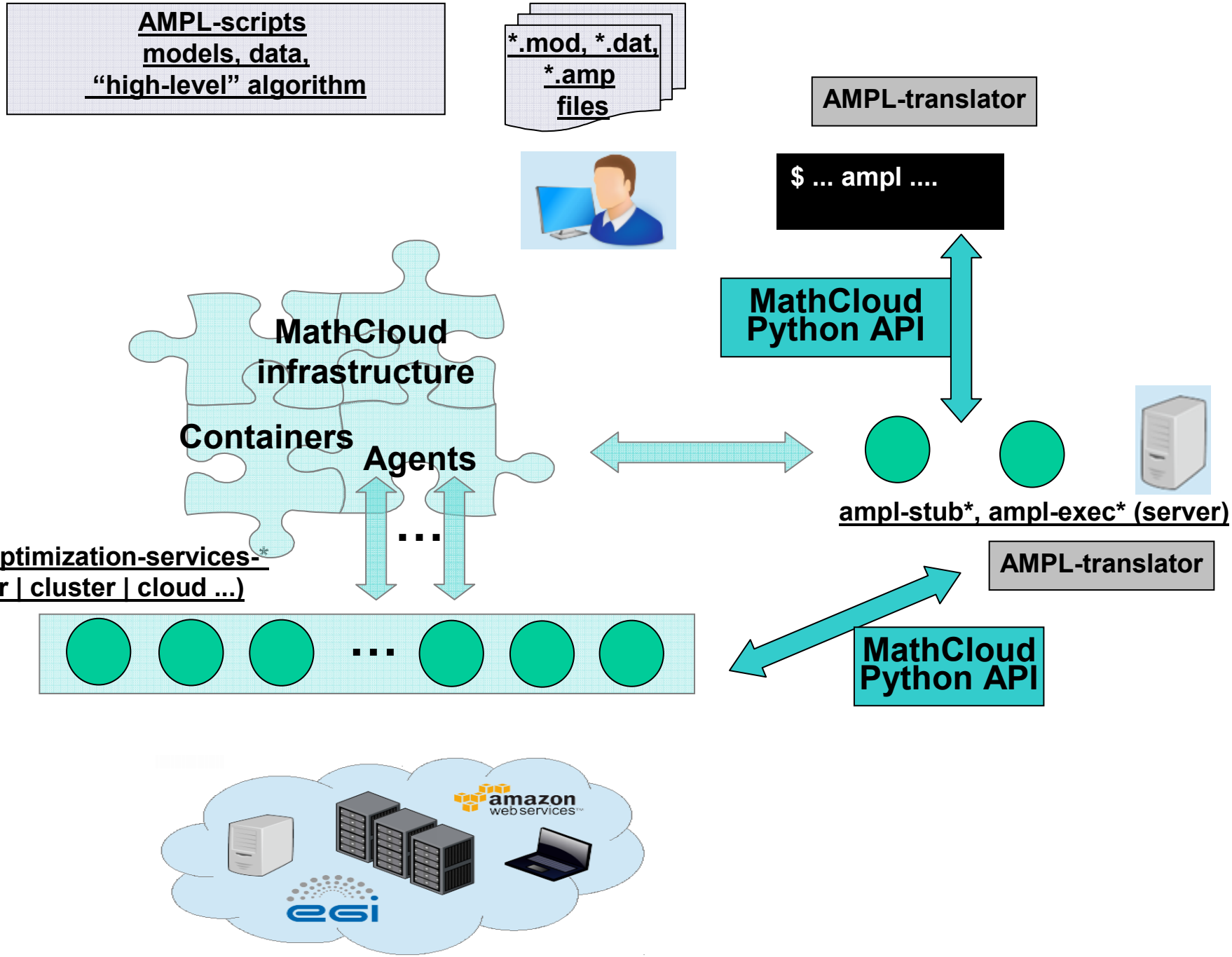
MathCloud
Python API



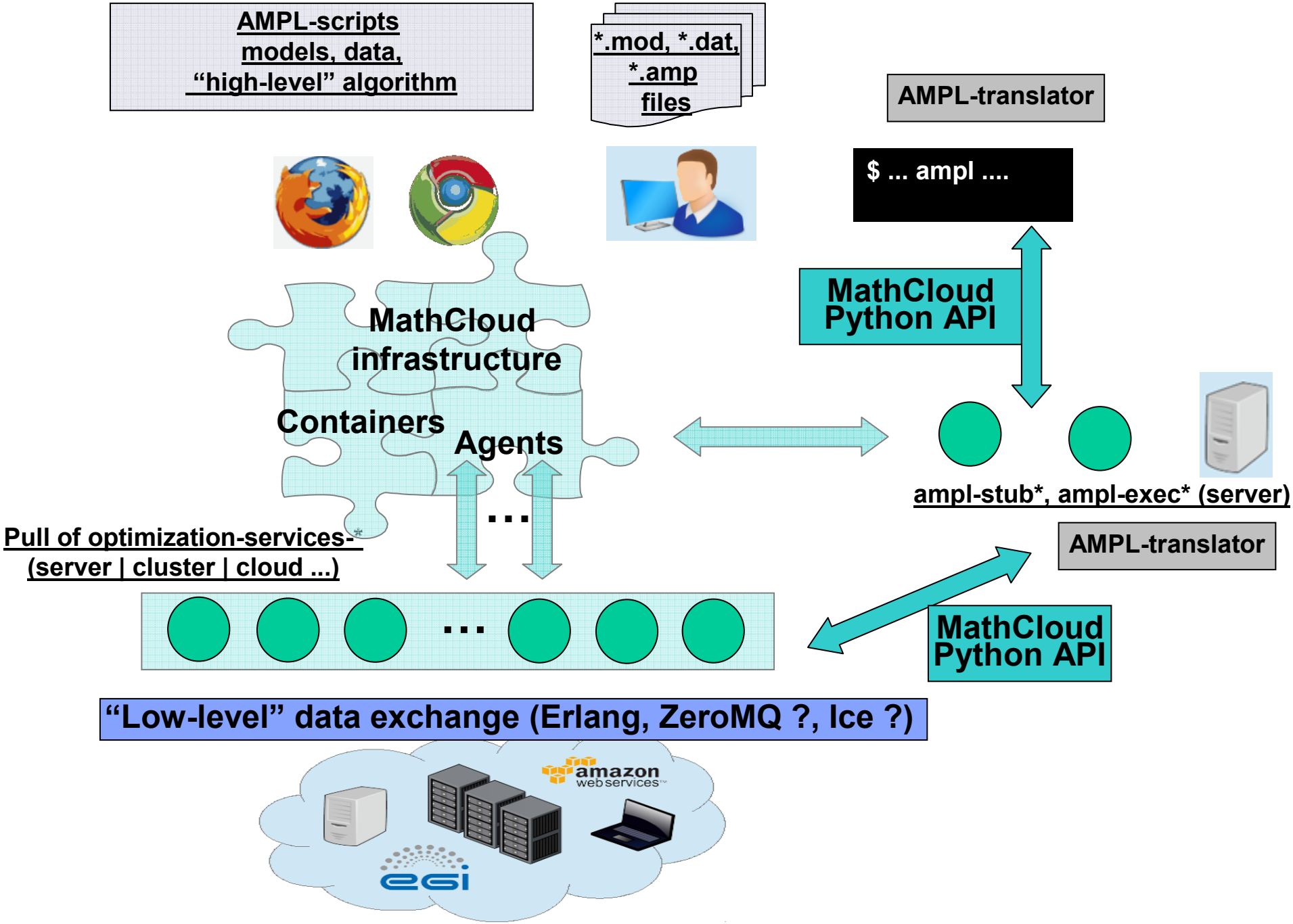
...

Pull of optimization-services-*
(server | cluster | cloud ...)

Our approach. Use case: run “local” AMPL-script “in MathCloud”



Our approach. Use case: additional “low-level” inter-solver data exchange



Our researches on optimization

- Optimal Control Problems with Mixed Control-State constraints
- Global & discrete optimization:
  - knapsack problems,
  - particles spatial optimization (to minimize inter-particle potential energy Lennard-Jones, Morse)
- Global opt. in combinatorial geometry (Tammes problem, 13-Spheres problem
- Crypto-resistance of known algorithms (A5/1)
- Algorithm of amorphous carbonaceous nanomaterial structure identification with a joint X-Ray and neutron diffraction experiments data analysis
- **Miscellaneous experiments with Traveling Salesmen, Task-Scheduling problems, LP with block-structure**

Set of "atomic" RESTful-optimization services

"Atomic" REST-services deployed in a dedicated servers, clusters, cloud by MathCloud Everest agents/container

Prepare input data & processing output (solution):

ampl-stub **- generate AMPL-stub from model and data**

ampl-pre-opt **- more complex AMPL-stub generation (model, data, AMPL-script)**

ampl-post-opt **- processing solution (model, data, solution AMPL-format, AMPL-script)**

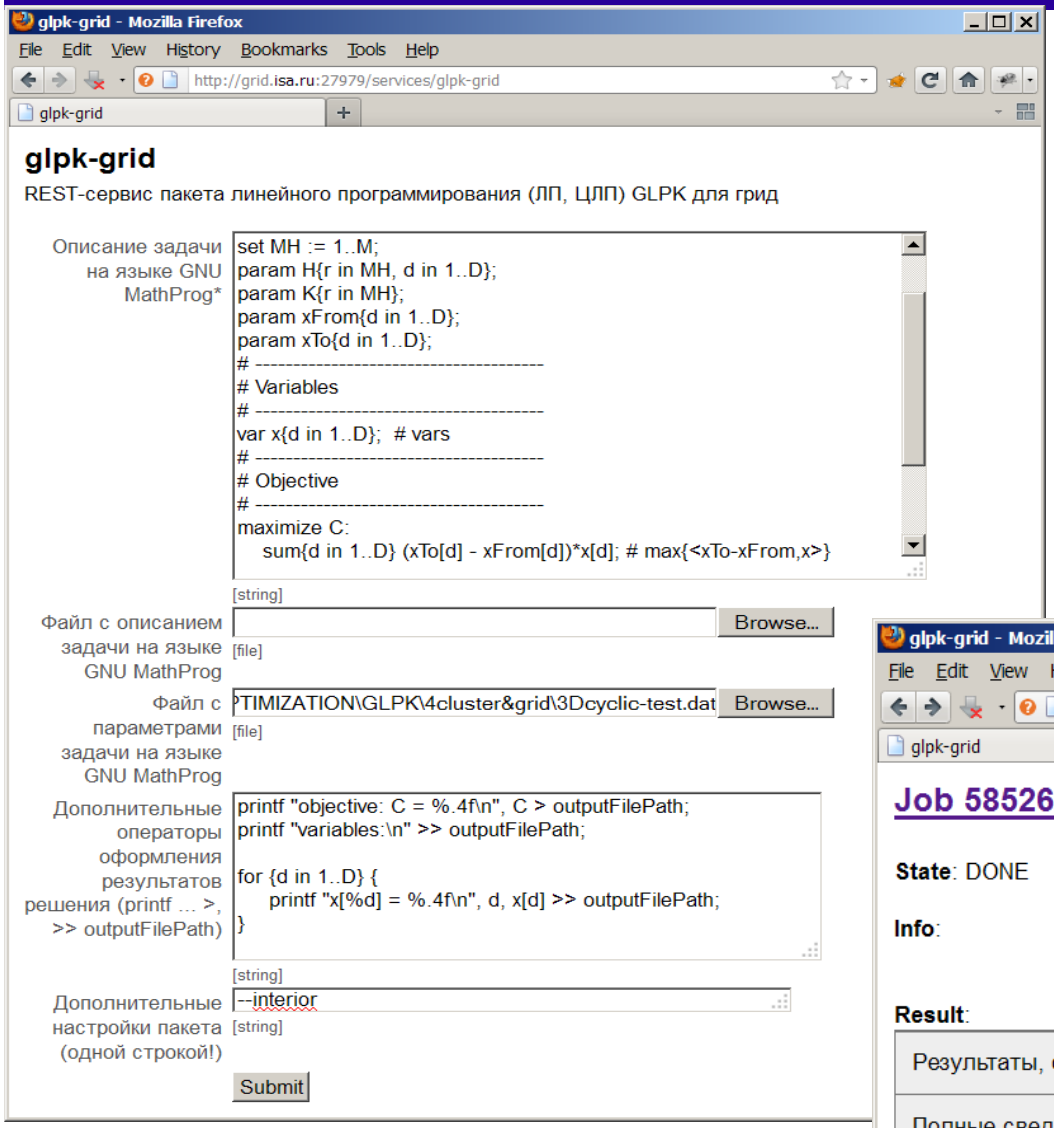
Solver services (via LPSOLVE, CBC/CLP, Ipopt, Bonmin other AMPL-solvers):

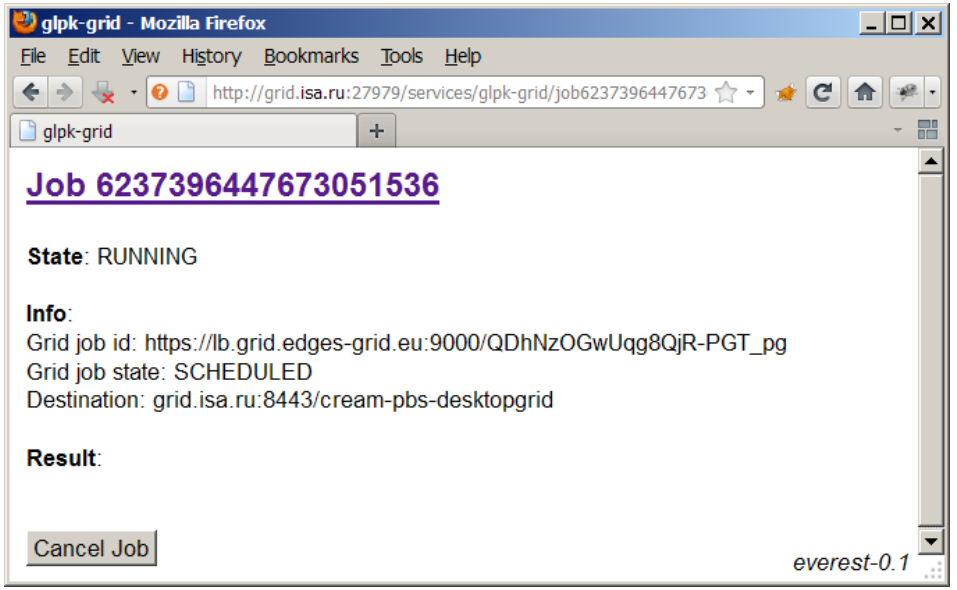
optimization-service-{command | cluster | grid} **- to solve LP/MILP, NLP/MINLP problems presnted by their AMPL-stub, respectively on dedicated server, cluster, grid-node**

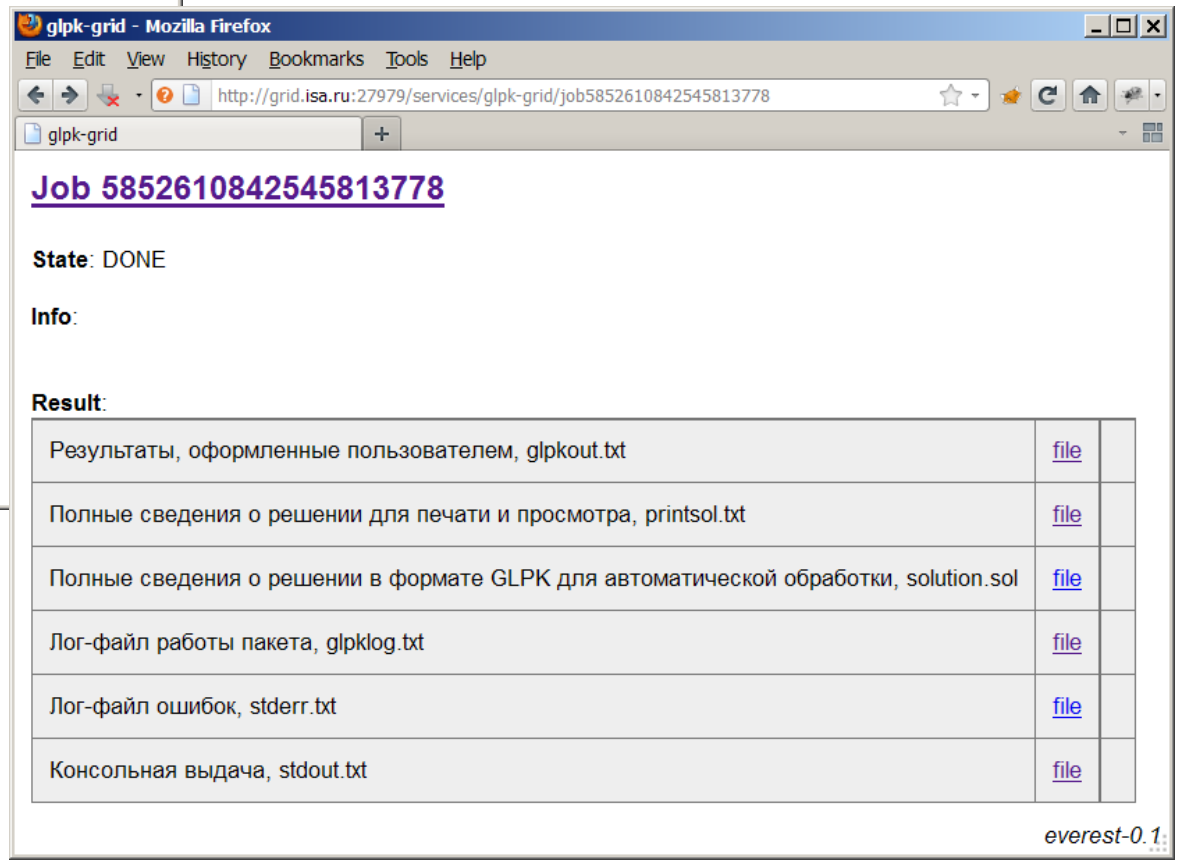
Set of GLPK services (GLPK includes GNU MathProg translator):

glpk-{command | cluster | grid} **- full scheme of optimization: model, data, pre opt. GMP script -> solution -> post opt. GMP respectively on dedicated server, cluster, grid-node**

Web-interface of RESTful-optimization services







Set of "composite" RESTful-optimization services

http://dcs.isa.ru/drupal/ru/development/mathcloud/optimizationServices

Composite services are implemented as WORKFLOWS of atomic ones.

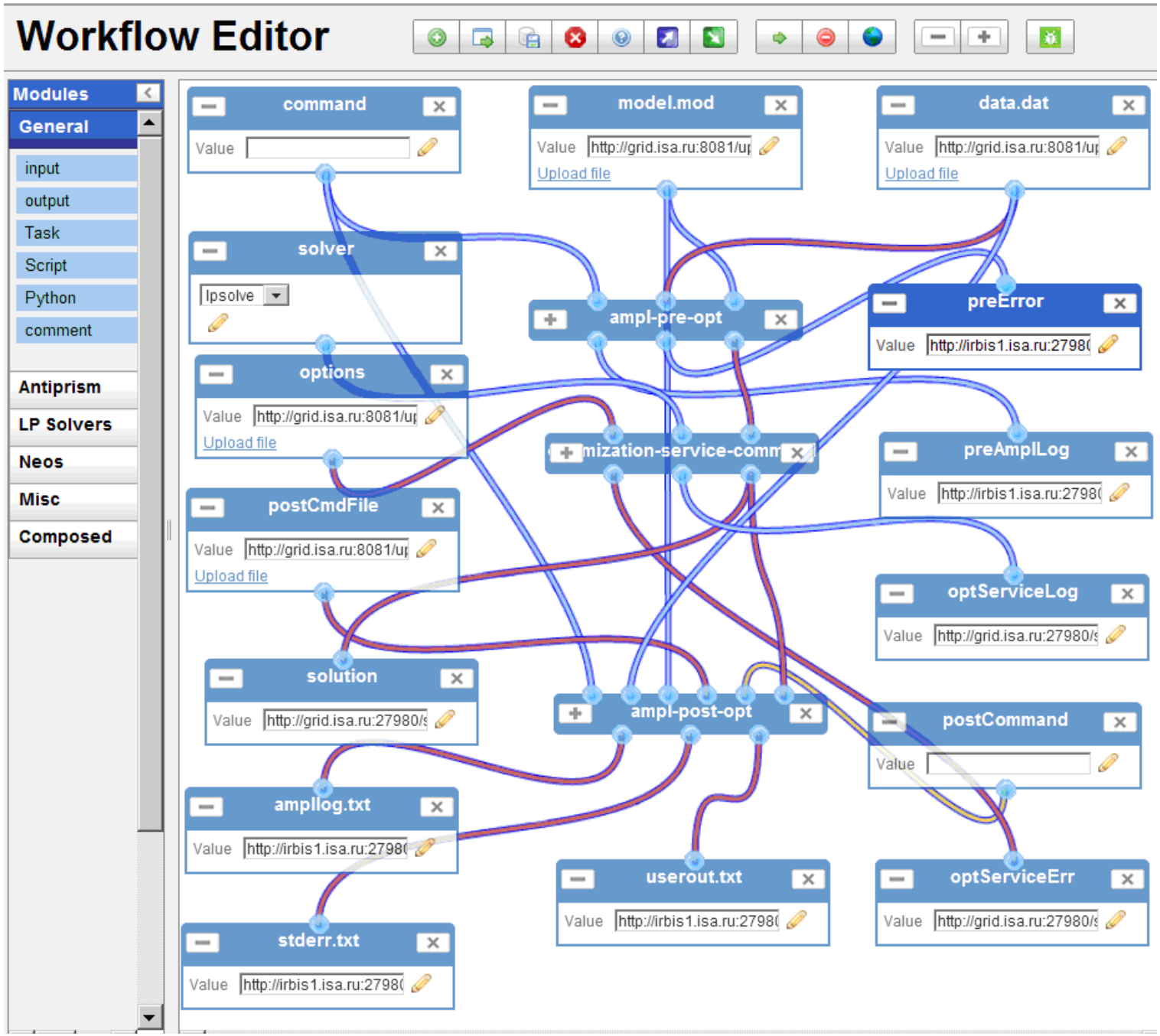
"Full optimization cycle" services (ampl-pre-opt, optimization-*, ampl-post-opt):

***ampl-optimization-service-{command | cluster | grid}* - full scheme of AMPL-optimization:**
model, data, pre opt. AMPL script -> solution -> post opt. AMPL
respectively by solvers on dedicated server, cluster, grid-node

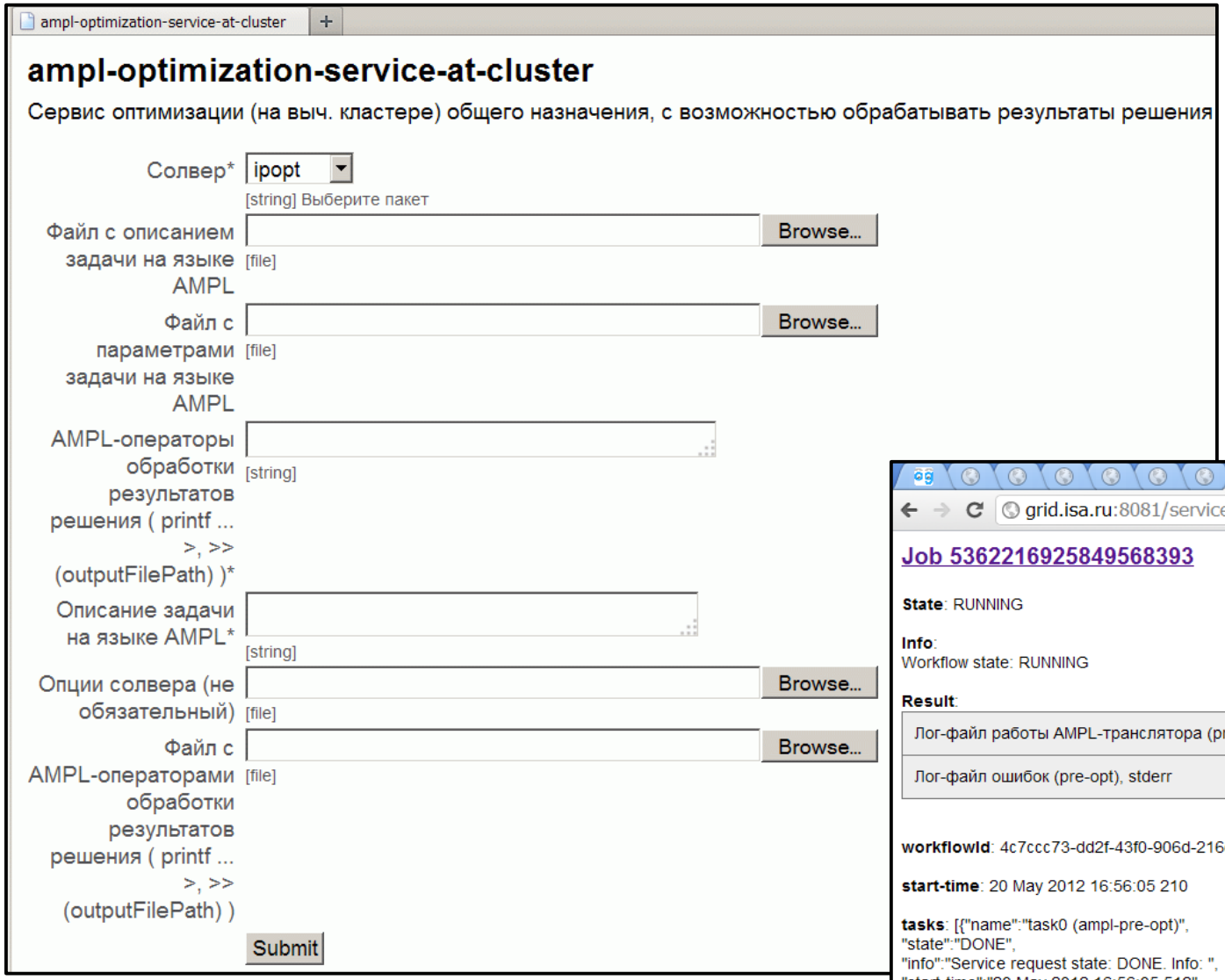
***mcl-control* - "enhanced" AMPL-translator, enables running any (!) AMPL-algorithm in distributed mode;**
all stubs are sent to a pool off *optimization-service-
and solutions are brought back to AMPL (and so on);
Includes simple task manager (Python) for load balance

... **a number of demo & test services**

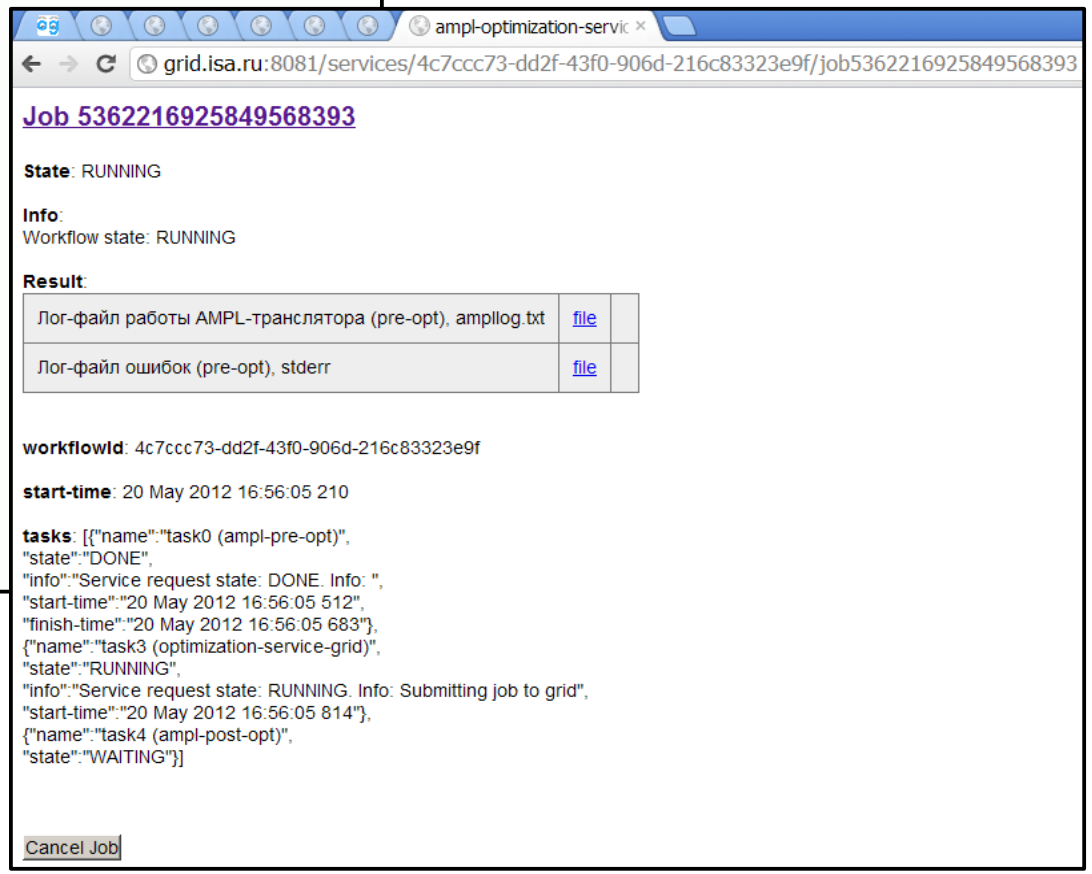
Workflow for ampl-optimization-service-(command). MathCloud WF Editor



Auto generated web-interface for composite ampl-optimization-service-(cluster)



Web-interface "inherits" WUI of atomic services



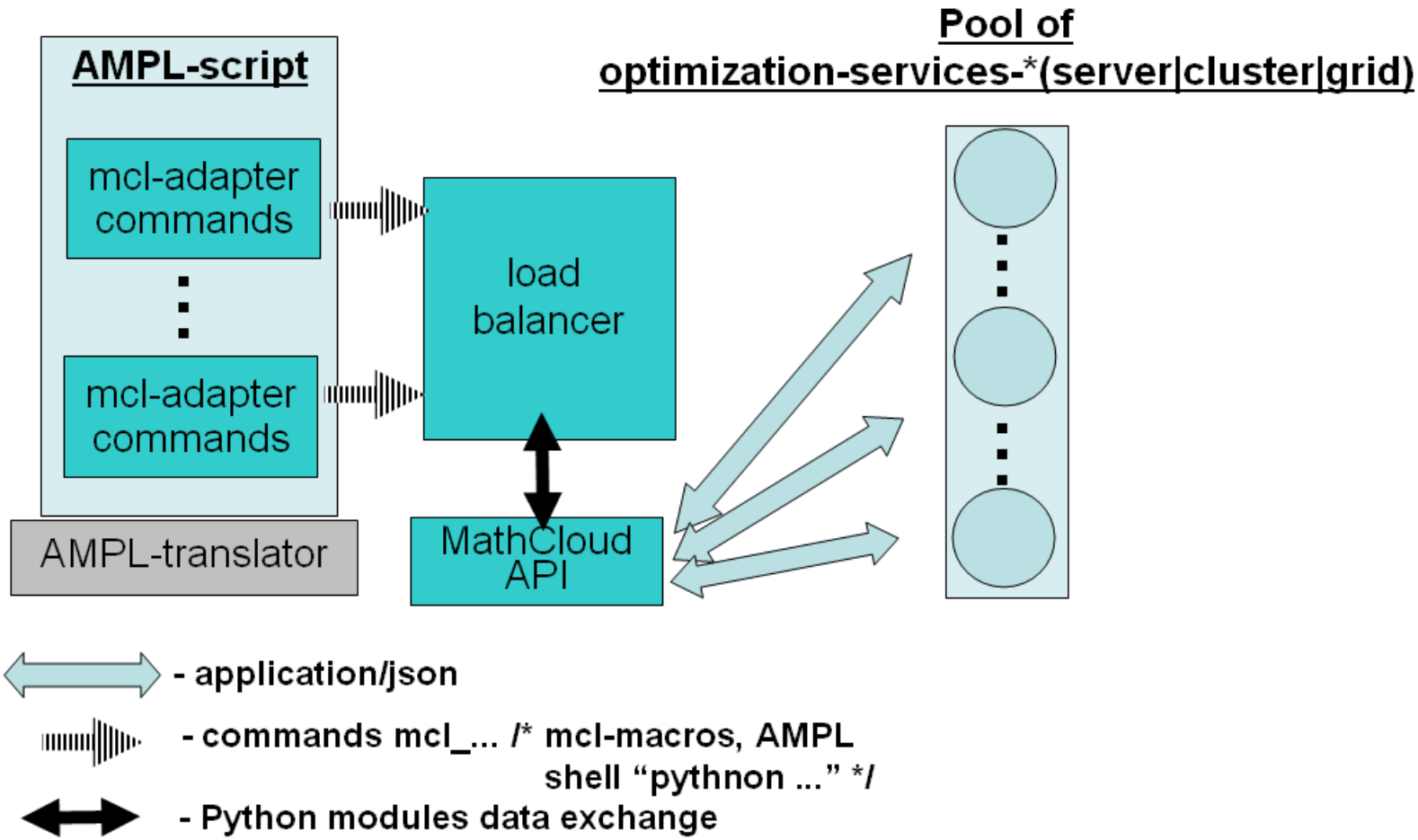
Automatic implementation by WFMS

Mcl-control logic. "Distributed mode" of any AMPL algorithms

Take any any AMPL-algorithm and run it in distributed mode:
- all stubs are sent to a pool off *optimization-service-*;*

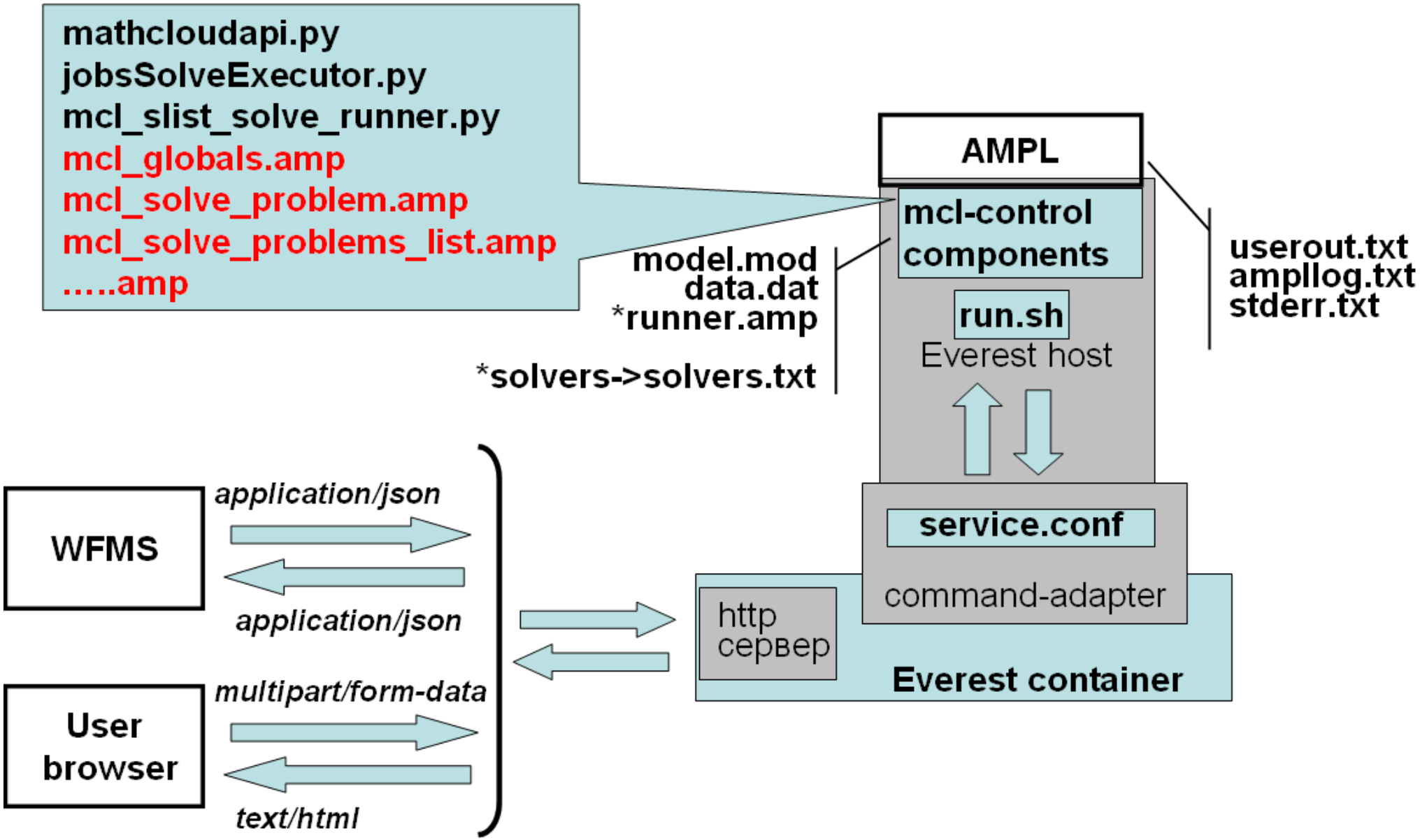
- solutions are brought back to AMPL (and so on);

- includes simple task manager (Python) for load balance in heterogeneous "environment", i.e. unknown task complexity and optimization services performance



Mcl-control architecture. MathCloud API (Python)

Everest atomic + Python + a number of AMPL "macroses"



Transport problem with block structure

A number of products must be delivered from initial placements (offers) to consumers (demands) over transport set of limited carrying capacity.

O **- set of initial placements,** ***D*** **- set of demands,** ***P*** **- set of products**

$Supply_{o,p}$ **- initial values of product** ***p*** **in** ***o***

$Demand_{d,p}$ **- requirement on** ***p*** **in** ***d***

$c_{o,d,p}$ **- cost of transportation (o->d) for unit of product** ***p***

$l_{o,d}$ **- capacity of (o->d)**

$$\sum_{o\in O,d\in D,p\in P} c_{o,d,p}\cdot x_{o,d,p} \to \min \text{ over } \{x_{o,d,p}\}, \text{(total transport cost) s.t.}$$

$$\sum_{o\in O} x_{o,d,p} = Demand_{d,p}\,(d\in D, p\in P) \text{ (products delivery constraints)}$$

$$\sum_{d\in D} x_{o,d,p} = Supply_{o,p}\,(o\in O, p\in P) \text{ (products supply constraints)}$$

$$\sum_{p\in P} x_{o,d,p} = l_{o,d}\,(p\in P) \qquad \text{(transport capacity)}$$

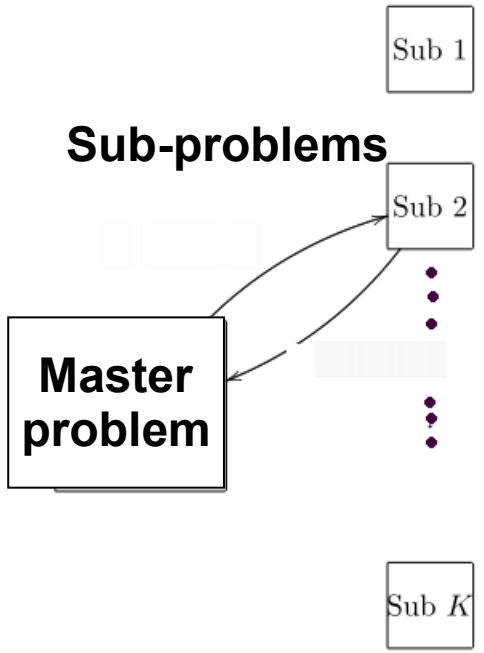
$$x_{o,d,p} \ge 0$$

Classical problem with block structure to demonstrate decomposition algorithms (Dantzig-Wolfe, Bendres etc) . AMPL-algorithm
http://www.ampl.com/NEW/LOOP2/ multi2.mod, multi2.run, multi.dat

Original AMPL DW-algorithm (multi2.run) is not parallel

$$\min c^T x$$
$$Ax = b$$
$$x \geq 0$$

$$Ax = \begin{pmatrix} B_0 & B_1 & B_2 & \dots & B_K \\ & A_1 & & & \\ & & A_2 & & \\ & & & \ddots & \\ & & & & A_K \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_K \end{pmatrix}$$



Original AMPL-algorithm

http://www.ampl.com/NEW/LOOP2/ multi2.run uses subsequent *for* cycle for subproblems solving

```
...
for {p in PROD} { printf "\nPRODUCT %s\n\n", p;
   solve SubII[p];
   ...
   if Reduced_Cost[p] < - 0.00001 then  {
   /* change subproblems parameters */;
   ...
   };
...
```

Modified (multi2_mclTest-remote.amp) is parallel

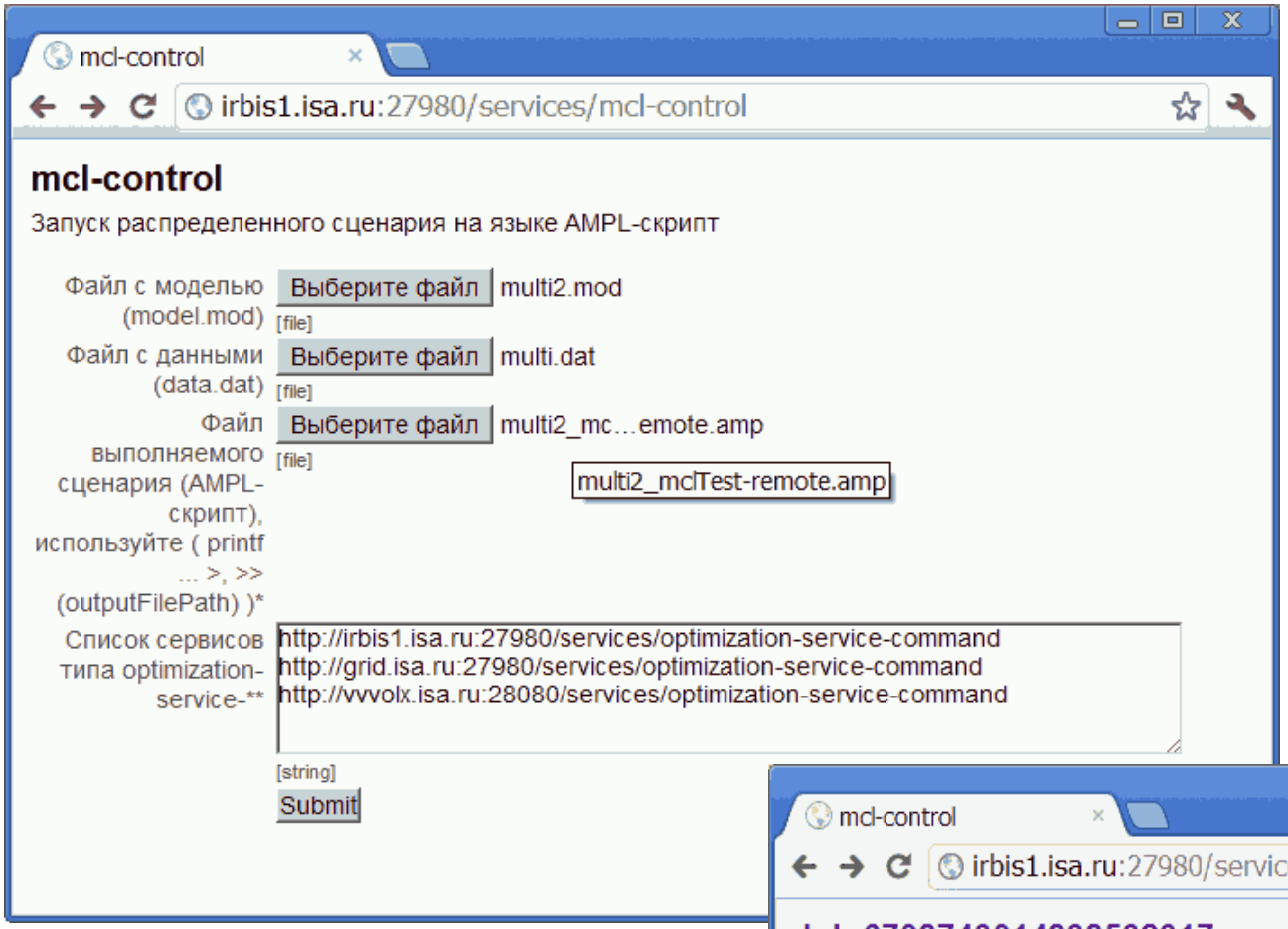
Just replace fragments of original AMPL code. "Map-reduce style"

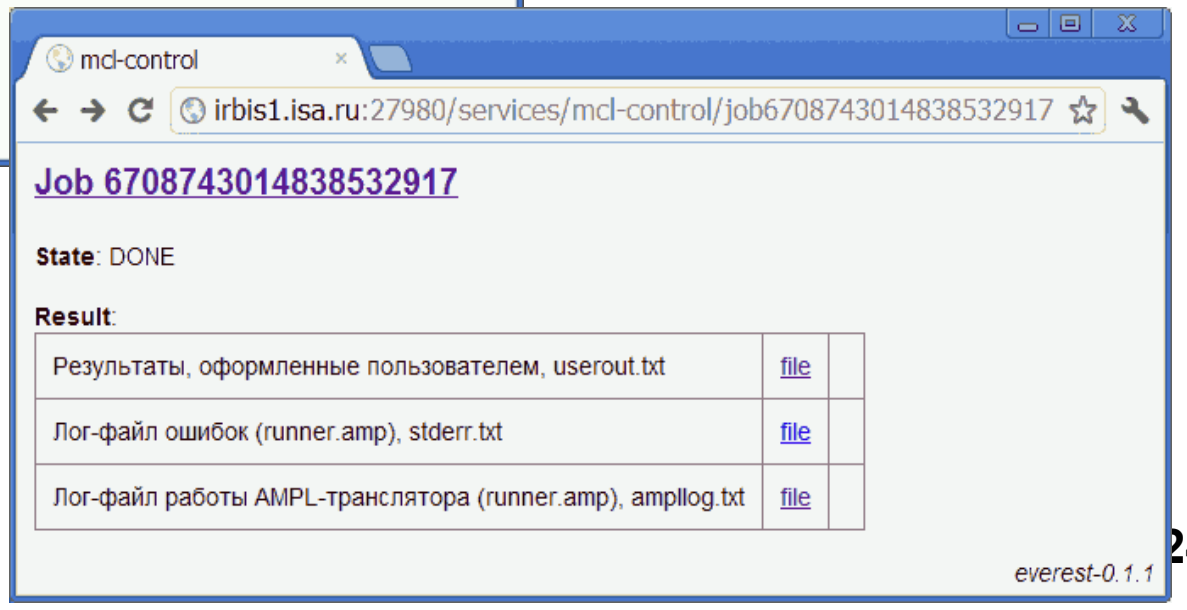
```
for {p in PROD} { printf "\nPRODUCT %s\n\n", p;
   solve SubII[p];
   ...
   if Reduced_Cost[p] < - 0.00001 then  {
   /* change subproblems parameters */;
   ...
   };
```

```
for {p in PROD} { printf "\nPRODUCT %s ==> stub \n\n", p;
   problem SubII[p];
   let __mcl_probName := ("SubII_" & p);
   commands mcl_write_problem_stub.amp; # Generates sub-problems AMPL-stubs
}

commands mcl_solve_problems_list.amp; # Parallel solving of SubII_*

for {p in PROD} { printf "\nPRODUCT %s <== solution\n\n", p;
      # solve SubII[p]
      problem SubII[p];
      solution ("SubII_" & p & ".sol");
if Reduced_Cost[p] < - 0.00001 then  {
   /* change subproblems parameters */;
   ...
   };
```

mcl-control runs multi2_mclTest-remote.amp

Start and finish web-forms





Fragments of userout.txt

irbis1.isa.ru:27980/services/mcl-control/job6708743014838532917/stdout.txt

```
Solving:  ['MasterII']
===== Get solutions =====
Done
http://grid.isa.ru:27980/services/optimization-service-command/job5215895567320271810/stub.sol ====> MasterII.sol
LP_SOLVE 5.5.2.0: optimal, objective 199500
3 simplex iterations

Weight [*,*] (tr)
:     bands       coils       plate      :=
4     0.312616    0           0
5     0           0.43745     0
6     0           0.00383496  0
8     0           0.253042    0
9     0.687384    0.236908    0.562153
10    0           0.0687649   0.437847
;

PHASE II -- ITERATION 11
PRODUCT bands
PRODUCT coils
PRODUCT plate

Solving:  ['SubII_bands', 'SubII_coils', 'SubII_plate']
===== Get solutions =====
Done
http://irbis1.isa.ru:27980/services/optimization-service-command/job4357482254026733380/stub.sol ====> SubII_bands.sol
http://vvvolx.isa.ru:28080/services/optimization-service-command/job5136434874483769410/stub.sol ====> SubII_coils.sol
http://vvvolx.isa.ru:28080/services/optimization-service-command/job7569490417622487565/stub.sol ====> SubII_plate.sol
```

“Fast” optimization-services may solve more problems than “slow” ones

Branch-and-bound for MI... problem (e.g. boolean)

General scheme of search tree traversal for problem $P(X_B,X_C)$

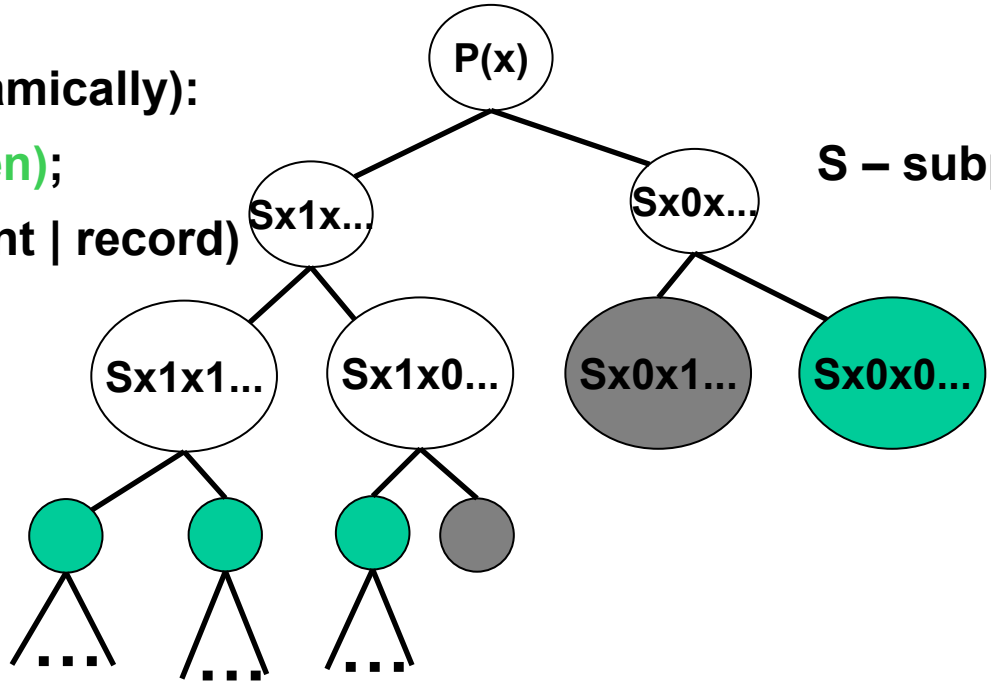
$$f_0(x_B, x_C) \to \min_{x_B, x_C} \quad (x_B, x_C) \in Q$$

Current state of B&B (changed dynamically):

- list of nodes to be processed (green);

- known upper-bound (aka incumbent | record)

$$\mathrm{UB} = f_0(x'_B, x'_C), \ (x'_B, x'_C) \in Q$$



S – subproblems

Node operation:

1) calculate lower-bound of S, LB(S), by relaxation of boolean constraints to, e.g. LP;

2) if, accidentally, feasible set of variables found $(x''_B, x''_C) \in Q$ – update UB

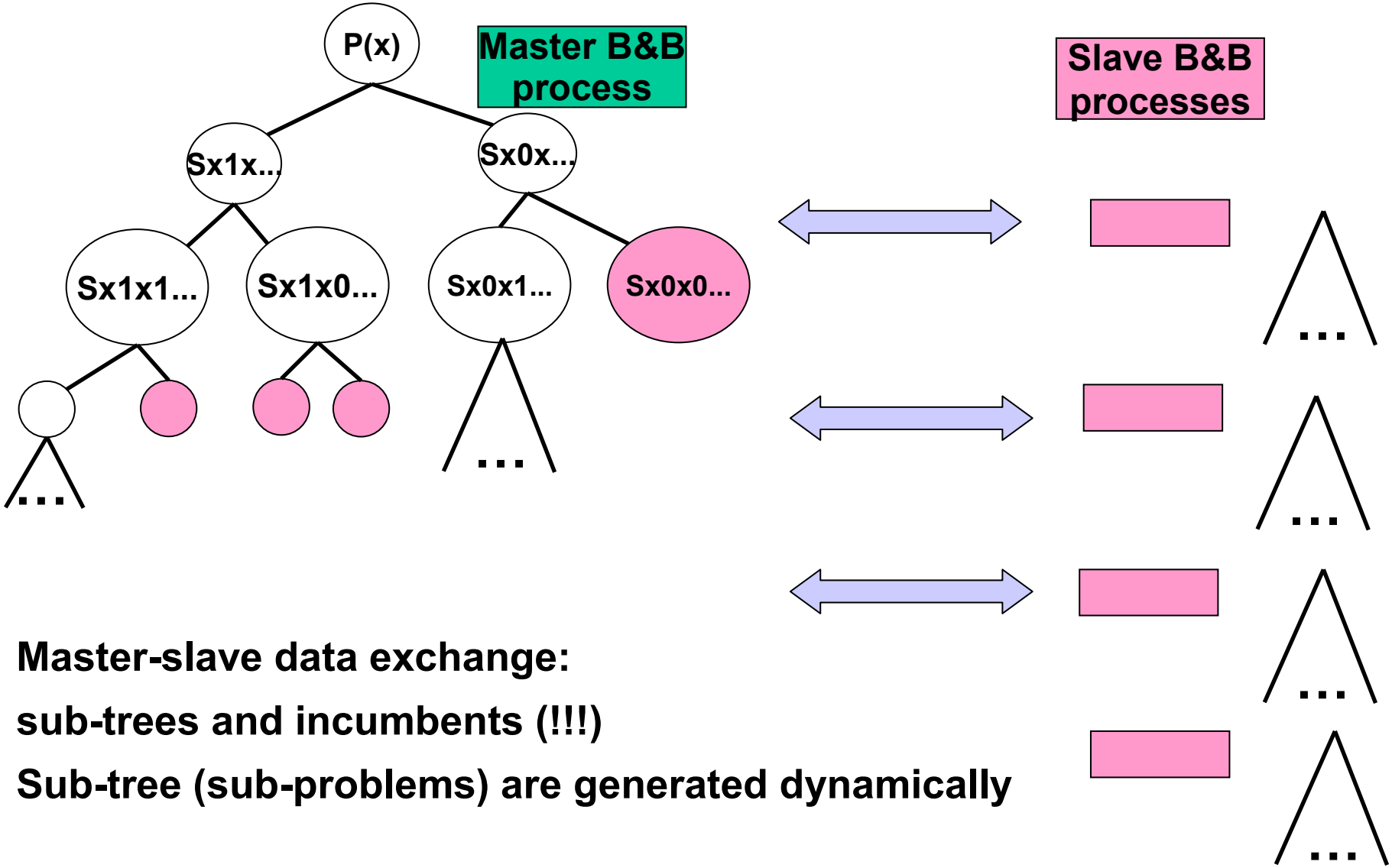
$$\mathrm{UB} := \min\{\mathrm{UB}, f_0(x''_B, x''_C)\}$$

3) if LB(S) >= UB – discard node from the list (grey);

4) select boolean variable to split node and add new ones to the tree

B&B is one of the best algorithms suited for parallelization

Fine-grained decomposition of B&B (traditional approach)



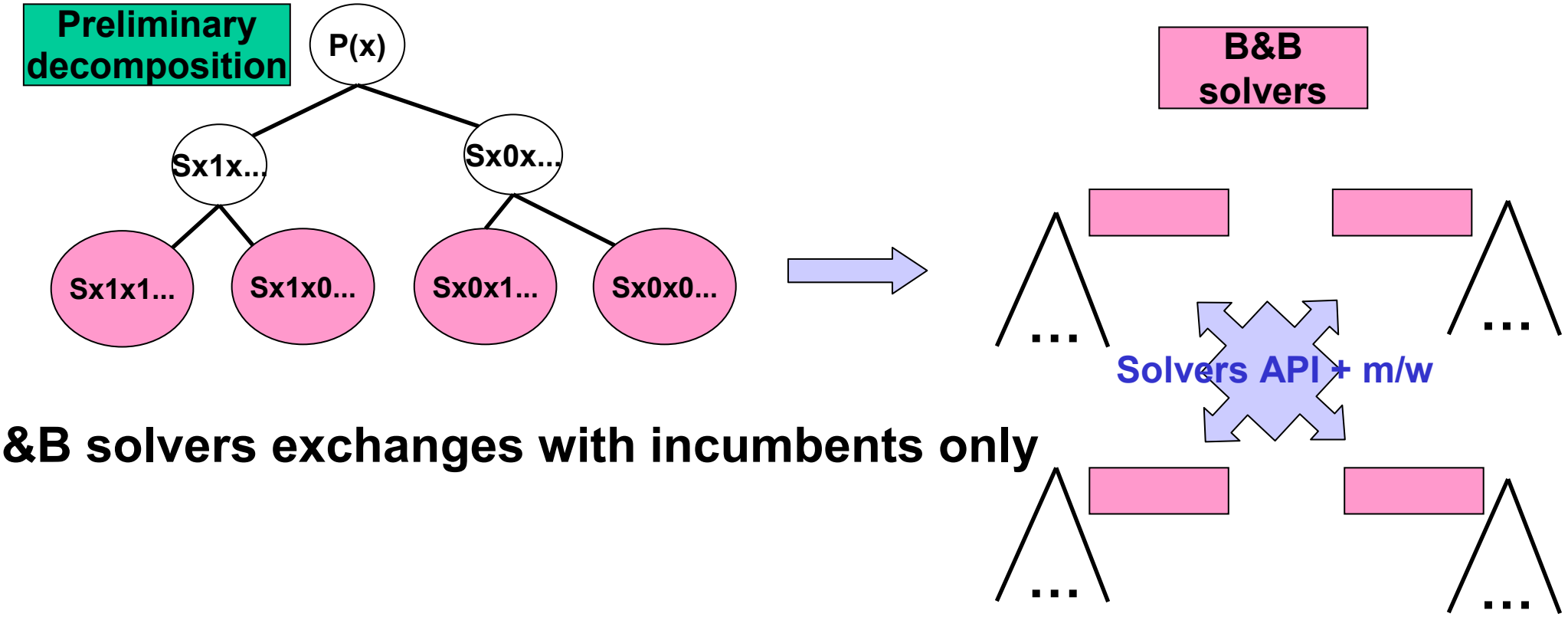
Master-slave data exchange:

sub-trees and incumbents (!!!)

Sub-tree (sub-problems) are generated dynamically

Usually, the approach is based on MPI and run at high-performance cluster

Coarse-grained (“static”) decomposition of B&B



B&B solvers exchanges with incumbents only

The approach is not so popular as fine-grained one, but is much more easy for implementation via solvers’ API and some “light-weight” middleware, e.g. Erlang, Zeroc Ice, ZeroMQ etc.

Preliminary decomposition is crucial for speed-up and requires analysis of the problem’s data ! E.g. by AMPL (!)

Travelling salesmen problem coarse-grained experiment (1)

$$\sum_{i>j} d_{ij} x_{ij} \to \min_{x_{ij}, f_{ij}} \text{ wrt:}$$

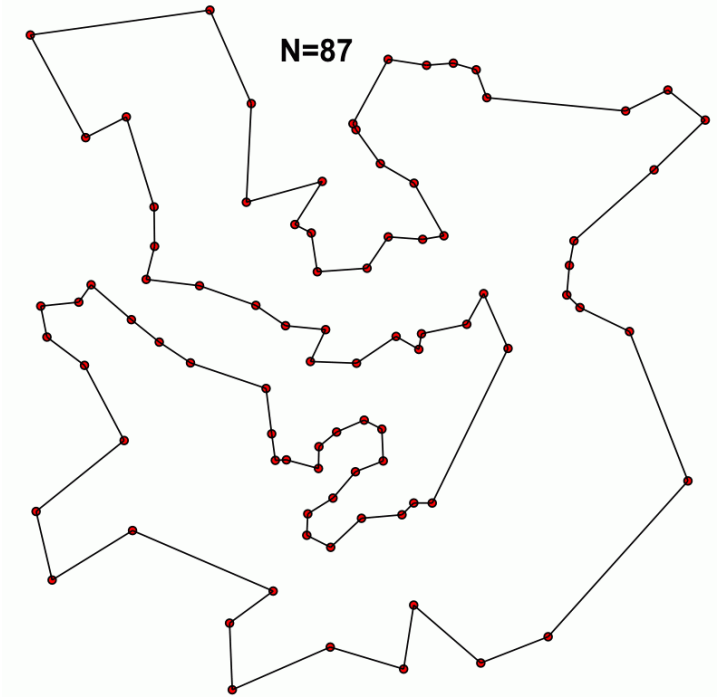
$$\sum_{j\in V, i>j} x_{ij} + \sum_{j\in V, i<j} x_{ij} = 2 \quad (i \in V = \{1:n\});$$

$$f_{ij} \le \left(\begin{cases} n, \text{ if } i = 1 \\ n-1, \text{ if } i > 1 \end{cases} \right) * \left(\begin{cases} x_{ij}, \text{ if } i < j \\ x_{ji}, \text{ if } i > j \end{cases} \right) \quad ((i,j) \in V \times V);$$

$$\sum_{j:(i,j)\in V\times V} f_{ij} - \sum_{j:(i,j)\in V\times V} f_{ji} \le \begin{cases} n-1, \text{ if } i = 1 \\ -1, \text{ if } i > 1 \end{cases} \quad (i \in V);$$

$$\sum_{j:(i,j)\in V\times V} f_{ij} \ge 1 \quad (i \in V);$$

$$x_{ij} = \{0,1\}.$$



"Random" selection of x_{ij} to decompose doesn't give speed-up

Heuristic rule: sort $\{d_{ij}\}$ in ascending order and decompose by x_{ij}:=0|1 corresponding to the smallest d_{ij} (to get "balanced" by incumbents subproblems ??)

Subproblems has been generated as AMPL-stubs by special AMPL "preprocessing" script

Travelling salesmen problem coarse-grained experiment (2)

dCBC prototype (CBC, CBC API + Erlang)

N	T(CBC), min	T(SCIP), min
80	5.3	1.6
90	20.3	6
100	623	10
110	>10000	75

N	n of Xij fixed	n of subprobs	T(CBCx1), min	T(dCBC), min
80	4	16	5.3	2
90	5	32	20.3	11
100	6	64	623	229
110	7	128	>10000	1212

Computing resources (12 CBC instances) :

8 CBC instances at 2 x Intel Xeon E5620 @ 2.40GHz

4 CBC instances at Intel Core i7-2600K @ 3.40GHz

Task-worker scheduling problem coarse-grained experiment (1)

$$z \to \min_{t_k, x_{kn}, z} \text{ wrt:}$$

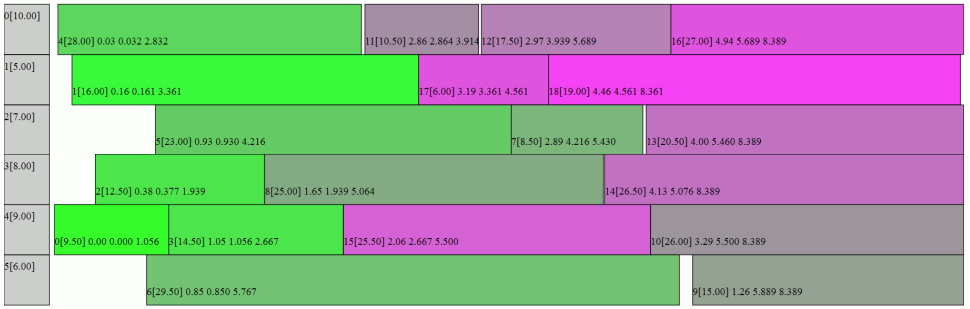
$$t_k \ge T_k \quad (k = 1:K);$$

$$\sum_{n=1:N} x_{kn} = 1 \quad (k = 1:K) \quad \left(\text{or} \sum_{n=1:N} x_{kn} \ge 1 \quad (k = 1:K) \right);$$

$$t_k + \frac{\tau_k}{p_n} x_{kn} \le t_{k'} + C \cdot \left(2 - x_{kn} - x_{k'n}\right) \quad (k = 1:K, k < k' \le K, n = 1:N);$$

$$t_k + \frac{\tau_k}{p_n} x_{kn} \le z \quad (k = 1:K, n = 1:N);$$

$$z, t_k \in \mathrm{R}^1, \; x_{kn} = \{0 \mid 1\}, \; k = 1:K, n = 1:N$$



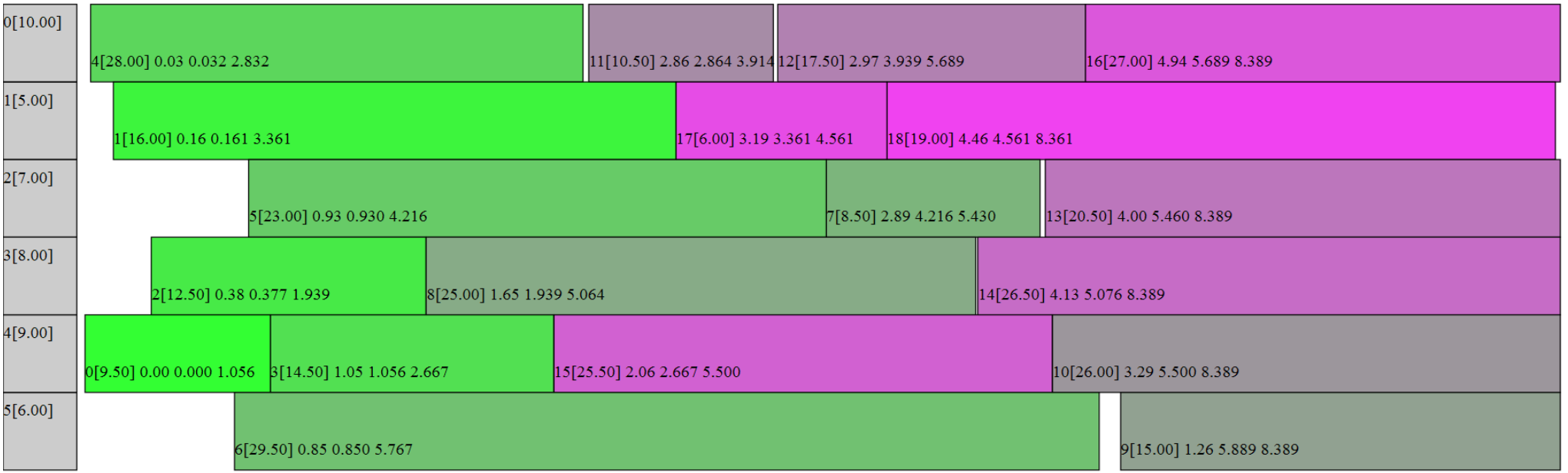
"Random" selection of x_{ij} to decompose doesn't give speed-up

Heuristic rule: sort $\{\tau_k / p_n\}$ in ascending order and decompose by x_{kn}:=0|1 corresponding to the smallest τ_k / p_n (to get "balanced" by incumbents subproblems ??)

Subproblems has been generated as AMPL-stubs by special AMPL "preprocessing" script

Task-worker scheduling problem coarse-grained experiment (2)

dCBC prototype (SCIP, SCIP API + Erlang)



6 workers, 19 tasks, exact solution

Computing resources (40 SCIP) :

6 boolean x_{kn} has been fixed

(64 subproblems) took 720 sec.

host	n of SCIP instances	T(SCIPx1), sec
server-1	8 X Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz	929.7
server-2	16 X Intel(R) Xeon(R) CPU E5620 @ 2.40GHz	1368.59
xen-vm-2	8 X Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz	2172.27
xen-vm-1	8 X Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz	2270.38

Very different performance,

no load balance

Instead of conclusion

To apply our experience and available software we are looking for problems requiring optimization modeling.

And we are open for collaboration, http://dcs.isa.ru.

Thank you for your attention.

Questions?

MathCloud JSON-description of mcl-control (inputs)

```
{ "name": "mcl-control",
"description": "Distributed running of AMPL-скрипт ",
"inputs": {
   "model.mod": {
      "type": "file",
      "title": "AMPL model (*.mod)",
      "optional": true
   },
   "data.dat": {
      "type": "file",
      "title": "Data-file (data.dat)",
      "optional": true
   },
   "runner.amp": {
      "type": "file",
      "title": "AMPL algorithm, use( printf ...>,>>(outputFilePath))"
   },
   "solvers": {
      "type": "string",
      "title": "List of optimization-service-*"
   }
},
```

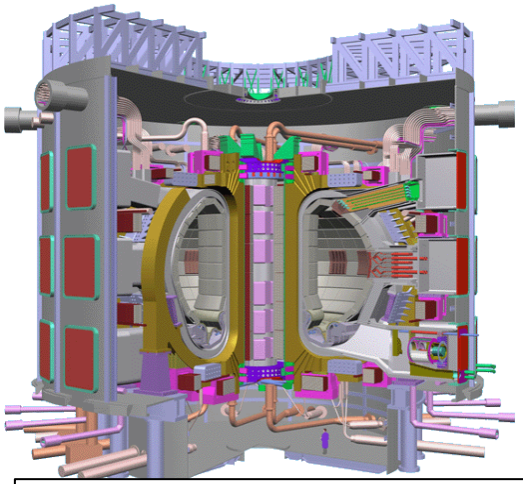
MathCloud JSON-description of mcl-control (outputs and impl)

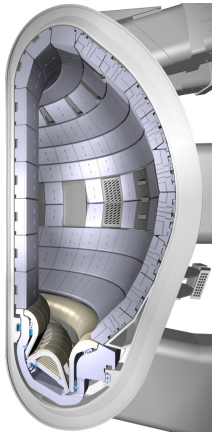
```
"outputs": {"userout.txt": {
      "type": "file",
      "title": "Results printfed by user, userout.txt" },
   "stderr.txt": {
      "type": "file",
      "title": "System error during running runner.ampl, stderr.txt"},
   "ampllog.txt": {
      "type": "file",
      "title": "Log-file of AMPL-translator, ampllog.txt"} },
"implementation": {"adapter": "command",
   "command": "bash ../../services/mcl-control/run.sh
...
```

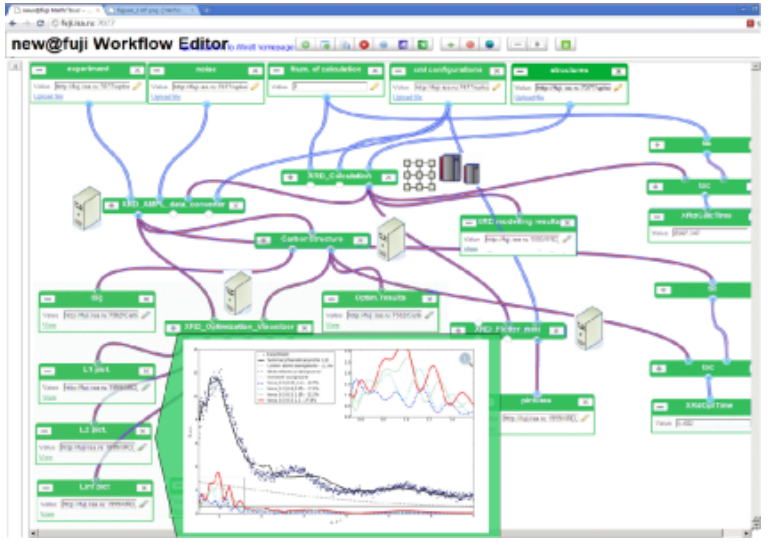
```
# run.sh, bash script
...
ln -s stdout stdout.txt
ln -s stderr stderr.txt
{echo "BASE_URL=\"$1/services/$SERVICE_NAME/job$(basename $PWD)\""
echo 'SOLVER_URLS=['
cat solvers.txt | tr '\r' '\n' | sed -e '/^[[:space:]]*$/d' \
   -e 's/[[:space:]]//g' -e 's/^\(.*\)$/"\1",/'
echo ']'} > mcl_config.py

ln $SERVICE_DIR/*.amp $SERVICE_DIR/*.py .

ampl runner.amp
```

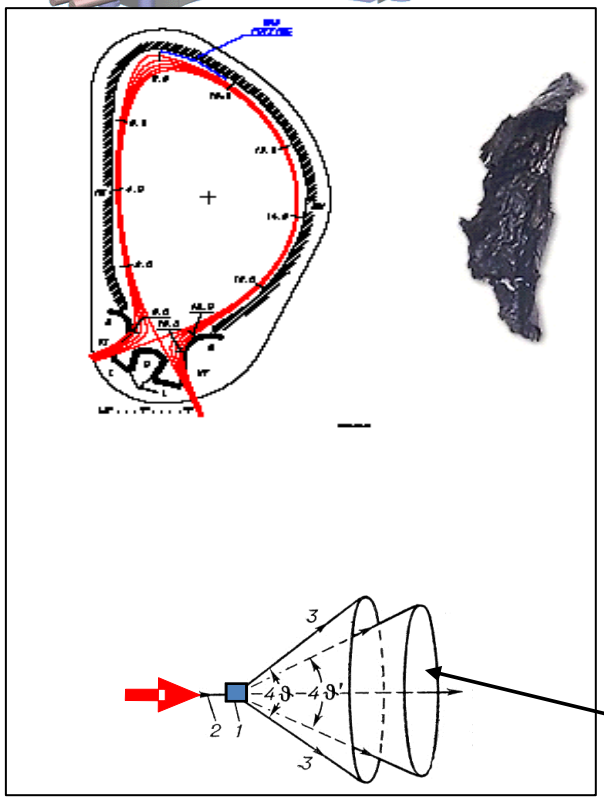
Optimization in processing of experimental data

Fine structure of carbon films deposited in thermonuclear reactor TOKAMAK T-10 by results of synchrotron X-ray scattering diffraction

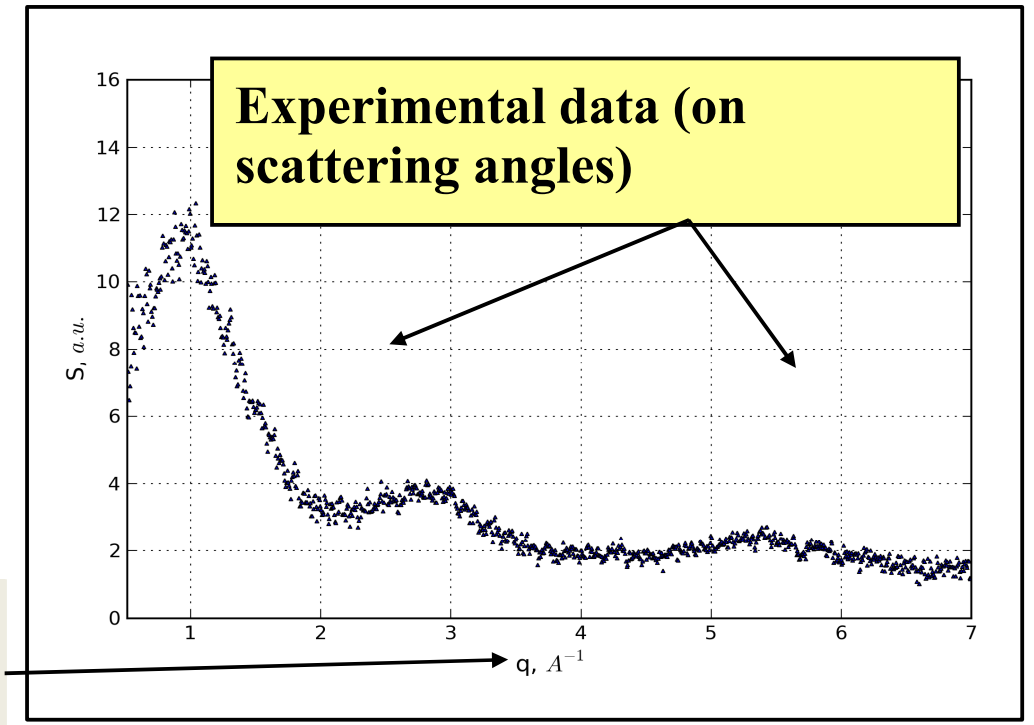








$$q = \frac{4\pi}{\lambda}\sin\left(\frac{\theta_x}{2}\right)$$



Unexpected interesting result

Dominance of toroidal spatial forms of carbon has been revealed (7 toroidal form over ~500 candidates for all criteria !)

A. B. Kukushkin, V. S. Neverov, N. L. Marusov, I. B. Semenov, B. N. Kolbasov, V. V. Voloshinov, A. P. Afanasiev, A. S. Tarasov, V. G. Stankevich, Svechnikov "Few-nanometer-wide carbon toroids in the hydrocarbon films deposited in tokamak T-10" // **Chemical Physics Letters** (14 March 2011) doi:10.1016/j.cplett.2011.03.036