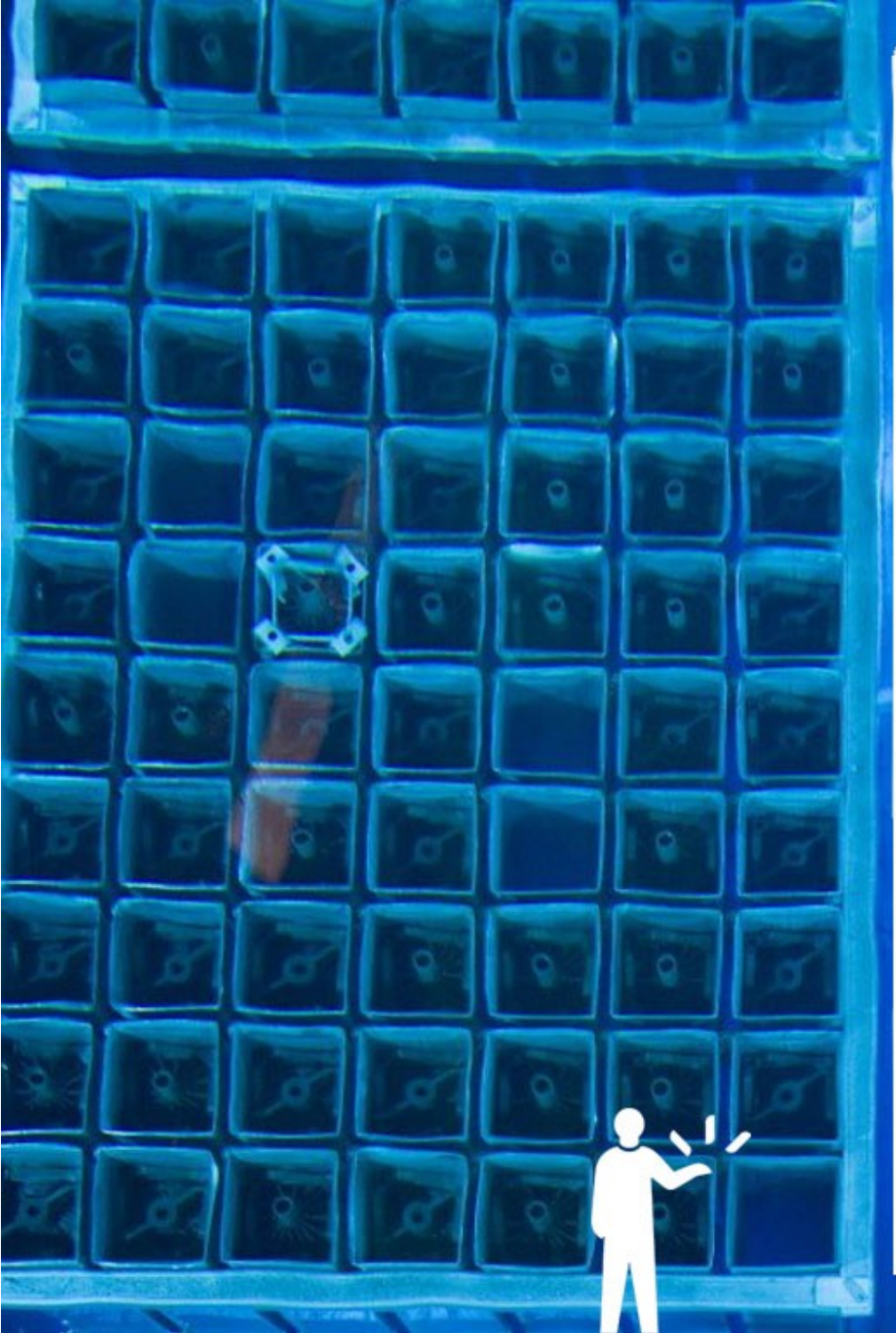


PROMISE: floating-point precision tuning with stochastic arithmetic

17th International Symposium on Scientific Computing, Computer Arithmetics and Verified Numerics
Uppsala, Sweden
26-29 September 2016

LIP6, Université Pierre et Marie Curie
Sorbonne Universités, Paris, France
EDF R&D, Palaiseau, France

Stef Graillat, Fabienne Jézéquel, Romain Picot,
François Févotte & Bruno Lathuilière

CHANGER L'ENERGIE ENSEMBLE

Introduction

- Development of computational resources
- Intensive use of floating-point arithmetic

- Use of the highest available precision
- Mixed precision approach [Baboulin & al, 2009]

Introduction

- Development of computational resources
- Intensive use of floating-point arithmetic

- Use of the highest available precision
- Mixed precision approach `[Baboulin & al, 2009]`

How to automatically tune floating-point precision?

- Exhaustive test has a complexity in $O(2^n)$

- How to validate a configuration?

State of the art

- CRAFT HPC [Lam & al., 2013]
  - binary modifications on the operations
- Precimonious [Rubio-Gonzàlez & al., 2013]
  - source modification with LLVM

State of the art

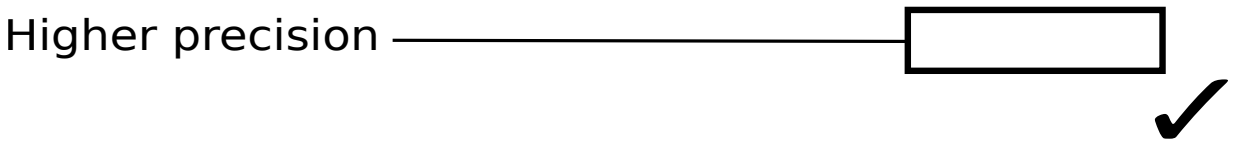
- CRAFT HPC [Lam & al., 2013]
  - binary modifications on the operations
- Precimonious [Rubio-Gonzàlez & al., 2013]
  - source modification with LLVM

Both relies on a comparison in the highest precision without a validation

1. Searching for a configuration

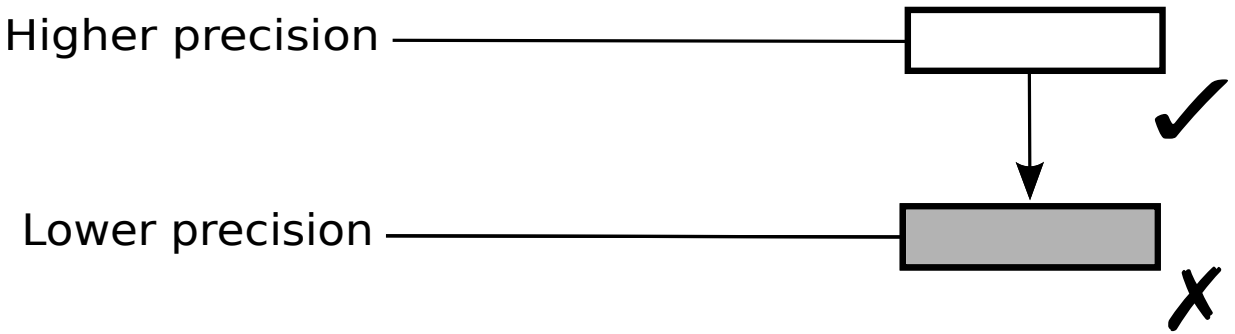
Searching for a configuration

Method based on Delta Debugging algorithm [Zeller, 2009]



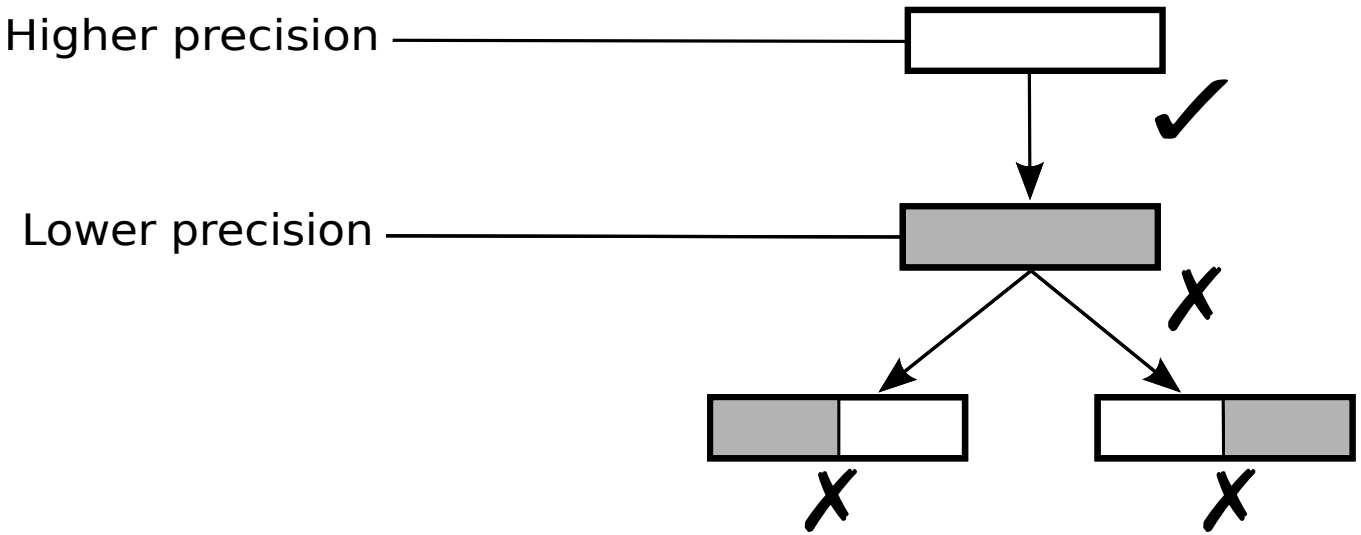
Searching for a configuration

Method based on Delta Debugging algorithm [Zeller, 2009]



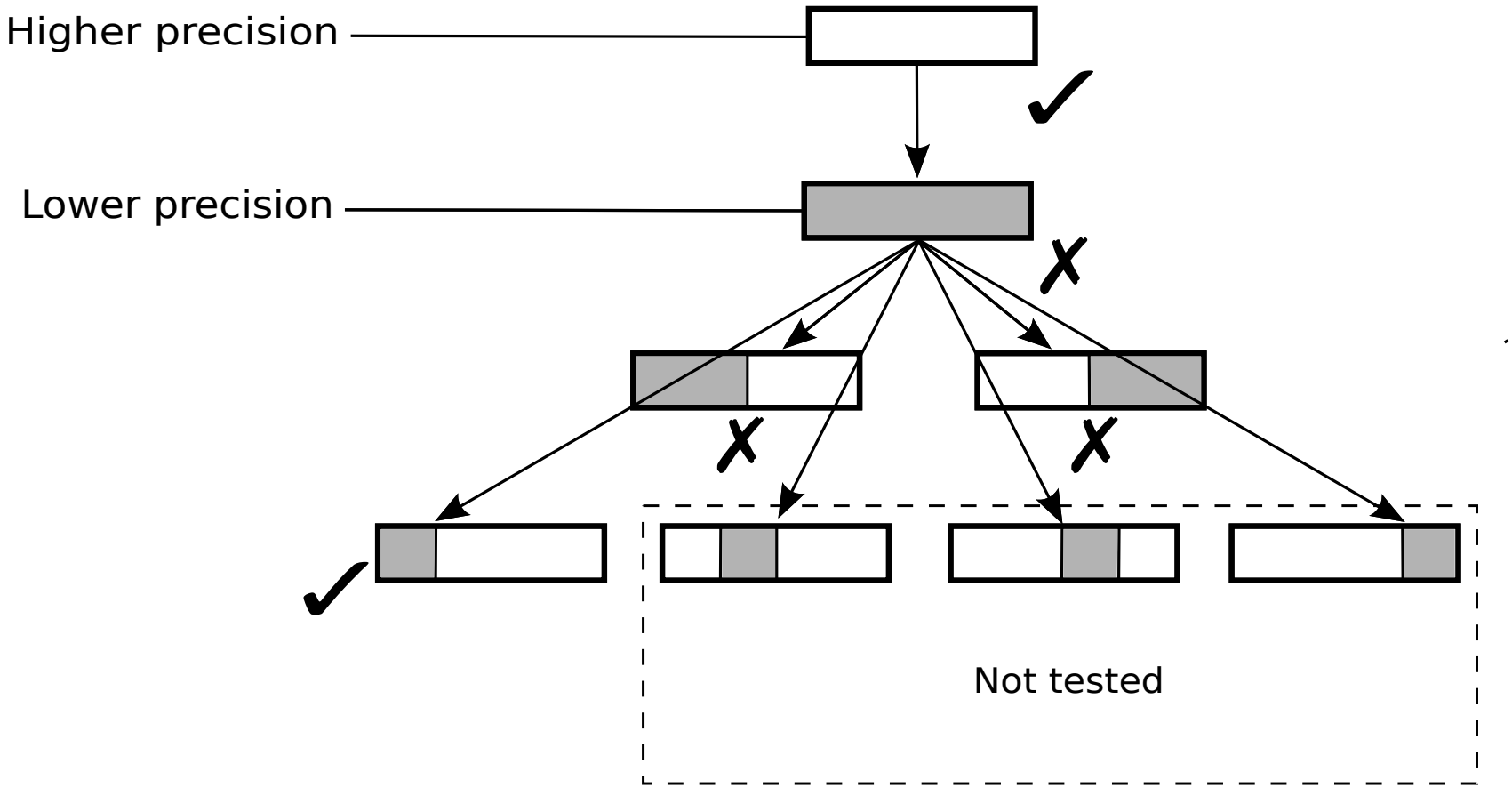
Searching for a configuration

Method based on Delta Debugging algorithm [Zeller, 2009]



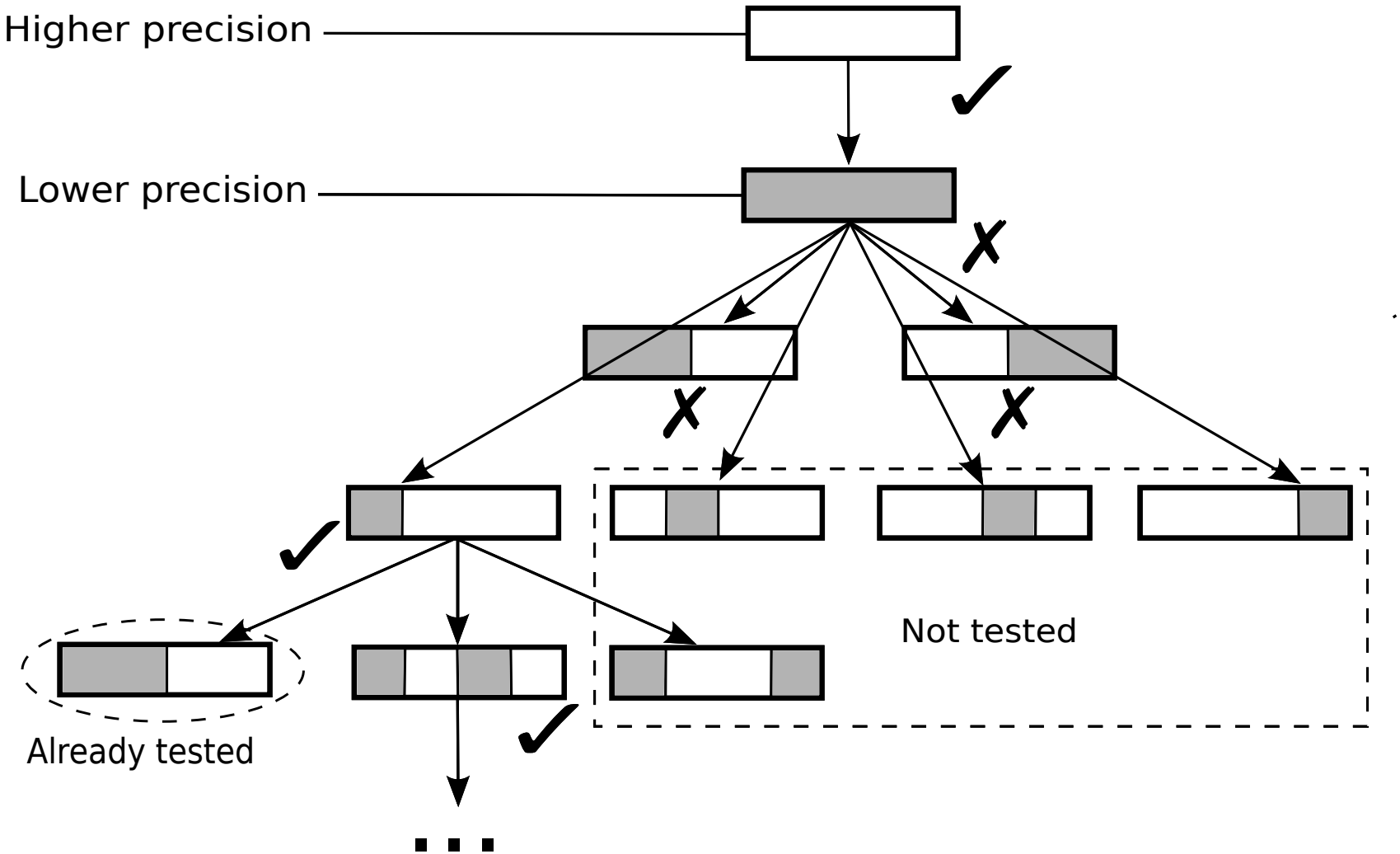
Searching for a configuration

Method based on Delta Debugging algorithm [Zeller, 2009]



Searching for a configuration

Method based on Delta Debugging algorithm [Zeller, 2009]



Searching for a configuration

- We will not have the *best* configuration.
- But the mean complexity is $O(n \log(n))$ and in the worst case $O(n^2)$

Searching for a configuration

- We will not have the *best* configuration.
- But the mean complexity is $O(n \log(n))$ and in the worst case $O(n^2)$

Efficient way of finding a local maximum configuration

2. Validate a configuration

1. Searching for a configuration

2. Validate a configuration
- Discrete Stochastic Arithmetic and CADNA
- Several possibilities to validate a configuration

3. PROMISE

4. Experimental results

5. Conclusion and Perspectives

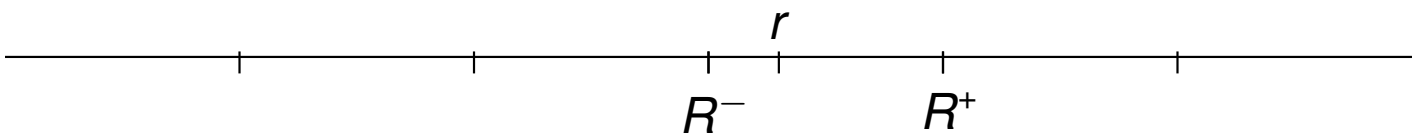
Round-off error analysis

Several approaches

- Inverse analysis
  based on the “ Wilkinson principle”: the computed solution is assumed to be the exact solution of a nearby problem
  - provides error bounds for the computed results
- Interval arithmetic
  The result of an operation between two intervals contains all values that can be obtained by performing this operation on elements from each interval.
  - guaranteed bounds for each computed result
  - the error may be overestimated
  - specific algorithms
- Probabilistic approach
  - uses a random rounding mode
  - estimates the number of exact significant digits of any computed result

CESTAC method

If the exact result r of an arithmetic operation is not a floating-point number, it is approximated by a floating-point number R^- or R^+.



The random rounding mode

Approximation of r by R^- or R^+ with the probability 1/2

The CESTAC method [Vignes & La Porte, 1974]

The same code is run several times with the random rounding mode.
Then different results are obtained.

Briefly, the part that is common to all the different results is assumed to be reliable and the part that is different in the results is affected by round-off errors.

Implementation of the CESTAC method

The implementation of the CESTAC method in a code providing a result R consists in:

- performing N times this code with the random rounding mode to obtain N samples R_i of R,
- choosing as the computed result the mean value $\overline{R}$ of R_i, $i = 1, \ldots, N$,
- estimating the number of exact significant decimal digits of $\overline{R}$ with

$$C_{\overline{R}} = \log_{10}\left(\frac{\sqrt{N}\,|\overline{R}|}{\sigma\tau_\beta}\right)$$

where

$$\overline{R} = \frac{1}{N}\sum_{i=1}^{N} R_i \quad \text{and} \quad \sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(R_i - \overline{R}\right)^2.$$

τ_β is the value of Student's distribution for $N-1$ degrees of freedom and a probability level $1-\beta$.

In pratice, $N = 3$ *and* $1-\beta = 95\%$.

The concept of computed zero

Definition [Vignes, 1986]

Using the CESTAC method, a result R is a computed zero, denoted by @.0, if

$$\forall i, R_i = 0 \quad \text{or} \quad C_{\overline{R}} \leq 0.$$

R is a computed result which, because of round-off errors, cannot be distinguished from 0.

The stochastic definitions [Vignes, 1993]

Definition

Let X and Y be two results computed using the CESTAC method (N-sample), X is stochastically equal to Y, noted $X \stackrel{s}{=} Y$, if and only if

$$X - Y = @.0.$$

Definition

Let X and Y be two results computed using the CESTAC method (N-sample).

- X is stochastically strictly greater than Y, noted $X \stackrel{s}{>} Y$, if and only if

$$\overline{X} > \overline{Y} \quad \text{and} \quad X \stackrel{s}{\neq} Y$$

- X is stochastically greater than or equal to Y, noted $X \stackrel{s}{\geq} Y$, if and only if

$$\overline{X} \geq \overline{Y} \quad \text{or} \quad X \stackrel{s}{=} Y$$

Discrete Stochastic Arithmetic (DSA)

The discrete Stochastic Arithmetic is defined as the joint use of:

- CESTAC method
- computed zero
- stochastic relation definitions

CADNA

http://cadna.lip6.fr

The CADNA library implements Discrete Stochastic Arithmetic.

CADNA allows one to *estimate the accuracy* of a result of a scientific program written in Fortran or in C++.

CADNA

http://cadna.lip6.fr

The CADNA library implements Discrete Stochastic Arithmetic.

CADNA allows one to *estimate the accuracy* of a result of a scientific program written in Fortran or in C++.

CADNA provides new numerical types, the *stochastic types*, which consist of:

- 3 floating-point variables
- an integer variable to store the accuracy

All operators and mathematical functions are redefined for these types.
$\Rightarrow$ CADNA requires only *a few modifications in user programs*.

Several possibilities to validate a configuration

- Comparison with an execution in the highest precision
- Validation of every execution using CADNA
- Validation of a reference using CADNA and comparison to this reference

3. PROMISE

1. Searching for a configuration
2. Validate a configuration
3. **PROMISE**
4. Experimental results
5. Conclusion and Perspectives

PROMISE

We developed two versions:

- `Full stochastic`: Validation of every execution
  In this version, we validate every configuration with CADNA.

  Each configuration result is evaluated based on:
  - The number of exact significant digits estimated by CADNA
  - The number of digits in common with the result obtained in double precision using CADNA

- `Stochastic reference`: Validation of a reference
  Because of the cost of CADNA, $\approx$ 10 times slower, we reduced its use to one execution.

  - The reference is evaluated based on the number of exact significant digits estimated by CADNA.
  - Each configuration result is compared to the reference result.

4. Experimental results

1. Searching for a configuration
2. Validate a configuration
3. PROMISE
4. **Experimental results**
   - Benchmarks
   - MICADO: the neutron transport equations solver
5. Conclusion and Perspectives

Benchmarks

- **Short programs:**
  - arclength computation
  - rectangle method for the computation of integrals
  - Babylonian method for square root
  - matrix multiplication
- **GNU Scientific Library:**
  - Fast Fourier Transform
  - sum of Taylor series terms
  - polynomial evaluation/solver
- **SNU NPB Suite:**
  - Conjugate Gradient method
  - a Scalar Penta-diagonal solver

Requested accuracy: 4, 6, 8 and 10 digits

Benchmarks results

Stochastic reference

Programme	# Digits	# comp - # exec	# double - # float	Time (mm:ss)	Result
arclength	exact				5.79577632241285
	10 8	21-21	8-1	0:13	**5.7957763224130**3
	6	26-26	7-2	0:15	**5.79577**686259398
	4	16-16	2-7	0:09	**5.796**19547341572
rectangle	exact				0.100000000000000
	10 8	15-15	4-3	0:06	**0.10000000000000**2
	6	16-16	3-4	0:06	**0.10000000**1490116
	4	3-3	0-7	0:01	**0.10000**3123283386
squareRoot	exact				1.41421356237309
	10 8	21-21	6-2	0:07	**1.41421356237309**
	6 4	3-3	0-8	0:01	**1.4142135**3816986

Comparison with Precimonious

- Two different ends:
  - PROMISE: Maximize the number of single precision types
  - Precimonious: Having the best speed-up
- Two different tests
- Without time measurement, Precimonious and PROMISE have similar results

MICADO: the neutron transport equations solver

- industrial code used to compute the neutron transport equation
- 11,000 code lines in C++
- iterative solver

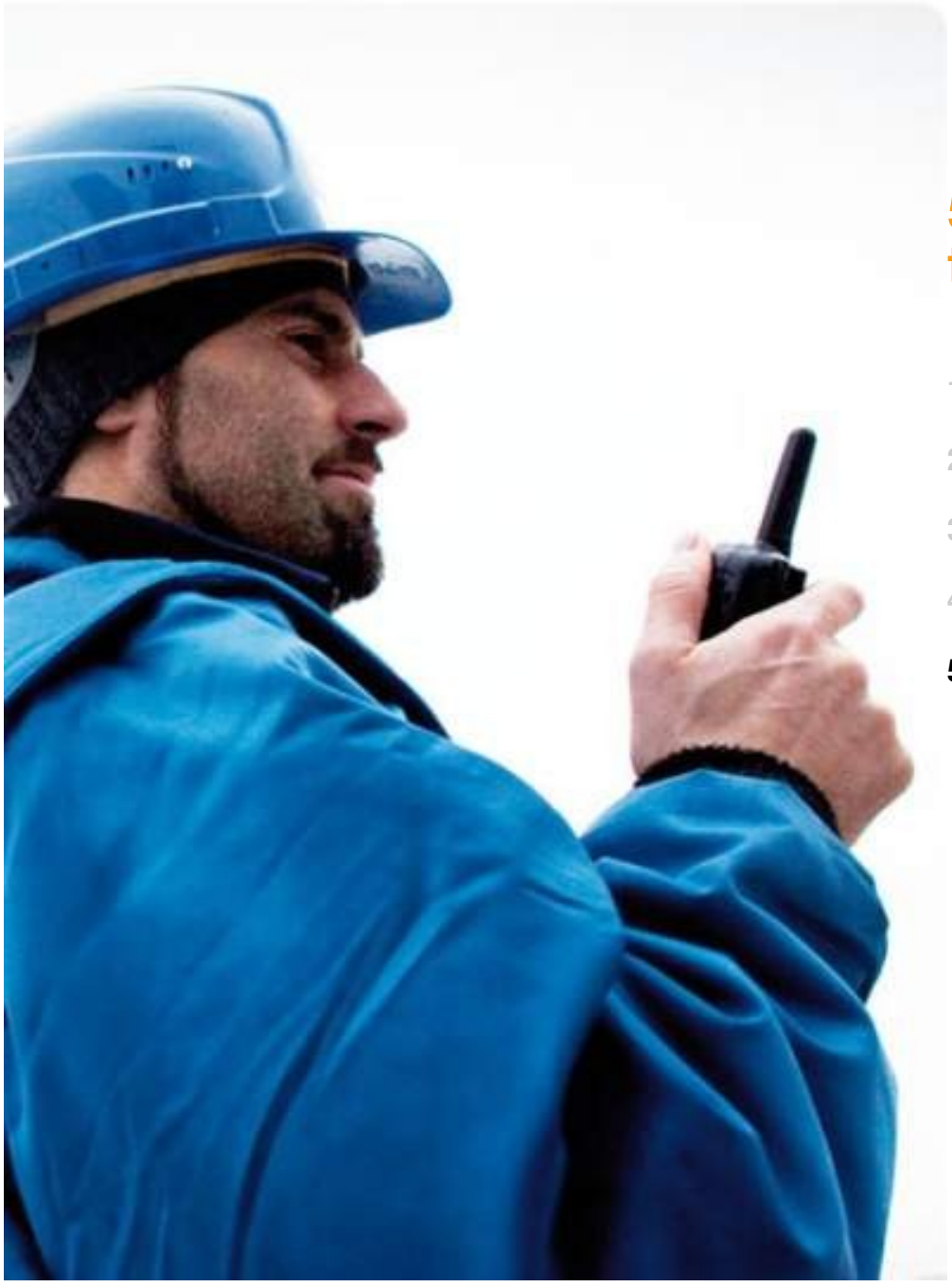
Experimental results

# Digits	# comp - # exec	# double - # float	Time (mm:ss)	Speed up	memory gain
10	83-51	19-32	88:56	1.01	1.00
8	80-48	18-33	85:10	1.01	1.01
6	69-37	13-38	71:32	1.20	1.44
5	3-3	0-51	9:58	1.32	1.62
4					

Experimental results

# Digits	# comp - # exec	# double - # float	Time (mm:ss)	Speed up	memory gain
10	83-51	19-32	88:56	1.01	1.00
8	80-48	18-33	85:10	1.01	1.01
6	69-37	13-38	71:32	1.20	1.44
5	3-3	0-51	9:58	1.32	1.62
4					

- Speed-up up to 1.32 and memory gain 1.62
- Mixed precision approach successful: Speed-up 1.20 and memory gain 1.44



5. Conclusion and Perspectives

1. Searching for a configuration
2. Validate a configuration
3. PROMISE
4. Experimental results
5. **Conclusion and Perspectives**

Conclusion and Perspectives

Conclusion:

- PROMISE has been successfully used on several codes
- PROMISE has found a new configuration each time

Perspectives:

- How to tune using three (or more) types instead of two?
- Can the performance be enhanced with a parallelization?

You can download PROMISE and a research report on:

`http://promise.lip6.fr`

Thank you for your attention
Any question?