Understanding and Improving the Trust in Results of Numerical Simulations and Scientific Data Analytics

Franck Cappello, **Rinku Gupta,** Sheng Di, Emil Constantinescu, Thomas Peterka and Stefan Wild

Argonne National Laboratory

The 10th Workshop on Resiliency in High Performance Computing (Resilience) in Clusters, Clouds, and Grids
August 2017

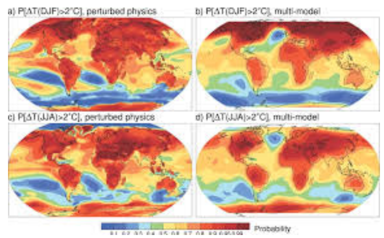
Agenda

- **Motivation and defining trust**
- Building trust and why existing techniques only help partially
- What strategies?
- This is just a beginning

Scientific Computing
from Petascale to Exascale

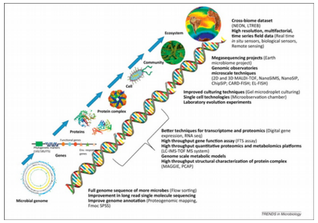
Many scientific and engineering domains rely on numerical simulations

Climate

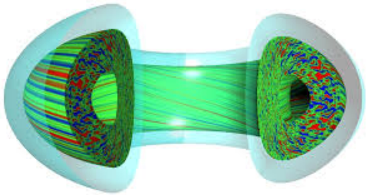


Biology

Modeling at multiple scales: atomic, genomic , cellular, ecosystems



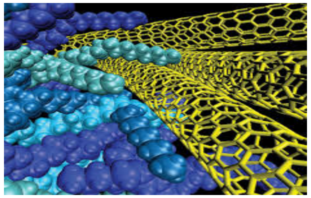
Fusion Energy



Brain simulation

Cosmology

Material science



Scientists need:

- More complex physical models to account for more aspects of the physical phenomena being modeled (multi-physics)

- Increases in the resolution of the system variables to improve simulation accuracy

Current Extreme Scale Systems

Today's Leadership Systems: Two Paths

Titan (Hybrid Multi-core)

- 27 PF peak, hybrid CPU/GPU
- 18,688 Compute nodes – 1,452 GF
  - 8/16 core AMD Opteron-141 GF
    - 32 GB DDR3 memory
  - 14 SM NVIDIA Kepler GPU-1,311 GF
    - 6 GB GDDR5 memory
  - Explicit PCIe link between processors
- Cray Gemini 3-D Torus Interconnect
- 32 PB / 1 TB/s Lustre file system

Mira (Many Core) BG/Q

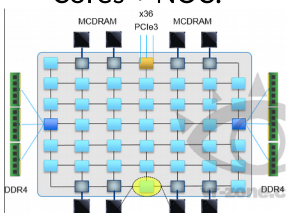
- 10 PF peak, many-core CPU
- 49,152 Compute nodes
  - 16 core IBM PowerPC – 205 GF
    - 16 GB DDR3 memory
- IBM BlueGene 5-D Torus Interconnect
- 24 PB / 240 GB/s GPFS file system

NSF BlueWaters (1.5 PB of memory) is close to Titan in terms of architecture

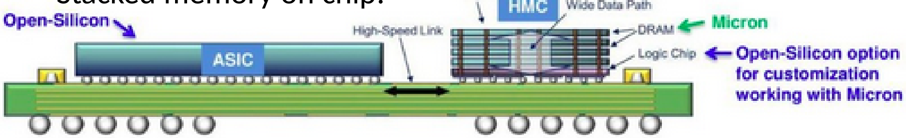
Future systems: even more complex

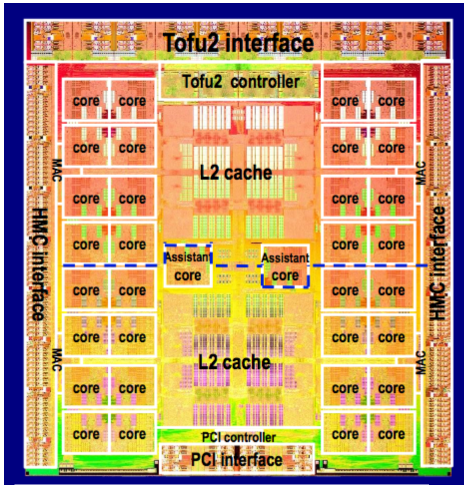
- More cores (some times: fat/assistant cores)
- Deeper memory hierarchy
- Non volatile memory
- Fast I/O interface (network) on the chip
- Relatively higher cost for communications and I/O (lower bandwidth to file system)
- **Higher disruption (fault, error, failure) rate**

Cores + NOC:

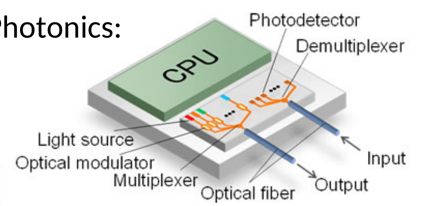


Stacked memory on chip:





Silicon Photonics:



Failures in the field and their techniques

- Fail stop errors

  Any hardware/software component that stops executing actions it is supposed to do.

- Correctable Intermittent/transient errors

  Component detected as failed that comes back by itself: Memory with ECC, Network with CRC.

- Uncorrectable transient errors

  Error is reported to the OS that Either kills the application OR inform the application (call back)

- Silent data corruptions

  Any corruption undetected by hardware (radiations, bugs, Attack)

Rollback recovery
Checkpoint/restart

Outcomes:

- **Execution crashes**
- **Corrupted results**

Open problem: some research results but not use in production

Relate to the **scientific data integrity** problem

Relate to **result trustworthiness**

What is Trust (briefly)?

- Trust research aims to improve the confidence (with some quantification if possible) on the results (of numerical simulations and data analytics)

- Trust focuses on the product of the execution
  → direct connection to the applications and users
  → defines required execution properties based on the result expectations

- What could impair trust on scientific results: **corruptions**

- Trust is not a simple problem!
  – Involves techniques for Validation and Verification, Uncertainty quantification, etc.
  – Caused by Errors + Bugs + Attacks
  – Involves users and their expectations!

Why Trust research is important?

- There are many examples of execution producing bad results due to some form of result corruption.
- Let's start with an example in the space industry:
  - Ariane 5 launch (501), 4th of June 1996 (just 20 years back)

Explosion of Ariane 5
Loss of more than US$370 million
+population evacuation
+ loss of scientific results

Why Trust research is important?

- Other examples with catastrophic consequences:
  - See http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.html for list of num. Errors
  - See https://en.wikipedia.org/wiki/List_of_software_bugs for list of bugs
  - See http://www5.in.tum.de/~huckle/bugse.html for an even longer list of bugs.

- Consequences can be significant in the context of scientific simulations and data analytics
  - Wrong decisions may have been taken
  - Large no. of executions may be corrupted before discovery
  - Post-mortem verification requires heavy checking
  - Leads also to significant productivity losses.

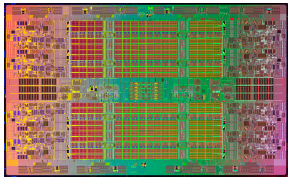
The sinking of the Sleipner A offshore platform (inaccurate finite element approximation)

- **In numerical analysis and scientific data analytics, there is a lack of trust metrics that can be used to quantitatively compute and express the trustworthiness of the execution results**

Corruption classification

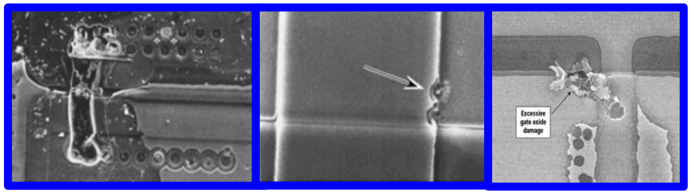
- **Not all corruptions are equal!**
  - **Some corruptions are expected, controlled and accepted** (discritization, round-off errors truncation) → intrinsic to methods/algorithms. V&V quantify them
  - **And then there are unexpected corruptions that stay undetected.**
- **A harmful corruption** is manifested as a silent alteration of one of more data elements.
- **Nonsystematic corruptions** affect data in a unique way; probability of repetition of the exact same corruption in another execution --> very low.
  - **Sources**: radiations (cosmic ray, alpha particles from package decay), bugs in some paths of nondeterministic executions, attacks targeting executions individually and other potential sources.
- **Systematic corruptions** affect data the same way at each execution. Executions do not need to be identical to produce the same corruptions.
  - Sources: (1) bugs or defects (hardware or software) that are exercised the same way by executions and (2) attacks that will consistently affect executions the same way.

Sources of corruption



Hardware Issues (usually called SDCs)

•Hard error: permanent damage to one or more elements of a device or circuit (e.g., gate oxide rupture, etc.).



•Soft error (transient errors): An erroneous output signal from a latch or memory cell that can be corrected by performing one or more normal functions of the device containing the latch or memory cell:

Cause: Alpha particles from package decay, Cosmic rays creating energetic neutrons

Soft errors can occur on transmission lines, in digital logic, processor pipeline, etc.

Bugs

Hardware

1994: Bug of the FDIV instruction of the Pentium P5 processor.

2014: Opteron's random jump/branch into code.

Libraries

2014: cuBLAS DGEMM (CUDA 5.5) on Blue Waters' Kepler GPUs: silent error: results of the cuBLAS DGEMM matrix-matrix multiplication are incorrect

Compilers

2012: IntelFortran: bugs affecting numerical results (in particular, in OpenMP vectorization and): "Loop vectorization causes incorrect results".

Frameworks

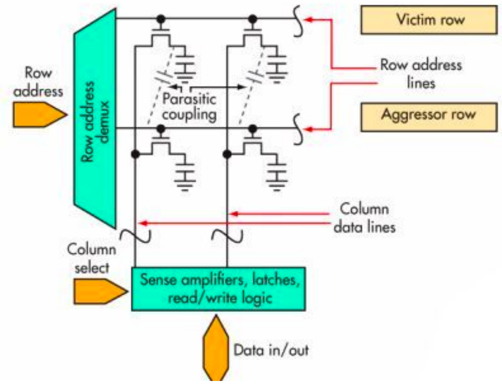
2008: Bug in Nmag micromagnetic simulation package leading to: "Calculation of energy (exchange , demag, Zeeman, total) energies, had wrong result"

•Detection time and notification time is a major issue:

Attacks (example)

2014 (ISCA) a group of Carnegy Mellon and Intel show how to flip bits without accessing the victim DRAM row.

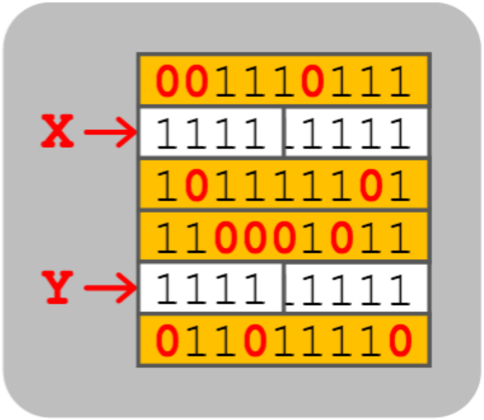
Observation: Toggling a row accelerates charge leakage in adjacent rows, because of row-to-row coupling



Technique:

- DRAM is refreshed every 64ms
- Accelerate charge leakage by writing on the same data at high frequency
- Flush caches to hit DRAM
- Victim rows will be corrupted before the next refresh

```
loop:
  mov (X), %eax
  mov (Y), %ebx
  clflush (X)
  clflush (Y)
  mfence
  jmp loop
```



All modules manufactured in the past two years (2012 and 2013) were vulnerable

As many as 4 errors per cache line: simple ECC (SECDED) cannot prevent all errors

2015 Googleprojectzero published a Linux attack based on row hammer

http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

Lessons

- Hardware issues (defect, radiation induced bit flips) happen (usual SDCs)
- Bugs are reported everywhere in the stack from the hardware to the users
- Software upgrades often introduce new functionalities that bring new sets of bugs and potential corruptions.
- Attacks exploiting technology weaknesses

- We run simulations and data analytics over very complicated, evolving and fragile stacks

- What can we do to improve the trust in scientific computing results?

Agenda

- Motivation and defining trust
- **Building trust and why existing techniques only help partially**
- What strategies?
- This is just a beginning

Building trust in application results

- Trust in numerical analysis and data analytics related to two notions
  1. **Correctness of computation**
  2. **Integrity of execution stack**
- Neither can be proven formally, resulting in users developing a process to build trust in their execution results
  – Build trust in smallest scale, simplest problem → scale to larger complexity and size
  – Any odd error is scrutinized and assumed to be an error until demonstrated otherwise

Expected result accuracy

- "Expected result accuracy" helps evaluate correctness of computation
  - Defined "*if the corruption of data* ***does not result in any measurable changes to any meaningful statistic*** *of an application between two executions (corruption-containing vs. not), it means users expectation of accuracy has been satisfied* "
  - Application dependent (some applications are sensitive to details of calculation; some follow trajectory)
  - Typical expected accuracies: 10^{-6} for HACC and 10^{-8} for Nek5K

Research: focus on detecting corruptions that make the end results diverge from the expected user accuracy

V&V and why they help only partially

- **Validation** compares the output of a simulation with experimental data. Determines faithfulness of mathematical/computational models to the read world
- **Verification** checks that the simulation code respects its specification or models (solution verification, code verification, unit and regression testing,…)
- Validation and verification attempt, though incompletely, that the process/code is a truthful implementation of the algorithms themselves

Limitations:

- Formal **validation and verification** presuppose a correct reference solution.
- Formal methods are limited to simpler or smaller subsystems than the apps.
- No solution for complex simulations performed for DOE.

Why replication and ABFT helps only partially

Harmful **nonsystematic** corruptions:

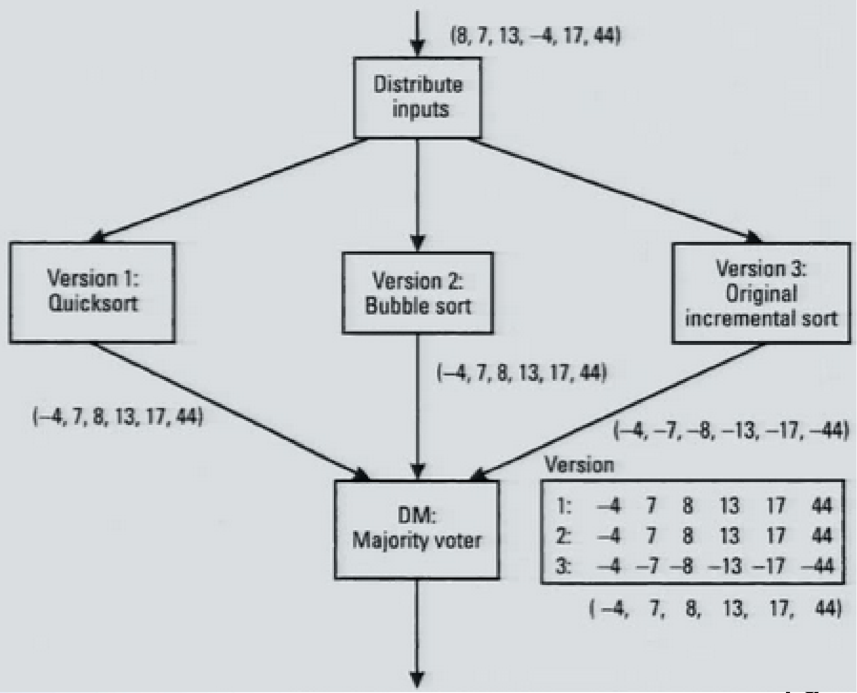
- **Replication** works but is too expensive to be applied on all executions,
- **ABFT** covers only the data protected by the ABFT scheme: other application data are not protected.

Harmful **systematic** corruptions:

- **Replication** does not work because it detects corruptions by comparing identical (or comparable) executions.
- **ABFT** may not detect corruptions affecting the ABFT calculation itself. ABFT is also not a solution for attacks because a sophisticated attack could target data sets not protected by ABFT or alter the ABFT calculation itself.

Why Multi-version does not help

- **N-version programming** was proposed almost three decades ago.
- It was proposed to detect bugs (systematic corruptions).
- Similar to the notion of “alternates” in “recovery blocks”,
- Principle: **compare the results of the executions of multiple different code versions responding to the same specification.**
- The higher the diversity of the versions (from hardware to application), the higher is the chance of detecting corruptions.
- This approach **does not seem applicable in our domain** because of the cost of developing multiple versions of all levels of the stacks, from the hardware to the application.
- Moreover, it has been demonstrated experimentally that **different versions may suffer the same bugs** (and lead to the same corruptions).



Agenda

- Motivation and defining trust
- Building trust and why existing techniques only help partially
- **What strategies?**
- This is just a beginning

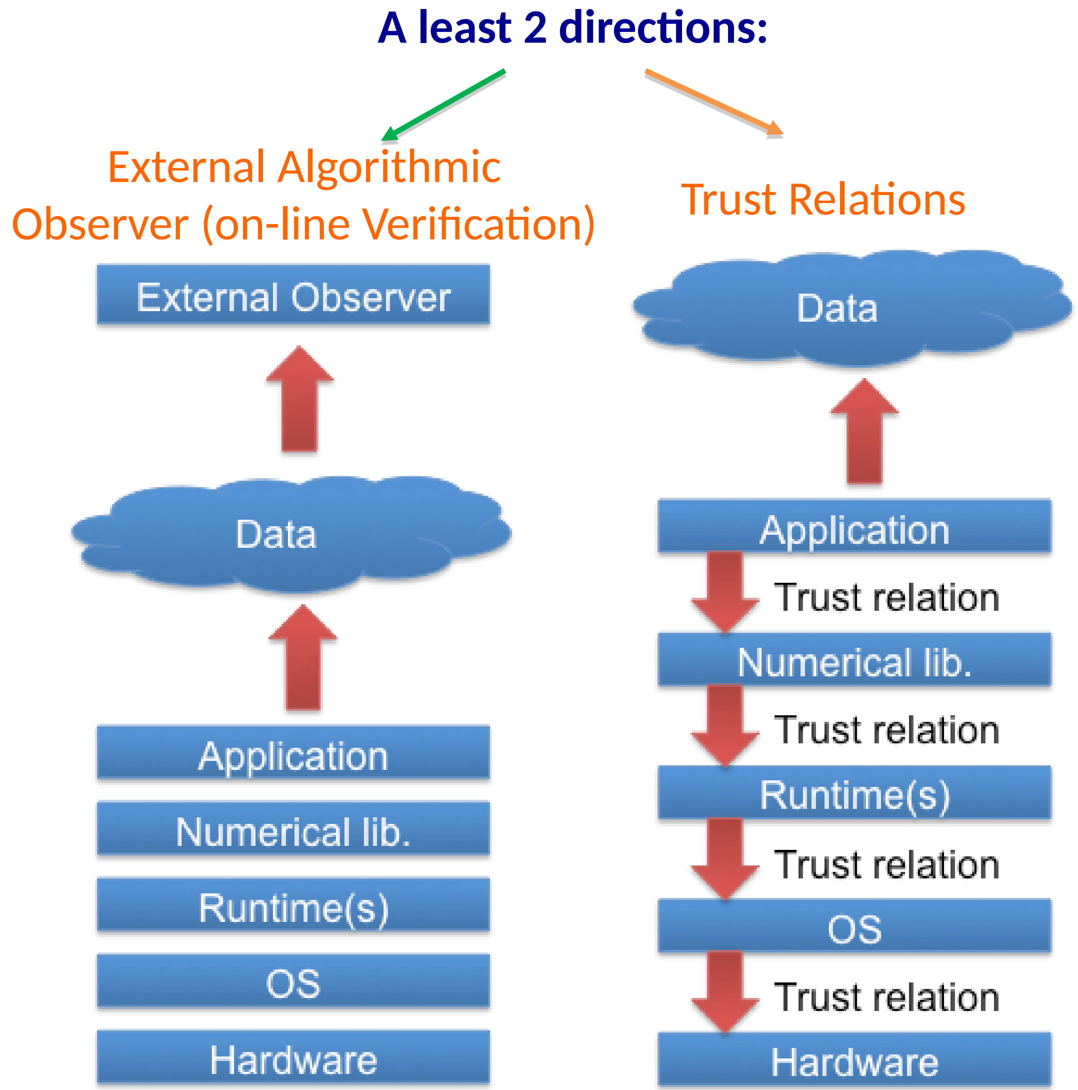
2 complementary directions

The Trust problem:

•spans over all layers between hardware and users.

•Is related to many aspects of numerical simulation and data analytics (modeling, initial conditions, numerical accuracy, parametric settings, etc.).

Only a holistic approach has a chance of succeeding.



External Algorithmic Observer Concept

Main idea follows Lui Sha's proposal for "**using simplicity to control complexity**" architecture for critical systems. **Separate critical requirements from desirable properties.**

Example

- One way to do that, is to keep device states in an envelope established by simple and reliable controller
- Example
  - Boeing 777 flight control system uses triple-triple redundancy (3 parallel control channels, each channel has 3 different processors i.e. Intel, Motorola, AMD)
  - Application-software level has 2 controllers, new sophisticated primary controller, and old 747-based secondary controller

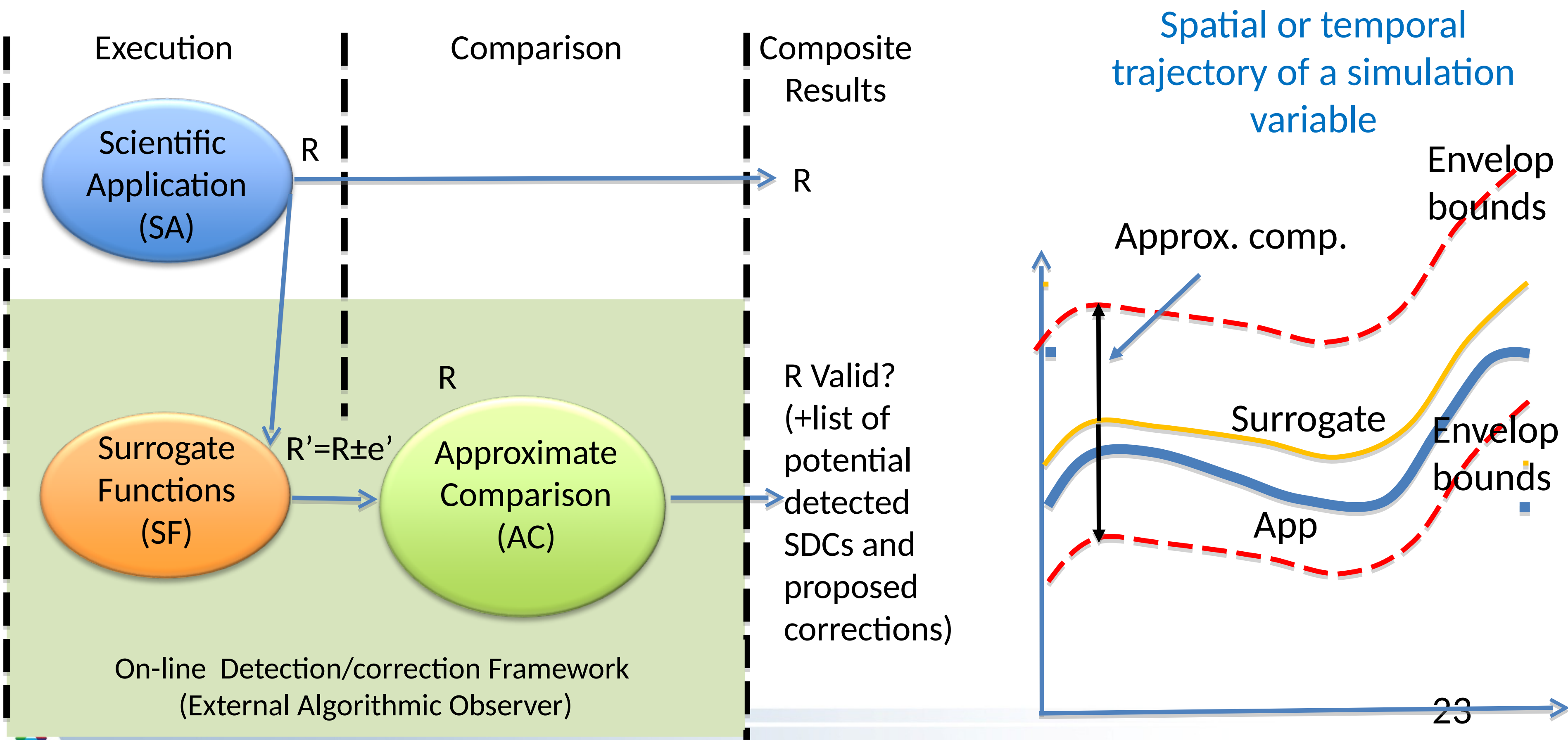
Lui Sha (UIUC), Using Simplicity to Control Complexity -– EEE Software, Jan 2001

External Algorithmic Observer Principles

External Algorithmic Observer for scientific applications:

•**Executes a surrogate function that models** the data transformation performed by the application.

•**Approximately compares** the result of the application and the surrogate function



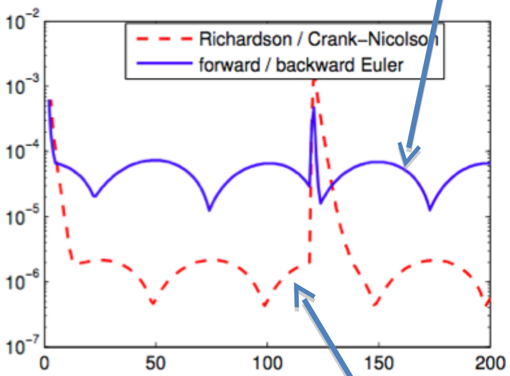
External Algorithmic Observer : Current Research

- Very few published research results (3 known groups)

- Two models so far (all for time stepping simulations):
  - **Auxiliary numerical method**:
    - Benson, Schmit, Schreiber 2014
    - Guhur, Zhang, Peterka, Constantinescu, Cappello
  - **Prediction based method**:
    - Gomez, Di, Berrocal, Cappello, 2014, 2015, 2016
    - Sharma, Bronevetski, Gopalakrishnan, 2015
    - Di, Cappello 2016
    - Subasi, Di, Gomez, Balaprakash, Unsal, Cristal, Labarta, Cappello 2016

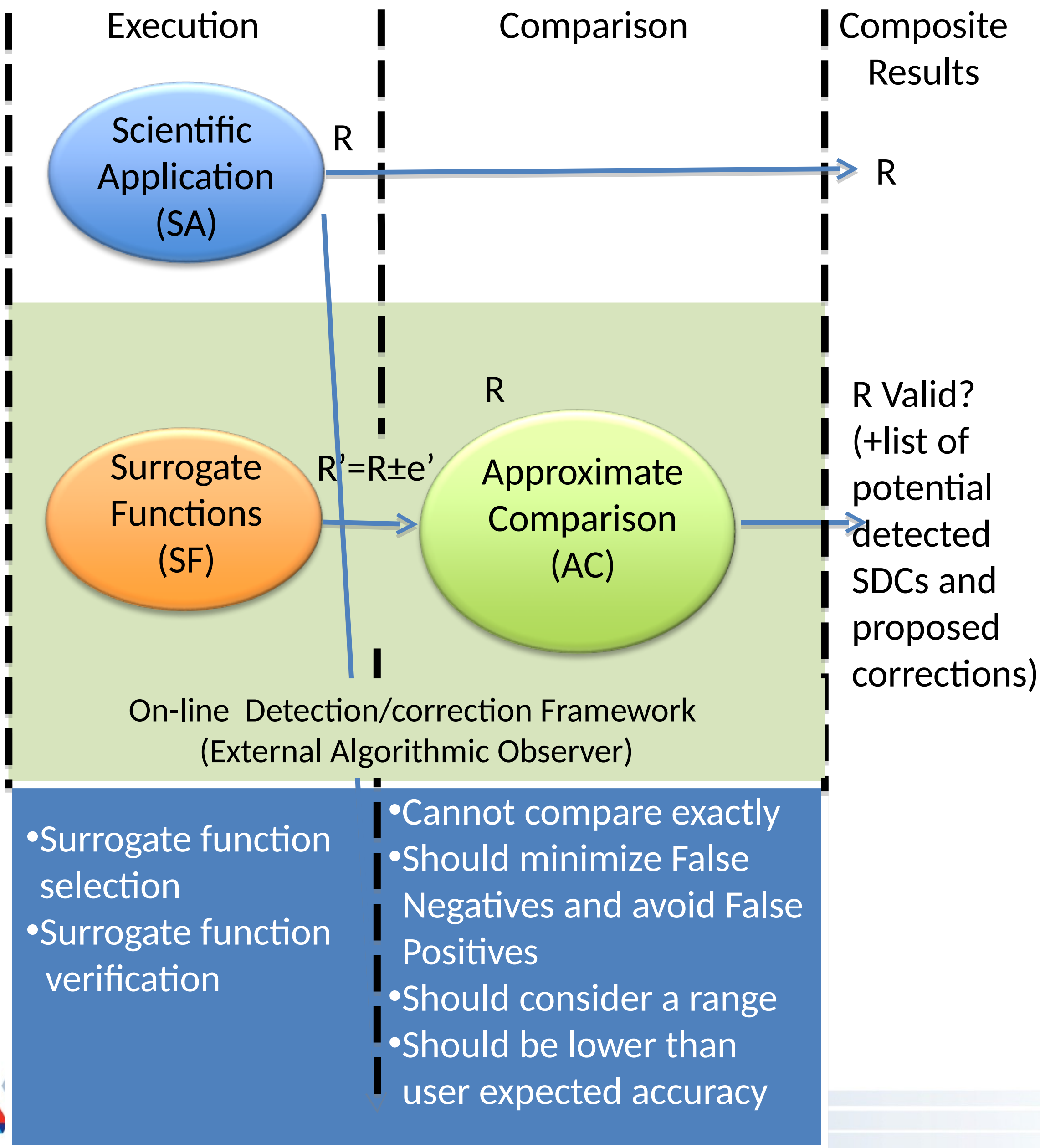
Please
Read cited papers
for details!

Lower order method



Higher order method

External Algorithmic Observer : Research directions



Observations

1) SF **cannot replace SA**. SF's predictions are valid only from one step to the next one.

2) Low-complexity SF models implement **trade-offs between complexity, accuracy, and other properties**:

3) Important advantage: **model is easier to verify and to protect than the** application. → Amenable to formal verification, multi-version programming and execution on a secure processor (FPGA for example).

Coverage of the Algorithmic Observer

Why does it cover (partially) non-systematic corruptions:

- Very unlikely to have the same non-systematic corruptions twice in the simulation and in the model

Why does it cover (partially) systematic corruptions:

- The simulation and model do not perform the same computations
- However data is very close.
- Low probability that a same operation (FPADD, FPMUL, etc.) is executed with close data in both the simulation and model→ Recommendation: execute the model in a different hardware (CPU+GPU)

→ More study is needed on the coverage

Trust Relations

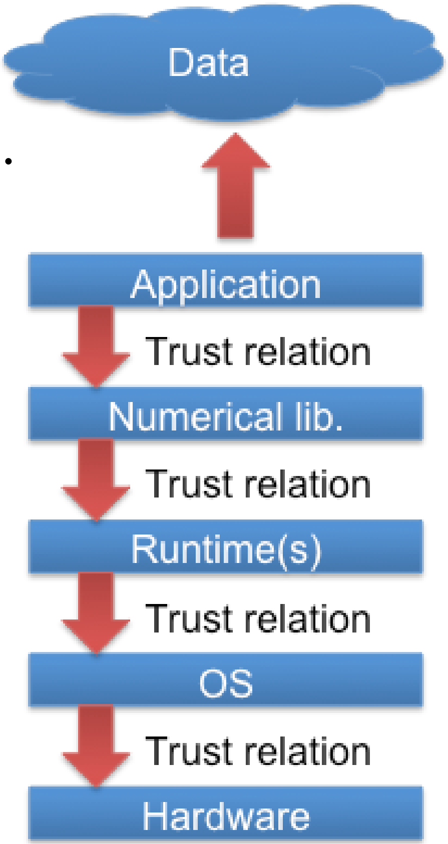
More mature: a large body of research in computer science

DOE report on Cybersecurity for Scientific Computing Integrity [pic on last slide] covers issues and approaches.

“Object” → any software of hardware that needs to be trusted.

→Trust relation supposes at least:

- a way to **certify** that **each used object** is actually the object it is supposed to be,
- a method to **evaluate a level of trust** for each object involved in the execution (reputation for example)
- a **metric of the level of trust**, and
- a way to **protect the trust level acquired** by an object



Trust Relations (Potential Research)

Certification and protection of trust level:

→**Trust Computing Group** produced Trusted Platform Module (TPM) specification

→Specifies Embedded crypto capability for user, apps., machine authentication

- More than 500 million PCs have shipped with TPM.
- Vulnerable to sophisticated attacks + TPM circuits showed vulnerability

Trust evaluation:

→Trust level could rely on **verification and validation** of that object by a combination of formal verification when applicable and empirical methods.

→In principle, **external observer approach** can be applied for each object.

Trust Metrics:

→**Not a new problem** in security and networking domains (solutions)

→**Metrics with multiple dimensions**: time since first trusted, time since last verification, number of independent verifications, , etc

→Trust metric should compute trust level for entire execution (function based on individual object trust – combinations of which user can explore based on needs)

All these precautions will not avoid corruptions from a highly trusted object. More research needed!

http://www.trustedcomputinggroup.org/

Comparing the 2 approaches

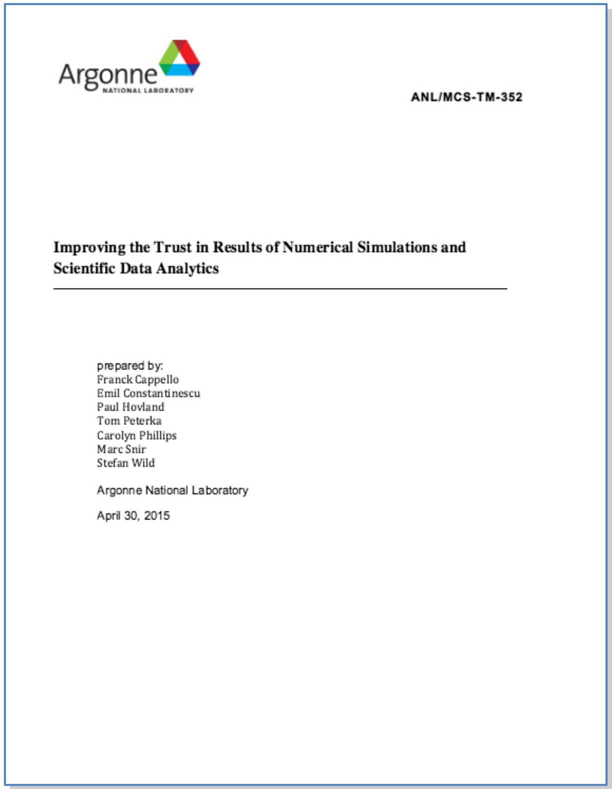
	External Observer	Trust Relations
Detection Approach	Simulation and observer are checking each other	Checking object results
Detection Assumptions	**External observer is correct** (should be verified, validated)	All **verifications and reputation calculations are correct**
Detection Latency	**Short** (depends on sampling rate, typically 1 application iteration)	**Long** (actual detection could be long: months)
Timeliness of Notification after Detection	**Short** (from one iteration to the next)	**Short** (immediate upper layer)
Time to build trust	**Low** (trust depends on accuracy of results not on components)	**High** (hard and soft components need to acquire trust level)
Targeted Level of Trust	**User-expected accuracy**	**Machine precision** (modulo round-off errors)
Development Time and Cost	**Low** (requires only to develop the observer)	**High** (affects all layers of the stack)
Tolerance	**High** (corruptions of the application data lower than user-expected accuracy are tolerated)	**Low** (any corruption at object level is suspicious since the consequence on application data is unknown)

Conclusion

- Trust in results of numerical simulation and data analytics is serious and insufficiently recognized problem in our community
- Lack of no trust metrics : Different domains have different definitions of trust : example: ecommerce has “reputation” as metric which wont work for us!
- Lack of research and results in this domain.
- Two directions (identified so far, more probably exist): (1) Algorithmic external observer and (2) Trust relation (much more mature in other domains)

→ it’s a fascinating and pretty open research problem!

Questions?



Argonne NATIONAL LABORATORY

ANL/MCS-TM-352

Improving the Trust in Results of Numerical Simulations and Scientific Data Analytics

prepared by:
Franck Cappello
Emil Constantinescu
Paul Hovland
Tom Peterka
Carolyn Phillips
Marc Snir
Stefan Wild

Argonne National Laboratory

April 30, 2015