The Uranie platform

The CEA/DEN Uncertainty and Sensitivity platform

Fabrice Gaudier

CEA/DEN/DANS/DM2S/STMF/LGLS

fabrice.gaudier@cea.fr

28e Forum ORAP

CNRS, Paris

October 14, 2011
Outline

• Introduce the ROOT framework
• Present the Uranie platform
• Applications of Uranie
CERN Large Hadron Collider (LHC)

- particle accelerator
- 27 km circumference tunnel in Geneva
- 4 experiments (ATLAS, CMS, ALICE, LHCb)

Study the structure of matter
- Search for the Higgs boson
- Search for new physics

- Data quantity generated: 20 PetaBytes/year
- ROOT is the framework to store, treat and analyze this data
ROOT ( http://root.cern.ch )

ROOT is an object-oriented framework for large scale data analysis and data mining.

- 20 years of development (C++ with 3-4 releases/year)
- multi-plateform (Unix, Windows, Mac OS X)
- Offer:
  - A C++ interpreter, but also python (PyROOT), ruby
  - A hierarchical object-oriented database (machine independent, highly compressed, supporting schema evolution and object versioning)
  - Shared libraries (automatic loading with "rootmap")
  - Advanced statistical analysis tools (subprojects RooStats, RooFit, TMVA)
  - Advanced visualization tools
- LGPL License
URANIE : CEA/DEN Uncertainty Platform

- **Root** (CERN), **Mixmod** (Gaussian Mixtures - INRIA), **OPT++** (Optimization - Sandia), **Club** (Text parsing - CNES)

- Data access:
  - Flat file with header (”Salomé Table”)
  - TTree (internal ROOT)
  - SQL Data base (MySQL, PostgreSQL, ...)

- Uncertainty/Sensitivity methods in URANIE
  - Design Of Experiments (SRS, LHS, ROA, qMC, MCMC, Copulas)
  - Clustering methods
  - Surrogate models (Polynomial, Artificial Neural Networks, Splines)
  - Non Intrusive Spectrale Projection : Generalized Polynomial Chaos
  - Sensitivity Analysis (Pearson, Spearmann, Morris, Sobol, FAST & RBD)
  - Optimization, Multi-Criteria (**Vizir** library : Genetic Algorithms)
  - Computing distribution
URANIE : Functional diagram

ROOT . . .

Uranie . . .

Application
• 62 000 lines & 111 classes

• Version of ROOT :
v5.26 (2009 Dec.)
v5.30 (2011 Jan.)

• Compilation with **cmake**
  (Linux-Makefiles/Windows-Visual Project)

• Unitary tests with **CppUnit**

• **Exceptions**
  (Warning, Error, Deprecated)

• **User Manual with DocBook**
  Generate the HTML and PDF documents from the same XML files

• **Developer Manual with Doxygen**
• Uranie Training sessions
  – 2011/10/17-19 : session "user" session - SAC CEA/DEN (8)
  – 2011/04/4-6 : session "user" session - NURISP (5)
  – 2011/01/19-21 : "user" session - SAC CEA/DEN (12)
  – 2010/10/13-15 : "user" session - SAC CEA/DEN (12)
  – 2010/02/25-26 : "user" session - MAR CEA/DEN (15)
  – 2009/10/19-20 : "integrator" session - NURISP (2)
  – 2009/09/7-8 : "user" session - projet ESS + DSM (5)
  – 2009/06/10 : "user" session - NURISP (3)
  – 2009/04/06 : Areva-TA (4)
  – 2008/10/28-29 : Areva-TA (6)
  – 2006-2008 : CEA/DEN (~ 50 persons)

• ROOT Training sessions at INSTN (12 students, 3 days)
  – February 2011 : DEN-SAC
  – February 2010 : DEN-SAC
  – June, 17-19 2009 : DEN-CAD
  – December, 3-5 2008 : DEN-SAC
  – March, 26-28 2008 : DEN-SAC

• Development is piloted by user request
URANIE : Batch mode

> root myScript.C
URANIE - XML User Interface

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE Problem SYSTEM "//uranie.dtd">

<Problem>
  <Header name="boreholeXML" title="Launch the Borehole function in XML" debug="0"/>
  <Application name="uranie" version="1.0"/>

  <DataDictionary>
    <DataField name="rw" law="uniform" min="0.05" max="0.15"/>
    <DataField name="r" law="uniform" min="100.0" max="50000.0"/>
    <DataField name="tu" law="uniform" min="63070.0" max="115600.0"/>
    <DataField name="tl" law="uniform" min="63.1" max="116.0"/>
    <DataField name="hu" law="uniform" min="990.0" max="1110.0"/>
    <DataField name="hl" law="uniform" min="700.0" max="820.0"/>
    <DataField name="l" law="normal" mean="1120.0" std="12.25"/>
    <DataField name="kw" law="uniform" min="9855.0" max="12045.0"/>
  </DataDictionary>

  <Sampler method="LHS" N="1000" export="waterhole_sampler_lhs.dat"/>
  <Launcher macro="UserFunctions.C" function="flowrateModel" output="ymod" export="waterholelhs.dat"/>
  <Sampler method="SRS" N="2000"/>
  <Launcher function="HoXuSurrogateModel" input="rw:r:tu:tl:hu:hl:l:kw" output="yhxs"/>
</Problem>
```

```c++
void evaluateXMLFile (TString xmlFile = "uranieproblem.xml")
{
  TXMLProblem * xmlProblem = new TXMLProblem(xmlFile);
  xmlProblem->submit();
}
```
Projects using URANIE

- LEONAR tool for severe accidents in french nuclear reactor (CEA-EDF)
- PSI-Matadator Methodology: Dosimetry computation in french nuclear reactor (CEA-EDF)
- EHPOC project: Meteor code (CEA)
- Sensitivity Analysis for Cathare code (Areva TA)
- Multi-criteria optimization (CEA CESTA/CELIA)
- Optimization Design for ESS
  The "European Spallation Source" is a project to design and construct (Lund - Sweden) a next generation facility for research with neutrons.
- ALLIANCES platform (CEA/ANDRA/EDF)
  is to provide a working environment for the simulation and analysis of phenomena to be taken into account for waste storage and disposal studies.
- European project NURESIM/NURISP
  The European Platform for NUclear REactor SIMulations, NURESIM, is a Common European Standard Software Platform for modeling, recording, and recovering computer data for nuclear reactors simulations.
Uranie Application with ANDRA : GdR MoMaS

• Context

Make Sensitivity Analysis with Surrogate Model : Artificial Neurals Networks ("ANN")

• CPU time single calculation : $1 < t < 5$ hours (Cluster)

• Original DataSet
  
  – $n_X = 17$ input variables
  
  – $n_Y = 5$ output variables
  
  – Build the 5 ANN : $n_S = 1500$ patterns

• Neural networks
  
  – different architectures (MLP)
  
  – Input with a logarithm PDF $x := \log(x)$

  – Output $y := \frac{1}{1+\log(y)}$

  – cross validation

  – Validate the 5 ANN on another dataset
    with $n_S = 1000$ patterns
### Computation Sensitivity Indexes (SI)

**First Order:**

\[
S_i = \frac{\text{Var}[\mathbb{E}[Y|x_i]]}{\text{Var}[Y]}
\]

1. **Regression Analysis**

\[
\hat{y} = b_0 + \sum b_i x_i
\]

- on the Values: "SRC" ("Standardised Regression Coefficient")
- on the Ranks: "SRRC" ("Standardised Rank Regression Coefficient")

The regression is valid when \( R^2 = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2} \) close to 1.0 (\( \geq 0.7 \))

2. **Brute-Force Method**

Cost: \( nX \times nCV \times nPts \)

- \( nX \) = number of inputs parameters
- \( nCV \) = n. of conditionnal values
- \( nPts \) = n. of pts over cond. values

Use the surrogate model ("ANN") in place of the true code
Original Database (nS = 1500) & ”s1”

ROOT . . .
Uranie . . .
Application
Original Database (nS = 1500) & other outputs

ROOT . . .
Uranie . . .
Application
Estimation of SI with ANN and $n_S = 1500$

Design $n_L = 100$ databases with size $n_S = 1500$ points

Simulate these databases using ANN surrogate models

Compute the first order SI by the Brute-Force method
Estimation of SI with ANN and nS = 1500
Same with nS = 20000 (« s1 » and « s2 »)
Same with $nS = 20000$ ( « s3 » and « s4 »)

1 500 points

20 000 points
Same loop (nL = 100 DoE) but for nS=2000 to 20000 by step 2000:

ROOT . . .
Uranie . . .
Application
Uranie Application : All simulations (2/2)

1. 5 inputs parameters are not influence (first order) on all the 5 outputs

2. Only 4 inputs parameters are sensitives (for all 5 outputs)

```cpp
ttre->Draw("Sobol:Var","","BOXCOL2Z")
```

```cpp
ttre->Draw("Var:Star","Sobol>0.3","BOXCOL2Ztext")
```
GPU - Description of the test

- Output: $\textbf{st4}$ with $nH = 16$ neurons in the hidden layer ($\mathbb{R}^{17} \rightarrow \mathbb{R}^{16} \rightarrow \mathbb{R}$)

- Learning:
  - 1 database with 1 400 patterns
  - 100 learning

- Running:
  - 6 databases with 1 500, 5 000, 10 000, 20 000 and 50 000 patterns
  - 10 running for each databases

- Criteria for comparison: time ($t$) and error ($e$)
  1. CPU: 2 quadri cores Xeon 5500 (Boost.uBLAS)
  2. GPU: 8 Fermi Tesla C2050 (CUDA/cuBLAS)
GPU - Learning Performance

Mean: GPU 42 s versus CPU 115 s
GPU - Evaluation Performance

Running time (w.r.t. input size)

Running time ratio (w.r.t. input size)

ROOT . . .
Uranie . . .
Application
GPU - Profiling

Height Plot

Column (CPU Time) Plot

0.00% 9.11% 18.22% 27.33% 36.44% 45.55%

scatter_plot

ROOT . . .

Uranie . . .

Application
GPU - Conclusions

1. GPU is $2 \rightarrow 23$ faster than CPU for evaluation ANN

2. GPU is only 3 faster than CPU for learning ANN for small database
   same speed-up for large database?

3. main time consumer: matrix computing
Conclusions

• Introduce the ROOT framework
• Present some libraries of the Uranie platform
• Applications of Uranie:
  – Sensitivity Analysis with Artificial Neurals Networks
  – Learning and Running ANN on GPU