



OpenTURNS and HPC within SALOME platform

O. Mircescu

HPC and Uncertainty Treatment
Examples with OpenTURNS
EDF – PhiMeca – IMACS -Airbus Group – CEA

Prace Advanced Training Center
May 18th 2018



MAISON DE LA SIMULATION

OUTLINE

1. Presentation of Salome
2. OpenTURNS graphical user interface
3. Step by step example
4. OpenTURNS graphical user interface distribution and future evolutions
5. Examples of OpenTURNS studies with HPC

Presentation of Salome (1/8)

- **What's Salome ?**

- Modular simulation platform
- An open framework to build domain specific solutions

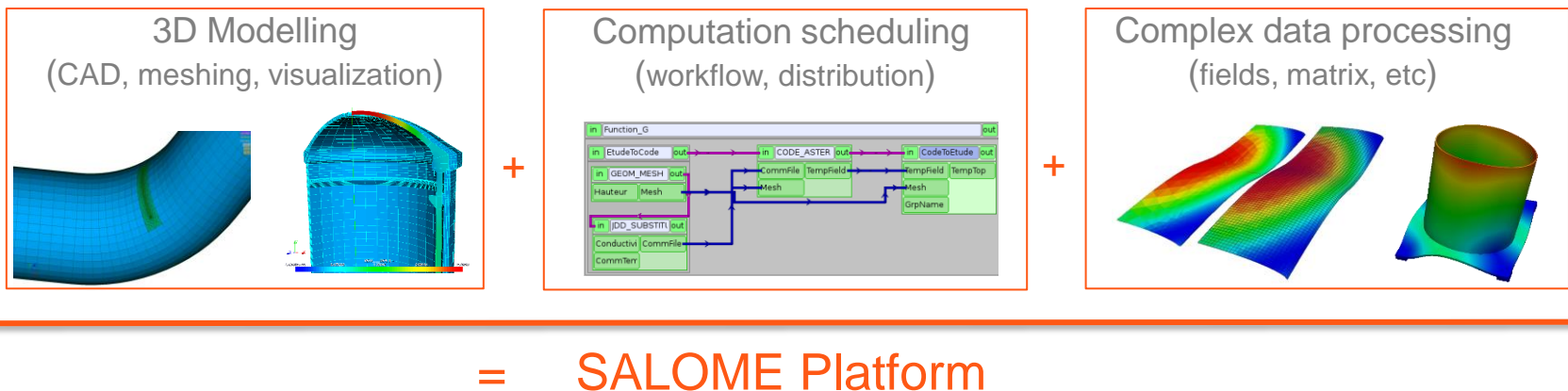
- **Who is developing Salome ?**

- EDF, CEA and OpenCascade partnership



Presentation of Salome (2/8)

- A middleware providing generic tools for numerical simulations
 - Geometry modelling, meshing, field handling and visualization
 - Data Exchange Model for interoperability between solvers and tools
 - Computation scheduling (YACS)

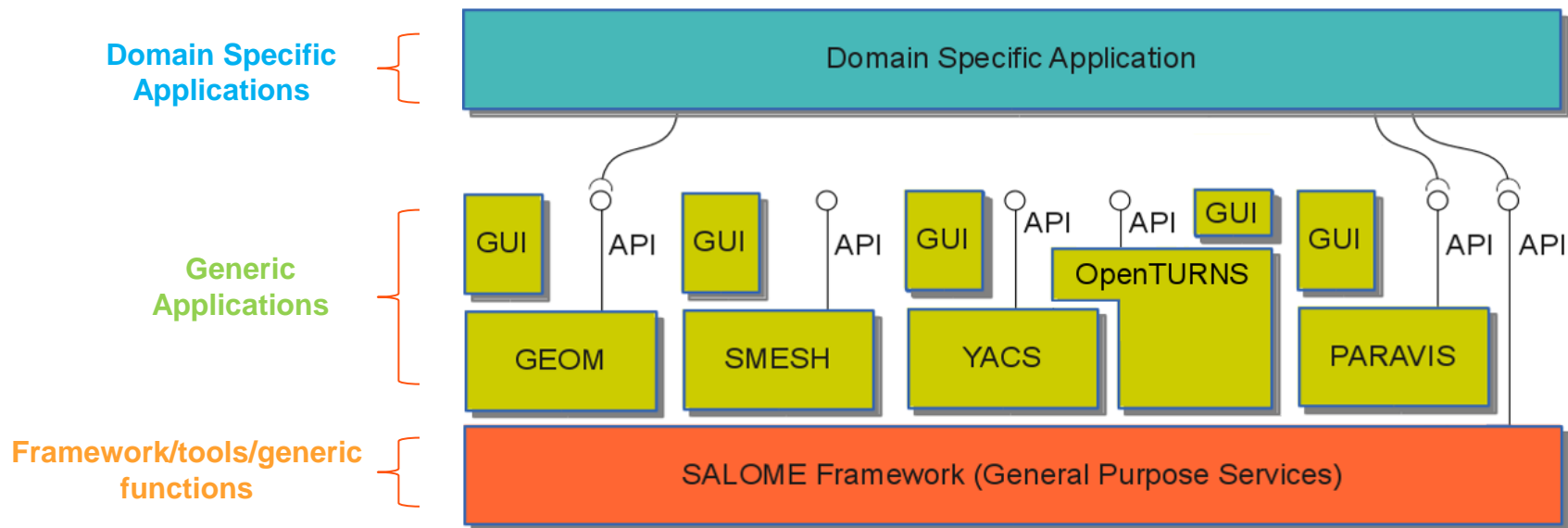


Presentation of Salome (3/8)

■ An open framework to build domain specific solutions

□ Each module provides :

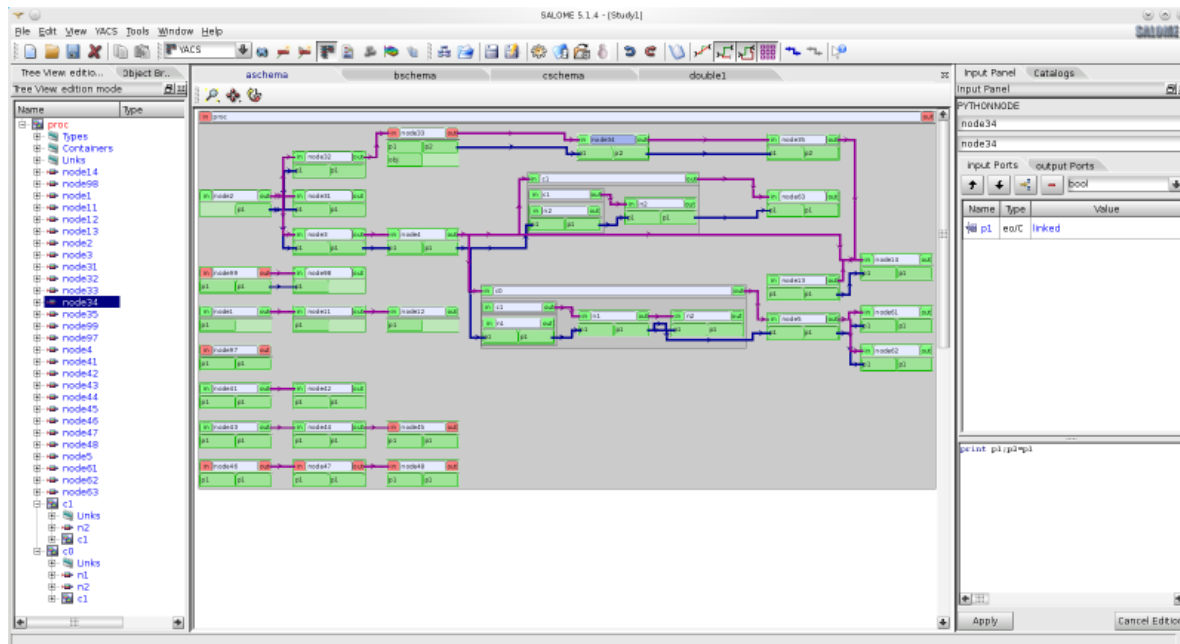
- A textual interface for scripting based on Python
- A graphical interface for interactive usages (C++, Qt, PyQt)
- A programming interface to build custom applications (API C++ and Python)



Presentation of Salome (4/8)

■ Presentation of YACS (1/6)

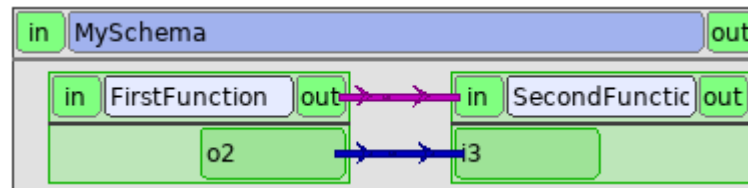
- Distribution of computations on multiple resources
- Parallelism and parametric computation
- Chaining computation nodes
- GUI and APIs for Python and C++



Presentation of Salome (5/8)

■ Presentation of YACS (2/6)

- Computation nodes have input and output ports
- Ports are typed variables (integer, double, string, collections, pyobj or user defined types)
- Computation nodes have placement constraints
- Python node: run a python script
- Salome node: run a function of a Salome component
 - YACSGEN – tool for developing your own SALOME component (C++, python, FORTRAN)
- Parallel “ForEachLoop”



Presentation of Salome (6/8)

■ Presentation of JOBMANAGER (2/6)

- Create, launch and monitor jobs on computer clusters
- Single interface for several batch managers (Slurm, PBS, COORM, OAR, SGE, LSF)
- GUI and APIs for Python and C++
- Jobs can run scripts or YACS schemas

The screenshot displays the JobManager interface, which is divided into several sections. At the top, there are tabs for 'Main', 'Management', and 'Refresh'. Below these are buttons for 'Create', 'Start', 'Get Results', 'Stop', 'Delete', 'Restart', 'Edit/Clone', and 'Refresh Jobs'. A table lists the jobs, with columns for 'Job Name', 'Type', 'State', 'Resource', and 'Launcher Id'. The table shows three jobs: 'MyJob', 'MyJob_0', and 'MyJob_1', all of which are 'Finished' and run on the 'porthos' resource. Below the table, there are tabs for 'Job Summary' and 'Job Files'. The 'Job Summary' tab is active, showing details for 'MyJob_0'. The details include 'Name: MyJob_0', 'Type: Command', 'State: Finished', 'Launcher Id: 1', 'Resource: porthos', 'Job File: /home/l35256/salome/training/relaunch_job/cas_sh/recommand.py', 'Env File:', and 'Preprocessing File:'. Below this, there are 'Run values' including 'Number of Input Files: 1', 'Number of Output Files: 1', 'Execution directory: /scratch/l35256/workingdir/myrun/2f3d244134d9c68ff1b58449a58a1e7', 'Result directory: /home/l35256/salome/tmp/work/2f3d244134d9c68ff1b58449a58a1e7/result', 'Launcher file:', 'Maximum duration: 00:01', 'Required memory: Default', 'Number of processors: 1', 'Number of nodes: 0', 'Exclusive: no', and 'Launcher args:'. On the right side of the interface, there is a 'Job Manager' panel with buttons for 'Load Jobs', 'Save Jobs', and 'Auto Refresh: 30s'. Below these buttons is a 'Messages' section showing a list of messages: 'New job added MyJob', 'New job added MyJob_0', 'New job added MyJob_1', 'New job added MyJob_2', 'New job added MyJob_3', and 'New job added MyJob_4'. At the bottom right, there is a 'Summary' panel with a 'Jobs Summary' section showing 'Number of jobs: 6', 'Number of created jobs: 0', 'Number of queued jobs: 0', and 'Number of finished jobs: 0'. Below this are buttons for 'Summary' and 'Resource Catalog'.

Job Name	Type	State	Resource	Launcher Id
1 MyJob	Command	Finished	porthos	0
2 MyJob_0	Command	Finished	porthos	1
3 MyJob_1	Command	Finished	porthos	2

Job Summary

Main values

Name: MyJob_0
Type: Command
State: Finished
Launcher Id: 1
Resource: porthos
Job File: /home/l35256/salome/training/relaunch_job/cas_sh/recommand.py
Env File:
Preprocessing File:

Run values

Number of Input Files: 1
Number of Output Files: 1
Execution directory: /scratch/l35256/workingdir/myrun/2f3d244134d9c68ff1b58449a58a1e7
Result directory: /home/l35256/salome/tmp/work/2f3d244134d9c68ff1b58449a58a1e7/result
Launcher file:
Maximum duration: 00:01
Required memory: Default
Number of processors: 1
Number of nodes: 0
Exclusive: no
Launcher args:

Job Manager

Load Jobs Save Jobs Auto Refresh: 30s

Messages

New job added MyJob
New job added MyJob_0
New job added MyJob_1
New job added MyJob_2
New job added MyJob_3
New job added MyJob_4

Summary

Jobs Summary:

Number of jobs: 6
Number of created jobs: 0
Number of queued jobs: 0
Number of finished jobs: 0

Summary Resource Catalog

Presentation of Salome (7/8)

■ Salome resources

- Salome has to be installed on every machine you want to use
- You have to declare each installation in your local catalog of resources

Edit/Add a resource

Main values

Name:

Hostname:

Protocol:

Username:

Applopath:

Component List:

Working Directory:

☒ This resource can be used to launch batch jobs

☐ This resource can be used to run interactive containers

Configuration values

OS:

Memory (mb):

CPU Clock:

Nb node:

Nb proc/node:

Batch Manager:

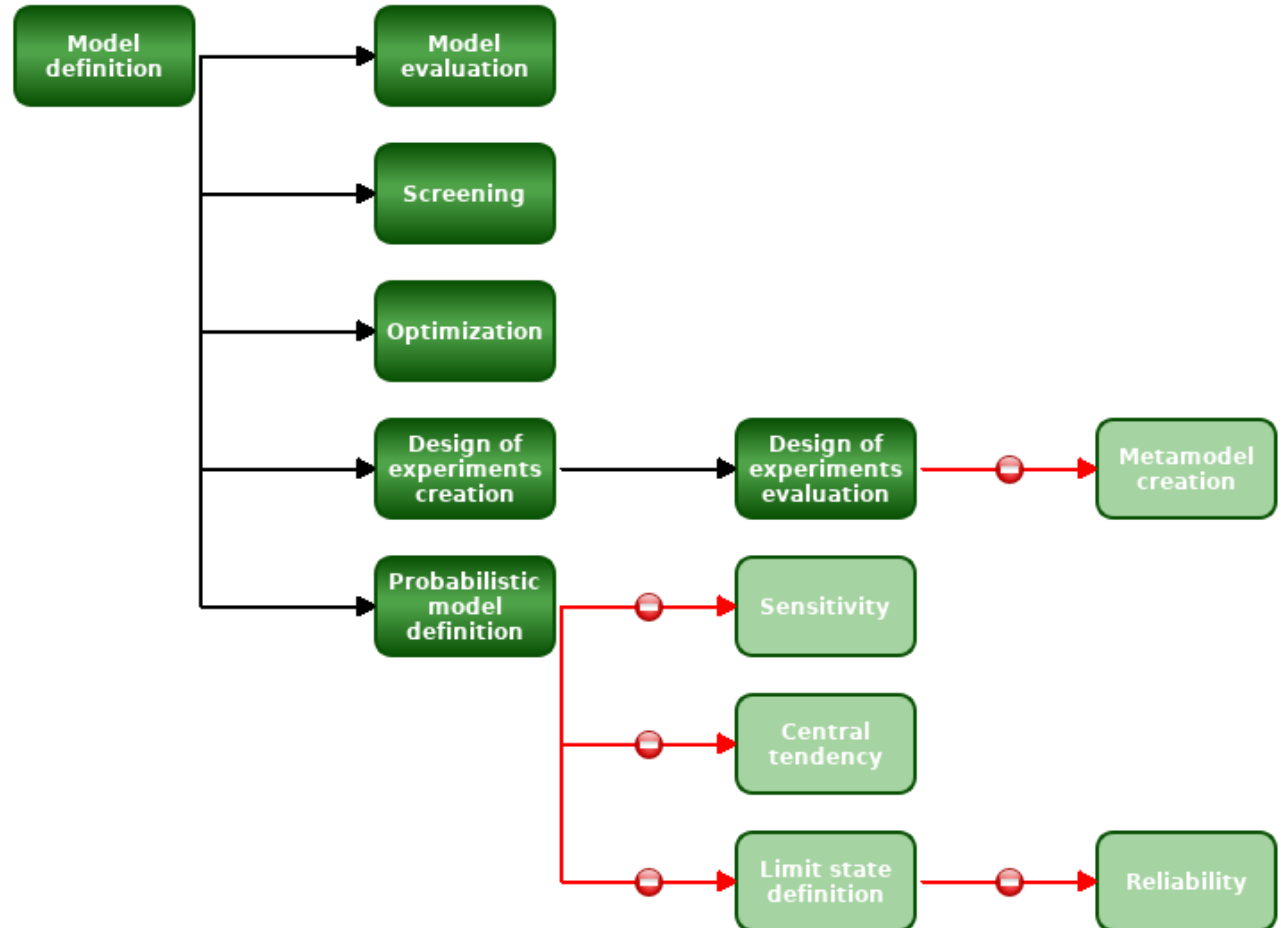
MPI impl:

Internal protocol:

OpenTURNS Graphical User Interface (1/5)

- Goal of OpenTURNS GUI

- Guide user in a homogeneous environment through the roadmap defined by the global methodology of treatment of uncertainties (physical model evaluation and display of result to take decisions...)



OpenTURNS Graphical User Interface (2/5)

- **OpenTURNS GUI is more than a simple GUI !**
 - OpenTURNS GUI is based on a high level object model (Main study, Deterministic study, Probabilistic study, ...)
 - OpenTURNS GUI includes a layer over OpenTURNS tools.
 - OpenTURNS GUI has been designed to mix beginners and advanced users
- **How can you use it ?**
 - Standalone binary called otgui
 - In a dedicated salome module
 - Can be used in customized salome module

OpenTURNS Graphical User Interface (3/5)

- Two complementary ways to pilot OpenTURNS GUI : Python and widgets
- The design of OpenTURNS GUI allows a strong relationship between Python scripting and graphical interface (Model/View paradigm).
 - Actions you perform on gui can be mapped into a Python representation.
 - Load python script and dump python script.
 - Start session with graphical interface then continue with script then...
- OpenTURNS GUI offers software bricks usable outside a dedicated tool

OpenTURNS Graphical User Interface (4/5)

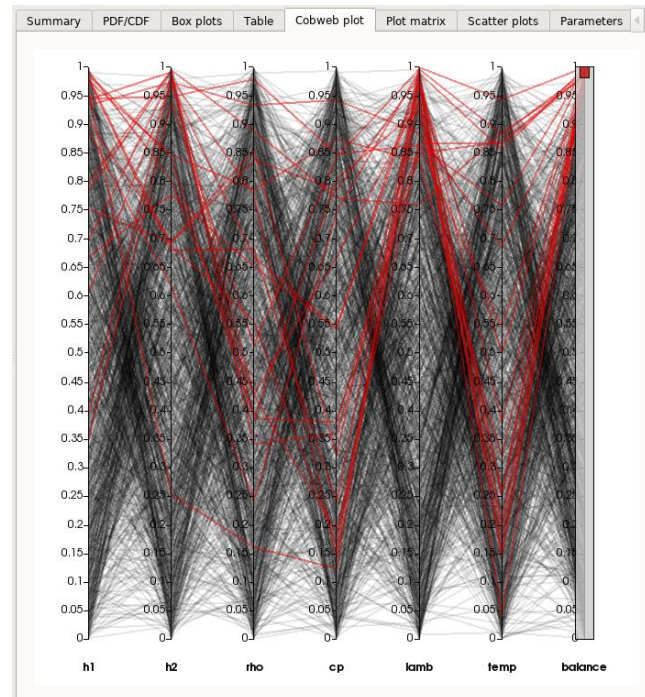
- **OpenTURNS GUI is an excellent target for High Performance Computing (HPC)**
 - A large number of independent computations
- **The usage of distributed resources is very dependent on the context:**
 - Use of a cluster (homogeneous, centralized) / a grid (heterogeneous, decentralized)?
 - Communication protocol with the cluster?
 - Which batch / grid manager?
 - Can we install software on the cluster?
 - Global / local (by node) file system?
 - Execution of OpenTURNS script on the client workstation / on the cluster?
 - Which middleware for the distribution on the cluster?
 - Size of input and output files of the solver code?



OpenTURNS GUI uses Salome

OpenTURNS Graphical User Interface (5/5)

- SALOME software bricks used OpenTURNS GUI
 - YACS & JOBMANAGER
 - GUI Python console widget
 - PARAVIS widgets



Step by step example (1/6)

■ Goal

- Compute an OpenTURNS study of a python function using a computer cluster

■ Prerequisite

- Create the python function of a unitary case
- Input parameters must be only float variables
- Return statement must contain only float variables
- Run and test the function outside Salome

Heat exchange computation with SYRTHES

h1 T1
Face S1



h2 T2
Face S2

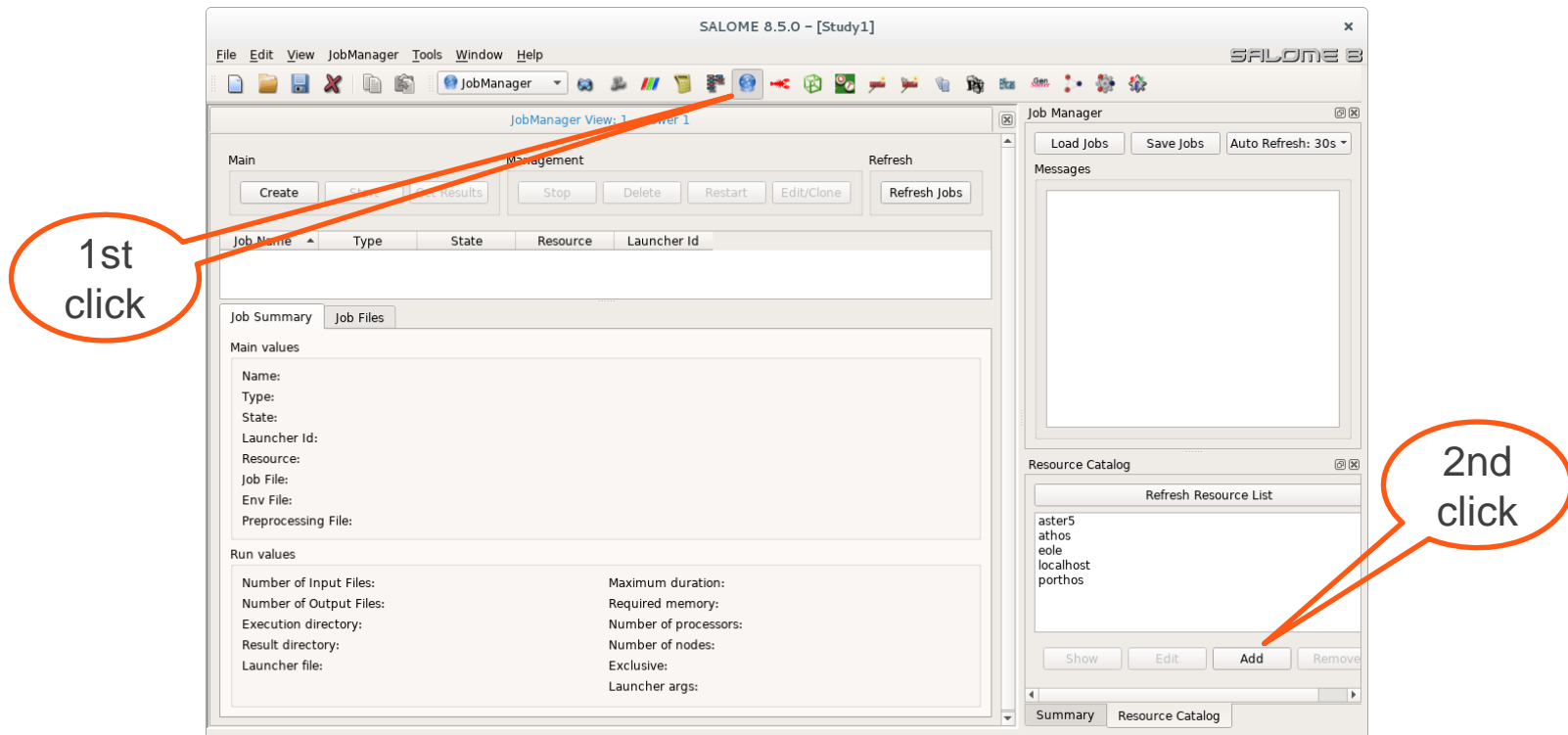
Heat exchange coefficient: h1, h2
Conductivity: lamb
Density: rho
Heat capacity: cp

```
def _exec(h1, h2, rho, cp, lamb):  
  
    etude=Etude()  
    etude.init(h1, h2, rho, cp, lamb)  
  
    #Run the calculation  
    etude.launch_calc()  
  
    #Temperature at the probe  
    temp = etude.extract_probe(2,2)  
  
    #Balance at 2  
    balance = etude.extract_balance(2, 2)  
  
    return temp, balance
```


Step by step example (2/6)

■ Installation steps

- Install Salome on your personal computer
- Install Salome on the remote cluster
- Make sure you can connect to cluster without typing the password (ssh-copy-id)
- Declare the cluster installation in your local resource catalog (JOBMANAGER module)



Step by step example (3/6)

■ OpenTURNS study steps

- Open the OpenTURNS module in a Salome session
- Create a new study with a YACS model
- Set the python script in the YACS model

The screenshot displays the OpenTURNS interface within a Salome session. The 'Studies' panel on the left shows a tree structure with 'OTStudy_0' and 'Physical models'. The 'Physical models' section is expanded, showing 'PhysicalModel_0' and its 'Definition' tab. The 'Definition' tab contains a table of inputs and outputs. The 'Python to YACS schema editor' window is open, showing a Python script. The script includes a function definition 'def exec(h1, h2, rho, cp, lamb):' and a return statement 'return temp, balance'. Red circles highlight these two lines. Red arrows point from the 'Physical models' section in the 'Studies' panel to the 'Python to YACS schema editor' window, and from the 'YACS model' button to the 'Python to YACS schema editor' window.

Inputs

Name	Description	Value
1 h1		0
2 h2		0
3 rho		0
4 cp		0
5 lamb		0

Outputs

✓ Name	Description	Value
1 ✓ temp		?
2 ✓ balance		?

Python to YACS schema editor

```
113 #Test with a C file
114 self.set_data(e_cfile, '$h2', h2)
115
116 def clean(self):
117     shutil.rmtree(self.etude_dir, ignore_errors=True)
118
119 def exec(h1, h2, rho, cp, lamb):
120
121     etude=Etude()
122     etude.init(h1, h2, rho, cp, lamb)
123
124     #Run the calculation
125     etude.launch_calc()
126
127     #Temperature at the probe
128     temp = etude.extract_probe(2,2)
129
130     #Balance at 2
131     balance = etude.extract_balance(2, 2)
132
133     return temp, balance
134
```

py -> YACS Cancel

Step by step example (4/6)

■ OpenTURNS study steps

- Choose the laws of the input parameters

The screenshot displays the OpenTURNS software interface. On the left, the 'Studies' panel shows a tree structure with 'OTStudy_0' expanded, containing 'Physical models' and 'PhysicalModel_0'. Below this, the 'Graph setting' panel is visible, showing the title 'PDF' and the 'PDF' radio button selected. The 'X-axis' is labeled 'h1' with a title 'h1', and the 'Y-axis' is labeled 'Density'. The 'Min' value is 250 and the 'Max' value is 1750. An 'Export' button is at the bottom right of this panel.

The main area is divided into two tabs: 'Marginals' and 'Dependence'. The 'Marginals' tab is active, showing a table of variables and their distributions:

Variable	Distribution
1 <input checked="" type="checkbox"/> h1	Normal
2 <input checked="" type="checkbox"/> h2	Normal
3 <input checked="" type="checkbox"/> rho	Normal
4 <input checked="" type="checkbox"/> cp	Normal
5 <input checked="" type="checkbox"/> lamb	Normal

Below the table is an 'Import Morris result' button. To the right of the table, a 'PDF' plot shows a normal distribution curve for the variable 'h1'. The x-axis is labeled 'h1' and ranges from 500 to 1500. The y-axis is labeled 'Density' and ranges from 0 to 0.004. Below the plot, the 'Parameters' section shows the 'Type' as ' μ, σ ', with ' μ ' set to 1000 and ' σ ' set to 100. The 'Truncation parameters' section has two checkboxes: 'Lower bound' and 'Upper bound', both of which are unchecked.

Step by step example (5/6)

- OpenURNS study steps
 - Launch the computation on the remote cluster

The screenshot shows the OpenURNS application window. On the left, a tree view under 'Studies' shows 'OTStudy_0' expanded, with 'Physical models' and 'Central tendency' (containing 'centralTendency_0') selected. The main area is divided into several sections:

- Analysis parameters:** A table with the following values:

Algorithm	Monte Carlo
Outputs of interest	[temp,balance]
Confidence level	95%
Maximum coefficient of variation	-1
Maximum elapsed time	- (s)
Maximum calls	1000
Block size	1000
Seed	0
- A progress bar showing 0% completion.
- Buttons for 'Run' and 'Stop'.
- The analysis is ready to be launched.**
- Launching parameters:** A section with a checked 'Parallelize status' and a 'Computing resource' dropdown set to 'porthos'.
- Specific parameters for clusters:** Fields for:
 - Number of processes: 32
 - Remote working directory: /I35256/workingdir/run_Thu_Jan_4_11_46_02_2018
 - Local result directory: /tmp
 - Working Characterization Key: P11U50:CARBONES
 - Time limit: 0 hours, 10 minutes
- Input files:** A text area containing a list of files, including training data, scripts (syntesis.py, run.sh, Makefile), and a Mesh file.

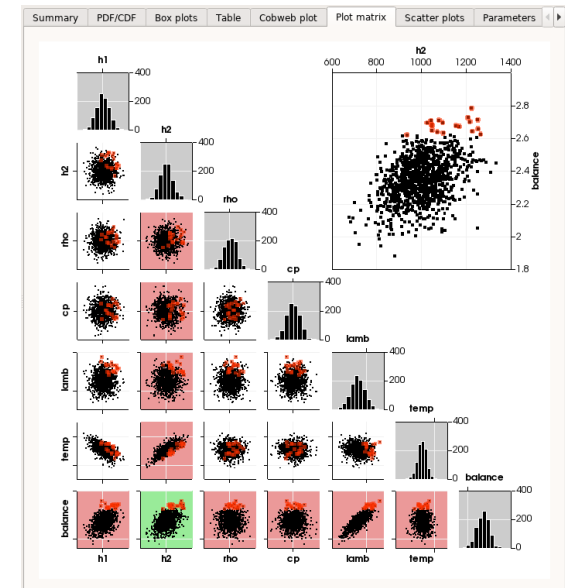
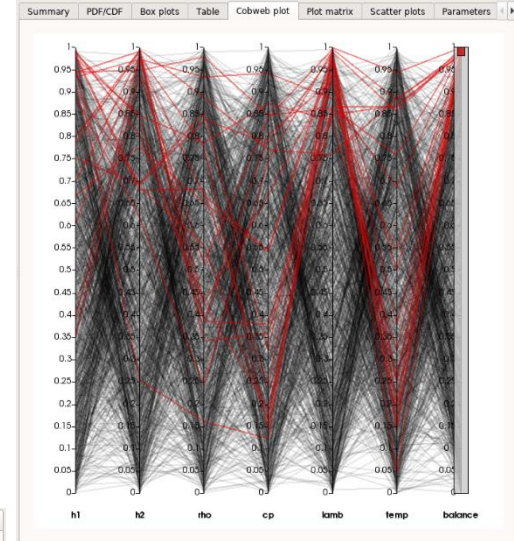
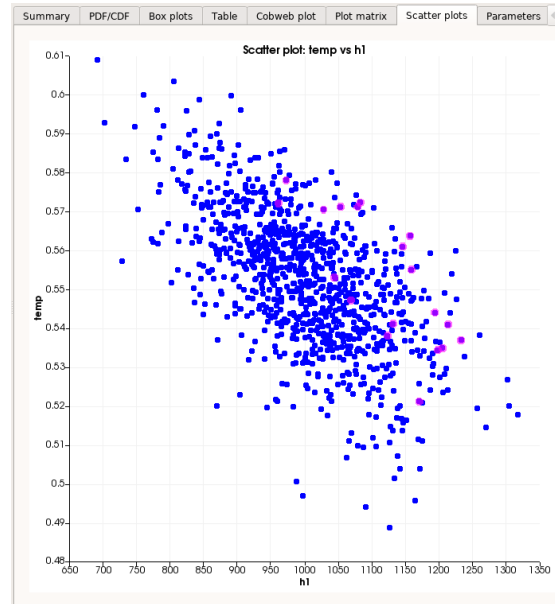
Step by step example (6/6)

■ OpenTURNS study steps

□ Result analysis

- Box plots, Cobweb plot, Plot matrix, Scatter plots

Summary PDF/CDF Box plots Table Cobweb plot Plot matrix Scatter plots Parameters								
Size : 1000								
Row ID	balance	cp	h1	h2	lamb	rho	temp	
0	809	2.78221	88.2456	971.477	1221.99	57.2548	107.978	0.577949
1	395	2.72433	100.421	1027.2	1204.91	53.5571	105.327	0.570425
2	677	2.7117	90.049	1155.87	1249.82	49.4445	98.4765	0.563661
3	900	2.70897	103.667	1231.73	1075.55	51.2866	98.5621	0.536851
4	45	2.706	91.837	1196.88	1039.72	52.682	105.865	0.534277
5	292	2.69981	115.659	1145.02	1213.72	49.833	114.578	0.56085
6	923	2.69412	108.327	1212.31	1092.05	50.7304	97.5064	0.54087
7	163	2.69324	106.613	1122.3	1022.64	54.2267	93.2996	0.537973
8	791	2.68221	89.2415	1204.1	1042.94	51.5744	102.454	0.534795
9	470	2.67682	109.86	960.104	1151.35	54.7742	101.376	0.572025
10	790	2.67592	94.6885	1130.53	1045.08	52.7527	111.048	0.541049
11	339	2.67058	110.041	1157.85	1166.83	49.4117	119.54	0.554934
12	280	2.65655	95.7377	1077.96	1250.38	49.0954	96.3021	0.571101
13	897	2.64857	107.58	1068.01	1044.95	53.1258	110.482	0.547118
14	851	2.6413	96.2933	1052.35	1225.55	49.5096	97.5399	0.571084
15	312	2.6371	100.4	1043.4	1068.36	52.6973	104.643	0.552994
16	802	2.63162	111.756	1192.92	1094.24	48.8332	108.194	0.543942
17	940	2.62326	107.312	1081.83	1261.66	47.7306	112.227	0.57224
18	217	2.61947	87.4748	1169.48	932.699	52.9009	90.4976	0.521085
19	325	2.61764	89.9805	1058.81	1125.39	50.322	118.691	0.559331
20	919	2.61495	110.336	1140.58	913.633	53.6697	89.6917	0.519977
21	301	2.61247	101.447	1093.59	978.467	52.8717	109.807	0.535357



OpenTURNS Graphical User Interface - Distribution and future evolutions (1/2)

- **Distribution of SALOME + OpenTURNS platform**
 - LGPL license for the whole platform (SALOME + OpenTURNS + OPENTURNS GUI)
 - Download SALOME platform with OpenTURNS here:
 - http://www.salome-platform.org/contributions/copy_of_combs

OpenTURNS Graphical User Interface - Distribution and future evolutions (2/2)

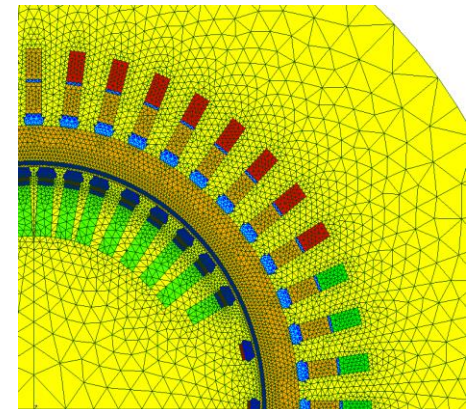
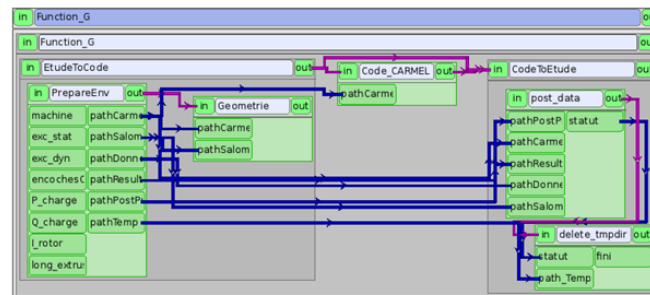
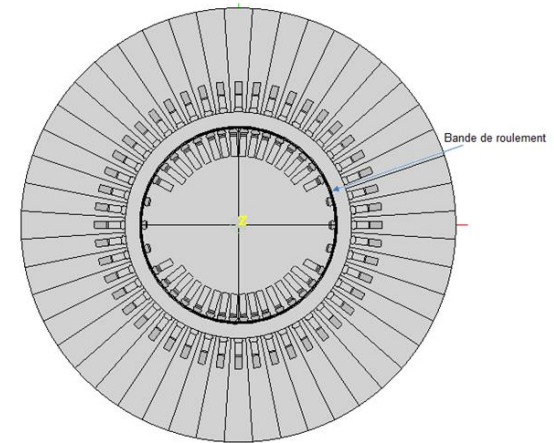
- **Improvements to come**

- Launch, leave and reconnect to a job
- Avoid multiple job submissions for one OpenTURNS study
- Management of a partially executed study
- Management of failed computations

Examples of OpenTURNS studies with HPC (1/3)

■ Probes in an alternator

- Optimization of the position of probes in an alternator to maximize the signal fidelity
- Solver: CARMEL
- Elementary calculation: Mesh with ~200000 cells. 92 hours
- OpenTURNS study: 115 elementary calculations → about 10000 CPU hours

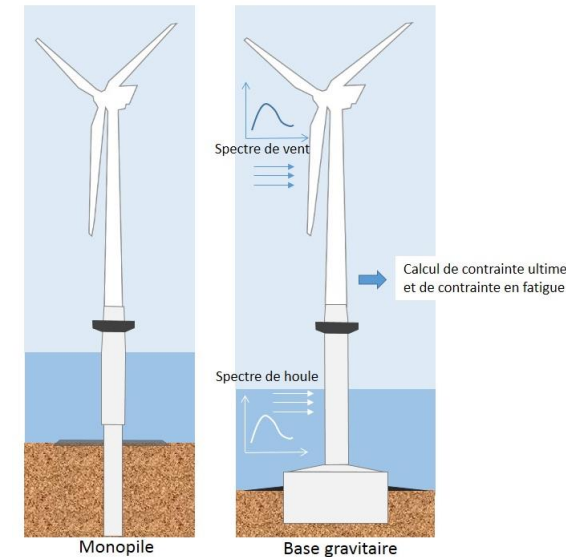


Desquiens 2016

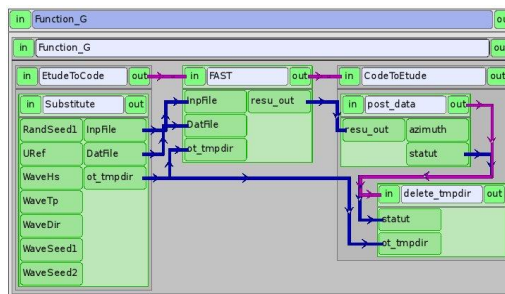
Examples of OpenTURNS studies with HPC (2/3)

■ Offshore wind turbine

- Study of lifetime of offshore wind turbine sustaining extreme scenarios in terms of strain.
- Solver: FAST
- Elementary calculation: No mesh. ~10 minutes.
- OpenTURNS study: 10000 elementary calculations → about 17000 CPU hours

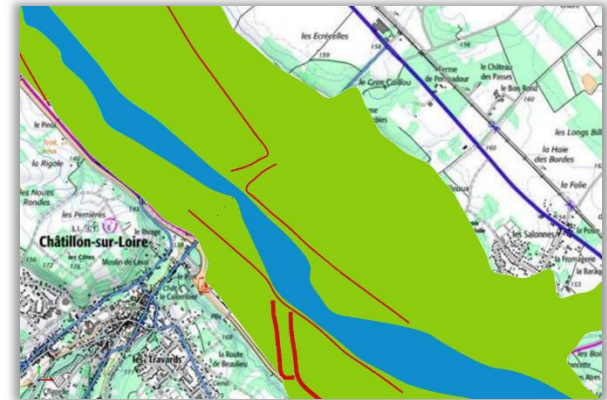
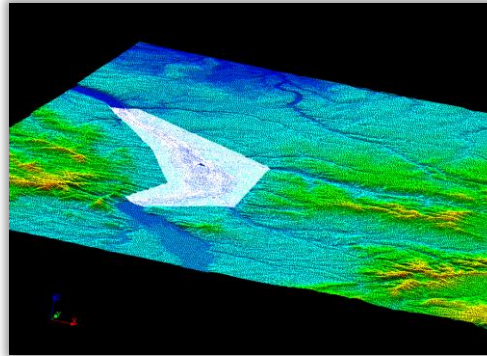
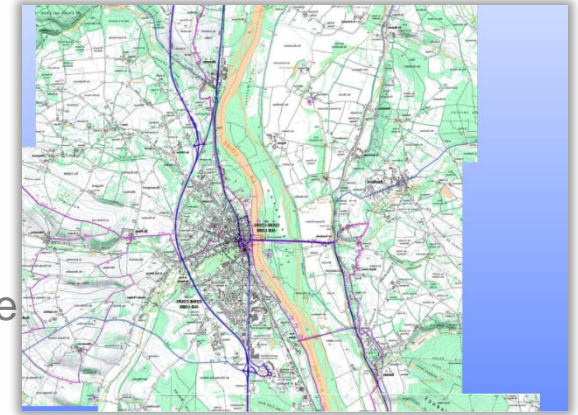


Bousseau 2016



Examples of OpenTURNS studies with HPC (3/3)

- Evaluation of the environment impact of the flood of a river
 - Solver: TELEMAC/MASCARET
 - Elementary calculation: Mesh with ~100000 cells. Convergence reached in about ~1-10 hours



Questions?