High Performance Computing
Uncertainty Quantification

C. Prud'homme

Laboratoire Jean Kuntzmann – EDP
Université de Grenoble

Cadarache
July 07 2011

Disclaimer

A compilation of presentations

- from the OPUS 5-th workshop on HPC and UQ. http://www.opus-project.fr/index.php/aroundopus/workshopreports
- from discussions with Sandia people prior to 5th workshop
- Some more personal considerations.

Contributors

C. Perez (LIP), R. Barate(EDF), F. Gaudier(CEA), D. Busby(IFPEN)

M. Heroux, J. Kamm, B. Adam, E. Philips, M. Eldred (Sandia)

Outline

1. UQ, M&S, SA, V&V and HPC
2. Trends in High Performance Computing
   - Advances in Hardware
   - Advances in Software
     - HPC Programming models
     - Software components assembly
     - HPC software development model
3. UQ, HPC, Software environments

Uncertainty Sources

Uncertainty Sources

- Inherent variability (e.g. industrial processes)
- Epistemologic uncertainty (e.g. model constants)

Epistemic Uncertainty

- Epistemology = "what distinguishes justified belief from opinion."
- Lack of knowledge about, say, the appropriate value to use for a quantity, or the proper model form to use.
- "Reducible uncertainty" : can be reduced through increased understanding (research) or more, relevant data.

Aleatory Uncertainty

- "Alea" = Latin for "die" ; Latin aleator = "dice player."
- Inherent randomness, intrinsic variability.
- "Irreducible uncertainty" : cannot be reduced by additional data.
- Usually modeled with probability distributions.

Uncertainty Quantification : Some facts

- UQ in computational science is the formal *characterization, propagation, aggregation, comprehension, and communication* of aleatory (variability) and epistemic (incomplete knowledge) uncertainties.
  - E.g., demand fluctuations in a power grid are aleatory uncertainties.
  - E.g., incomplete knowledge about the future (scenarios uncertainty), the validity of models, and inadequate "statistics" are epistemic uncertainties.
- A huge range of technical issues arise in the M&S components of problem definition and execution phases.
- Another huge range of technical issues arises in the delivery phase, especially in high-risk decision environments.
- "Probability" is the main foundation for current "quantification."
- More complex epistemic uncertainties, for example arising in human interaction modeling, lead to other quantification formalisms (evidence theory, fuzzy sets, info-gap methods, etc.)

UQ impact on HPC

Any large-scale computational problem's computing requirements increase (usually significantly) with UQ.

SA : Sensitivity Analysis

SA is ONE procedure under the overall UQ umbrella — it helps drive parsimony.

- SA seeks to quantify at the influence of the uncertainty in the input on the uncertainty of the output
- SA strives to help answer the question : "How important are the individual elements of input x with respect to the uncertainty in output y(x) ?"
- SA can be used to :
  - Rank input parameters in term of their importance relative to the uncertainty in the output ;
  - Support verification and validation activities ;
  - Drive, as part of an iterative process, uncertainty quantification (UQ) analyses towards the input parameters that really matter.
- SA is typically the starting point for a more complete UQ.
- SA is not a replacement for full UQ or V&V.

V&V : Verification and Validation

Verification

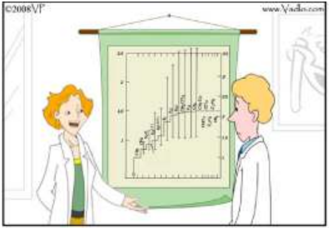
- Verification seeks to answer the question :
  "Is my computational model, as instantiated in software, solving the governing equations correctly ?"
- Verification is primarily about math and computer science.
- Verification comes in different flavors :
  - software and code verification
  - calculation verification

Validation

- Validation seeks to answer the question :
  "Is my computational model, as instantiated in software, solving the proper governing equations ?"
- Validation is primarily about modeling and physics.
- Validation necessarily involves data.
- Validation intersects with other difficult problems :
  - Calibration
  - Uncertainty Quantification
  - Sensitivity Analysis

What is Uncertainty Quantification (UQ) ?

- Broadly speaking, UQ seeks to gauge the effect of system/model uncertainties on the observed/ computed outputs.
- The execution of UQ in M&S and the delivery of “prediction” typically has two distinct components :
  - Characterization of uncertainty, typically quantitative characterizations for physical science M&S.
  - Reduction of uncertainty for purposes of improving prediction “accuracy.”



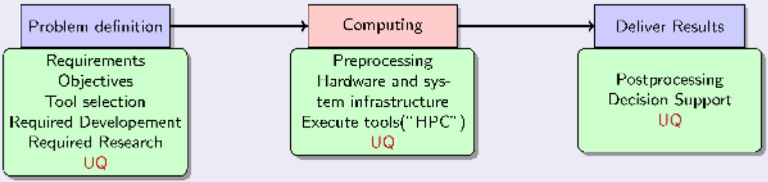
Do you really have to show the error bars?

Error Bars

in numerical simulations, they are ONE element of characterized uncertainty. And yes we should display error bars !

UQ and M&S

Consider Modeling and Simulation (M&S)



UQ is everywhere !

The presence of acknowledged uncertainty is fundamental — and fundamentally challenging.

- It complicates all aspects of the computational science that are required to address the problem.
- Choice of non-intrusive UQ is predominant at the moment but intrusive UQ(e.g. Galerkin methods) gains lots of momentum
- V&V is just as ubiquitous !

From computational analysis to support high consequence decisions

Simulation capability, demand on software ↑

- Systems of systems
- Optimization under Uncertainty
- Quantify Uncertainties/Systems Margins
- Optimization of Design/System
- Robust Analysis with Parameter Sensitivities
- Accurate & Efficient Forward Analysis
- Forward analysis

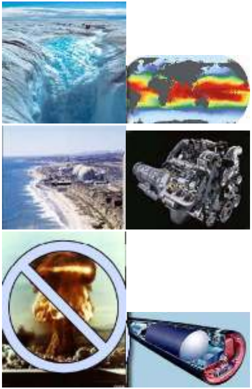
Each stage requires greater performance and error control of prior stages
Always will need :

- more accurate and scalable methods.
- more sophisticated tools.

Towards Exascale : The example of US DOE

Missions/Challenges for US-DOE and Assoc. Nat. Labs

- Climate Change : Understanding, mitigating and adapting to the effects of global warming
  - Sea level rise
  - Severe weather
  - Regional climate change
  - Geologic carbon sequestration
- Energy : Reducing countries reliance on foreign energy sources and reducing the carbon footprint of energy production
  - Reducing time and cost of reactor design and deployment
  - Improving the efficiency of combustion energy systems
- National Nuclear Security : Maintaining a safe, secure and reliable nuclear stockpile
  - Stockpile certification
  - Predictive scientific challenges
  - Real-time evaluation of urban nuclear detonation



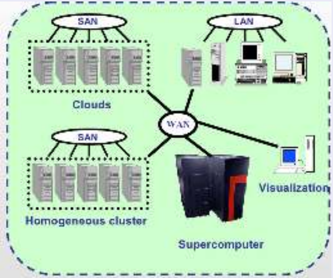
Accomplishing these missions requires exascale resources.

Outline

1 UQ, M&S, SA, V&V and HPC
2 Trends in High Performance Computing
- Advances in Hardware
- Advances in Software
  - HPC Programming models
  - Software components assembly
  - HPC software development model
3 UQ, HPC, Software environments

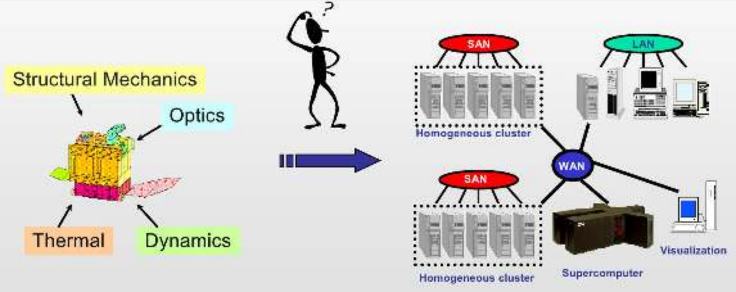
Context : Super/Grid/Cloud/Sky Computer

- Computing resources
  - Homogeneous clusters
    - Many-core nodes
    - With/without GPU
  - Supercomputers
  - Grids
  - Desktop Grids
  - Clouds
- Hierarchical networks
  - WAN
    - Internet, Private WAN, etc.
  - LAN
    - Ethernet
  - SAN
    - Infiniband, ...



- Fast evolution !
- Heterogeneity !

How to displatch applications on ressources ?

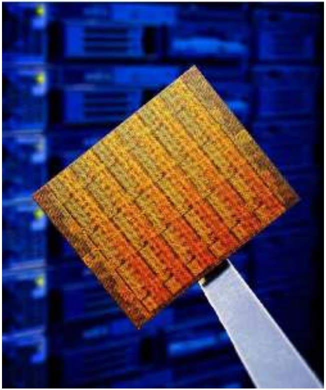


Standard Hardware

- Interconnected set of nodes
  - 1 node = 1 mono-processor + memory + 1 network card
- Network
  - Latency 1 micro-seconds
  - Bandwidth 10+ Gb/s
- Recent evolutions
  - 1 processor to several processors
    - Systems with up to 32 processors on a board
    - Usually a few
  - Mono-core processor to multi-core processor
    - "Classical" 4 12+ cores
    - Terascale project : 80 cores
  - IBM Power 7
    - "Programmable" L3 caches

Intel Single-Chip Cloud Computer

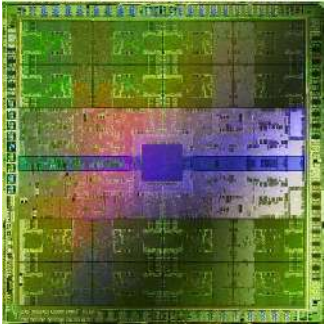
- Research processor
  - 48 cores Scalable to 100+
- Interconnected
  - 24-router mesh network
  - 256 GB/s bisection bw
- Hardware support for message-passing !
  - No more shared memory



Reproduce at the scale of a processor a cluster using message passing interfaces for communication

From GPU to GPGPU

- GPU SIMD
  - G80 : 128 CUDA cores
  - GT200 : 240 CUDA cores
  - Fermi : 512 CUDA cores
- From "graphic" model...
  - Few shared memory
  - Optimized for graphics
  - No IEEE 754, no ECC
- To a "compute" model
  - IEEE 754-2008
  - ECC supports
  - More shared memory
  - L1/L2 caches
  - Unified Address Space
  - Concurrent kernels
  - ...



3.0 billion, 512 SP Fermi core

HPC Cloud

- Clouds
  - Virtualization-based
  - On-demand computing
  - Clouds usually based on “low” performance processors
    - Almost no constraints on network
    - But for fault-tolerance ?
- HPC Clouds
  - Powerful compute node, lot of memory
    - Nodes with GPU available
  - Guarantees on network performance
    - E.g., 10 Gbps, low latency
  - Running an application on 4096 cores (512 nodes) on XLarge EC2
    - Feasible but not straightforward (infrastructure failure, node availability, etc)
    - Cost : $418/h
  - Some attempts to put a supercomputer (Blue Gene) in a cloud
    - Show real costs of HPC computing

Towards Exascale computing

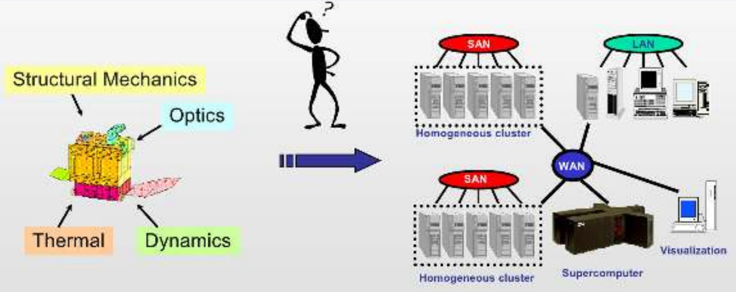
- Top 500
  - 1st is 186,368 cores, 2.5 petaflops, 4 MW
  - 2nd is 224162 cores, 1.8 petaflops, 7 MW
- Technology trends from IESP [a] Roadmap
  - Increase concurrency up to ten billion threads
    - Moore law is still in effect
    - Frequency wall
  - Increase reliability
  - Decrease power consumption
    - Limit a system to a few tens of MW
- Other big issue
  - I/O

a. IESP : The International Exascale Software Project, `http://www.exascale.org`

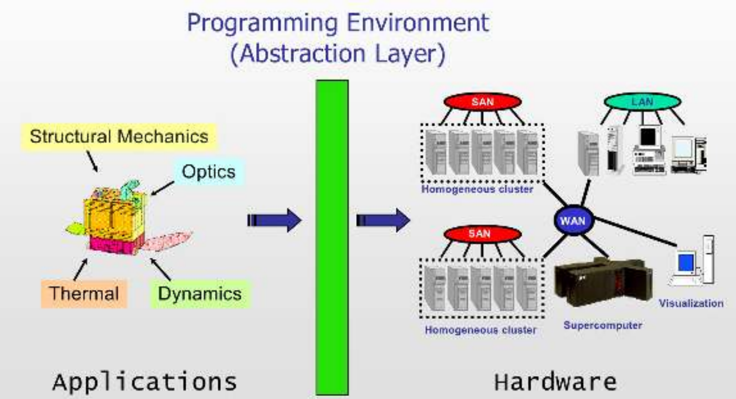
Conclusions on hardware advances

- GP-GPUs or how to re-use SIMD
- SCCC or how to put a cluster (MPI) into a chip
- Clouds or how to make application adaptable
- System with billions of core approaching
- Fault tolerance as a path to exascale
- Energy efficiency as a new metric/constraint

Applications on ressources



Applications on ressources : Abstractions !



From low level to high level abstractions

- From low level to high level of programming abstraction
- Low level == close to hardware abstractions
  - Ex. CUDA, OpenCL, MPI, ...
  - Low portability
  - High complexity
  - High efficiency
- High level == close to functional abstractions
  - Ex. OpenMP, PGAS, ...
  - High portability
  - Low complexity
  - ? efficiency

Low level "Programming Models"

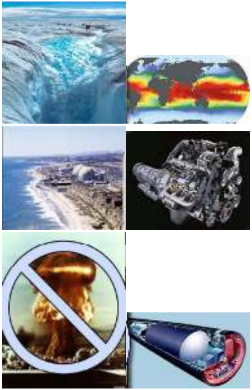
- Programming GPUs
  - From CUDA 1.0 to CUDA 4.0
  - 32 bits to 64 bits, simple to double floats
  - Multiple specialized memories to unified virtual addressing (CPU+GPU)
  - Single GPU to multiple GPUs
  - SIMD to MIMD
- OpenCL
  - Portable across GPGPUs, multi-core, "CELL", etc.
  - Data & task parallelism
  - Compile code at runtime
    - Enable optimized code for targeted hardware
- Message passing
  - Towards MPI 3.0
  - Non blocking collective operations
  - Fault tolerance
  - Hybrid programming : Pthreads/OpenMP, GPU, PGAS

Observations

- MPI-Only is not sufficient, except . . . much of the time.
- Near-to-medium term :
  - MPI+[OMP—TBB—Pthreads—CUDA—OCL—MPI]
  - Long term, too ?
- Concern :
  - Best hybrid performance : 1 MPI rank per UMA core set.
- Long- term :
  - Something hierarchical, global in scope.
- Conjecture :
  - Data-intensive apps need non-SPMD model.
  - Will develop new programming model/env.
  - Rest of apps will adopt over time.
  - Time span : 10-20 years.

Applications : Numerical Scientific Simulations

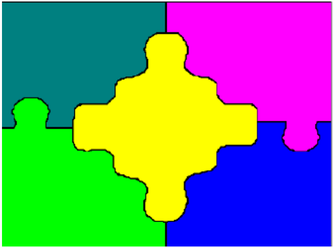
- More and more complex
  - how to handle code coupling ?
- Need model to handle such complexity
  - weak coupling through runtime interface and software components
  - stronger coupling through API



- Software components and Assembly of software components (e.g. Salome framework)
- But HPC software development model also change... (e.g. The example of Sandia with Trilinos/Dakota)

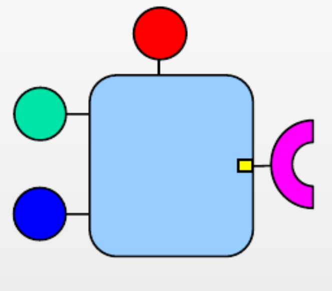
Software Components

- Technology that advocates for composition
  - Old idea (late 60's)
  - Assembling rather than developing
- Aim of a component : to be composed !
  - Spatial composition
  - Temporal composition
- Example of success stories
  - Pipe (in Operating Systems)
  - Data flow models



Software component : Blackbox and Ports

- A component is a black box that interacts by its ports
- A component is a black boxPort access point
  - Name
  - Description and protocol
- Usual interactions
  - (Object) Interface
  - Message passing

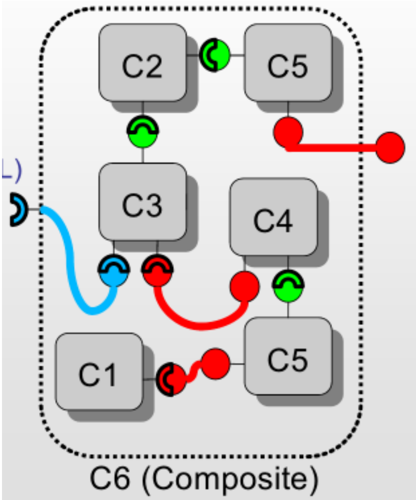


```
component MyComponent
{
provides IExample1 to_client1;
provides IExample2 to_client2;
uses     Itfaces2  to_server;
};
```

```
interface IExample1
{
void factorise(in Matrix mat);
};
interface IExample2 { ... };
interface Itfaces2 { ... };
```

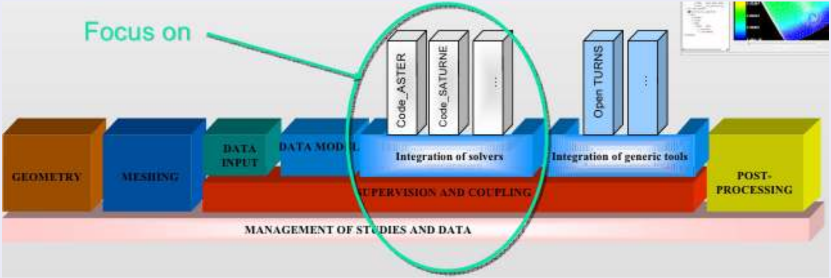
Assembly of Software Components

- Description of an assembly
  - Dynamically (API)
  - Statically
- Architecture Description Language (ADL)
  - Describes
    - Component instances
    - Port connection
  - Available in many CM
- Primitive and composite components



Example : The Salome Platform

- Platform for pre/post processing and integration of solvers for numerical simulation
- Developed in Open Source cooperatively by EDF and CEA
- http://www.salome-platform.org



Sandia HPC Software

Trilinos : HPC solution components

- Compatible space-time discretizations
- Linear & nonlinear solvers
- Partitioning and dynamic load balancing
- Automatic differentiation
- Optimization (fully coupled)
- `http://trilinos.sandia.gov`

Dakota : Risk informed decision making

- Optimization (multiparallel, surrogate,..)
- UQ (aleatory & epistemic)
- New sparse-collocation methods ...
- Optimization under uncertainty
- Rapid deployment of new algorithms
- `http://dakota.sandia.gov`

Strong algorithms R&D...

Trilinos HPC solution

Cutting edge algorithmic research on :

- Sparse, distributed, parallel linear algebraIterative and direct parallel linear solvers,
- Iterative, parallel eigensolvers,
- Multilevel parallel preconditioners,
- Load-balancing, AD, and more...

State-of-the-art solver framework :

- Generic, object-oriented programming
- Extensible, scalable design
- Common core for linear algebra with abstract solver API
- SQE infrastructure with requirements and best practices for SQA

Impact

- Available in most ASC applications codes
- Impacts several CRADA projects
- Used by all major DOE labs (¿2300 registered users)
- 2004 R&D 100 award
- IEEE 2004 HPC Software Challenge Award

Trilinos : Hierarchical software framework

Tighter software integration

- Trilinos[a] is an evolving framework to address these challenges :
  - Fundamental atomic unit is a package.
  - Includes core set of vector, graph and matrix classes (Epetra/Tpetra packages).
  - Provides a common abstract solver API (Thyra package).
  - Provides a ready-made package infrastructure
  - Specifies requirements and suggested practices for package SQA.
- In general allows to categorize efforts :
  - Efforts best done at the Trilinos level (useful to most or all packages).
  - Efforts best done at a package level (peculiar or important to a package).

a. translates to "A String of Pearls"

Allows package developers to focus only on things that are unique to their package.

Conclusions

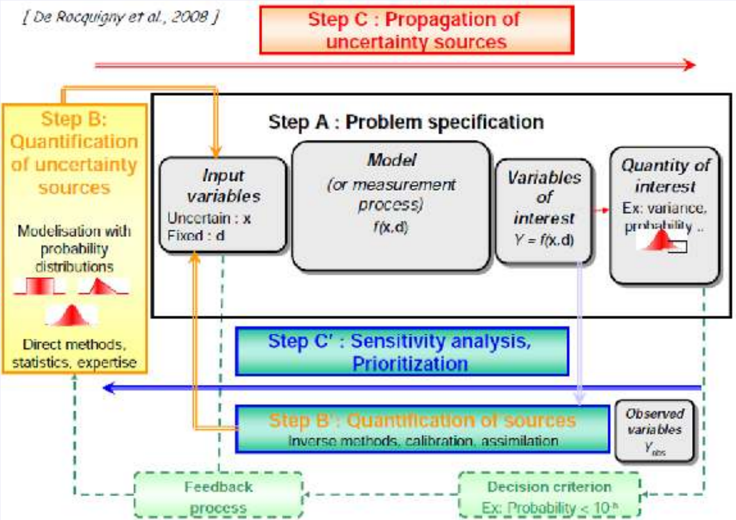
- Evolution of hardware
  - CPU and GPU towards many-core computing
  - Hierarchy of cores and of memories
  - Importance of placement and data movement
  - Energy and fault tolerance major issue for exascale
- Programming model
  - Many "low" level /specialized models
  - Component model as a programming model
  - Many composition operators has been already defined & prototyped
  - Proof-of-concepts for AMR & Salome

Outline

1. UQ, M&S, SA, V&V and HPC
2. Trends in High Performance Computing
   - Advances in Hardware
   - Advances in Software
     - HPC Programming models
     - Software components assembly
     - HPC software development model
3. UQ, HPC, Software environments

UQ Process Current Status

Software design



UQ in France I

- Software design built upon UQ process current status
- Three (non-exclusive) ways to handle computational complexity :
  - use accelerated MC strategies
  - use meta-modelling instead of the forward code
  - increase computing power (HPC)

Shared view

No brute force, Need for specific methods and tools.

UQ in France II

Frameworks for UQ

- Cougar (IFPEN & partners) — reservoir simulation
  - Commercial
  - Features
    - Sensitivity analysis (variance based)
    - Parametric surface responses (polynomials)
    - Non-parametric surface responses (kriging)
    - Adaptive planification (towards decision making)
  - Grid computing
- Uranie (CEA/DEN & partners) — nuclear plants
  - Not distributed
  - Features
    - 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 (library Vizir : Genetic Algorithms)
    - Computing distribution
  - Some level of integration with Salome

UQ in France III

Frameworks for UQ

- OpenTURNS (EDF/EADS/Phimeca)
  - Open Source (e.g. Debian/Ubuntu)
  - Features
    - Provide various methods for all steps in actuall UQ process
    - Integrated into Salome (Module OpenTURNS not distributed at the moment ?)
    - Provides blackbox wrapper based on Python and XML
    - Two ways of exploiting HPC :
    - Multi-threading (wrapper)
    - Grid computing (wrapper with batch scheduling, Salome level)

Some Conclusions : UQ & HPC I

- No real parallel effort in UQ codes
- Non intrusive approach is still the norm (blackbox), however emerging intrusive methods will require intrusive use of hpc
- Interface with computational ressources : at the moment use of batch system
- Need for middleware(mpi, grid...) infrastructure : grid middleware such as DIET seems to be an answer to the current UQ needs.
  DIET : Distributed Interactive Engineering Toolbox
  `http://graal.ens-lyon.fr/DIET`
- Communication with computational codes :
  - use of input/output files and databases to manipulate data and communicate between simulation codes and UQ codes
  - Integration through Salome
- Interface with users : GUI, domain specific (embedded) languages
- Postprocessing tools are very important (towards high consequence decision making)

Some Conclusions : UQ & HPC II

- UQ Software framework : Developments well aligned on both sides of the atlantic
  - efficient tractable algorithms for UQ, no brute force
- Wide range of applications for UQ ranging from traditional mechanics (heat transfer, CFD) to nuclear energy
  - Typical models are large scale spatio-temporal partial differential equations
- UQ Framework current status based on
  - non-intrusive (blackbox,semi-intrusive) methods
  - single point model execution

Some Conclusions : UQ & HPC III

- In terms of HPC
  - Trying to get each individual simulation faster
  - Queueing/scheduling of jobs + concurrent MPI jobs, local threading, system calls
  - Machine range from linux workstations, cluster to very large scale computer
  - No real parallel effort in UQ software (no need too)
- Intrusive UQ(e.g. Trilinos/Stokhos) :
  - Library to solve stochastic PDEs using intrusive Galerkin type methods
  - based on Trilinos which brings state of the art parallel computing framework (MPI)
  - Until now very few developments to understand what would be gained with this type of approach
- Hybrid computing steers some interest in UQ. In HPC many projects develop strategies to benefit form hybrid architectures
- Towards petascale and exascale
  - May need to rethink the way UQ is done (difficult to justify these machines with "embarrassingly parallel jobs"
  - Towards intrusive methods
- Software
  - Risk : maintainability, need for abstract evolutive frameworks that can handle (i) new methodologies at a reduced cost, (ii) new architectures