When accuracy and performance meets

A brief overview of why accuracy matters

Arnault Ioualalen, CEO
ioualalen@numalis.com

Nicolas Normand, CTO
nnormand@numalis.com

Farah Benmouhoub, Student
farah.benmouhoub@etudiant.univ-perp.com

Nasrine Damouche, Assistant professor
nasrine.damouche@univ-perp.fr

Matthieu Martel, Professor
matthieu.martel@univ-perp.fr

Sleipner A

Collapses at installation

Patriot

Failed interception

Two meaningful examples

Float are not Reals

```
float counter  = 0.0f;
float increment = 1.0f;
for (int i = 0 ; i < 100000000 ; i++){
 counter = counter + increment;
}
print(counter);
```

What is printed at the end ?

Float are not Reals

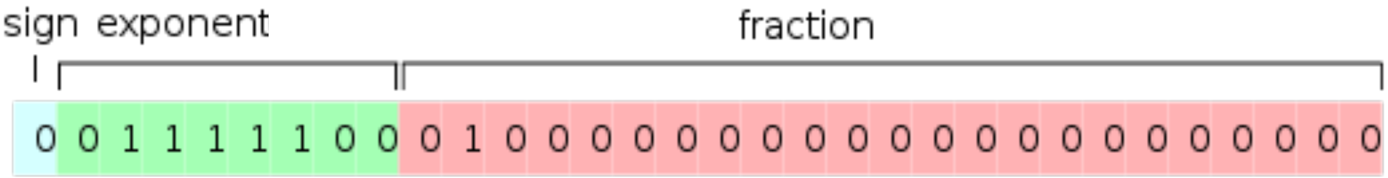
```
float counter  = 0.0f;
float increment = 1.0f;
for (int i = 0 ; i < 100000000 ; i++){
 counter = counter + increment;
}
print(counter);
```

```
$>  16 777 216.0
```

… ?!

Yes… that’s a true program !

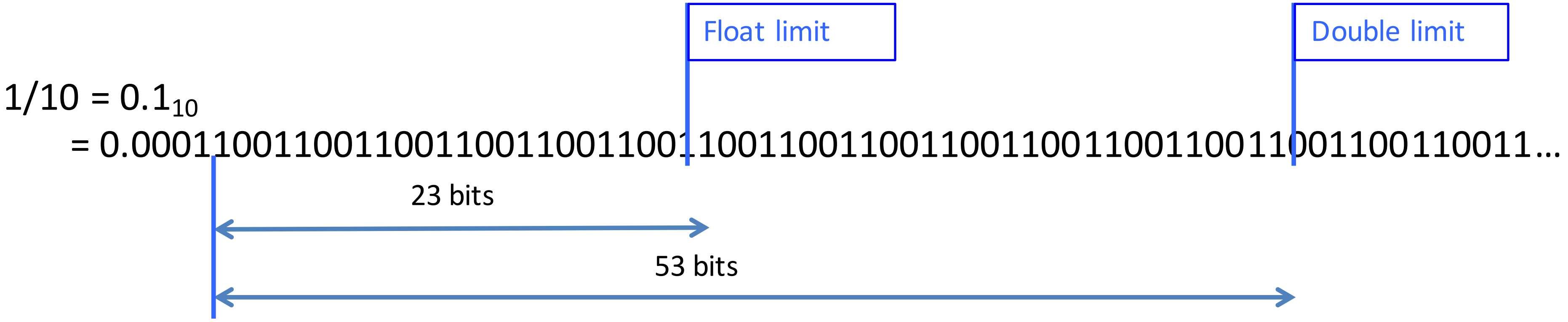
Float are not Reals



- Correct rounding for +, -, /, x, and $\sqrt{}$
- Cos, sin, tan, log, exp are not

Source of numerical errors

- Value rounding : 0,1 ≠ 0.100000001490116119384765625



Float limit

Double limit

$1/10 = 0.1_{10}$

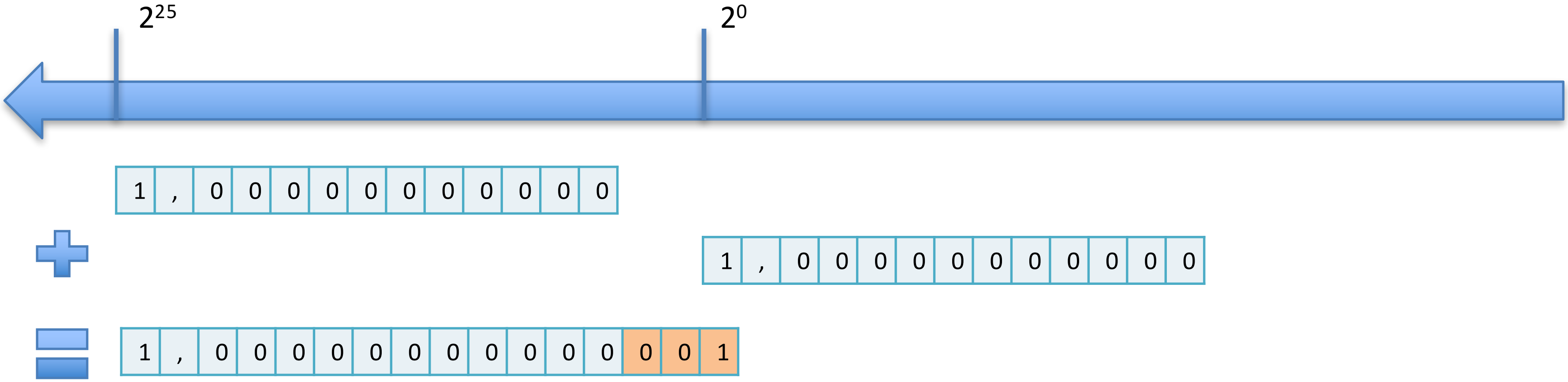
$= 0.000110011001100110011001100110011001100110011001100110011001100110011…$

23 bits

53 bits

Source of numerical errors

- Value rounding : 0,1 $\neq$ 0.100000001490116119384765625
- Absorption : $a + b = a \; and \; b \neq 0$

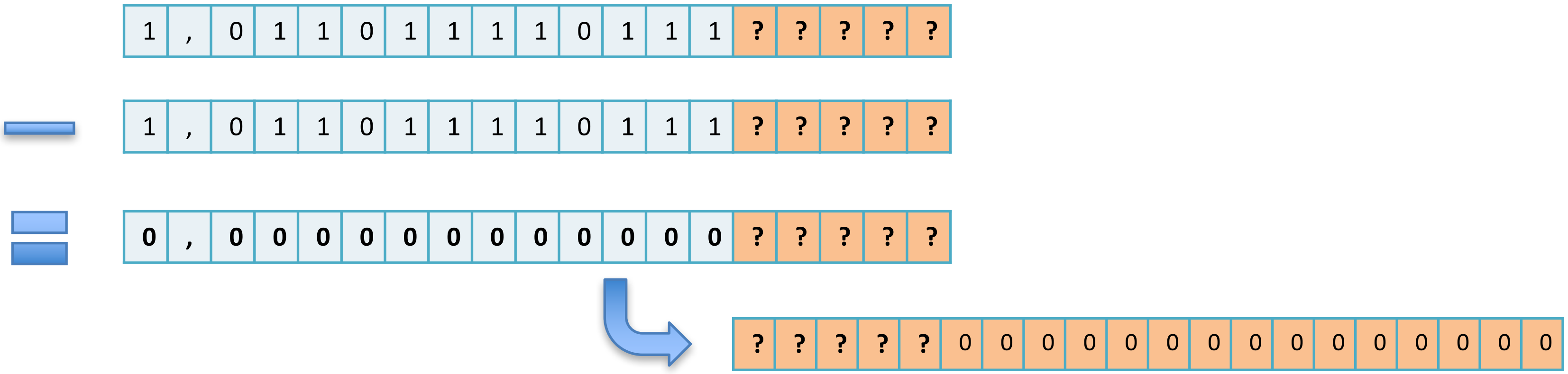


Source of numerical errors

- Value rounding : 0,1 ≠ 0.100000001490116119384765625

- Absorption : $a + b = a \text{ and } b \neq 0$

- Catastrophic cancelation : $c = a - b \text{ and } a \approx b$



1 , 0 1 1 0 1 1 1 1 0 1 1 1 ? ? ? ? ?

− 1 , 0 1 1 0 1 1 1 1 0 1 1 1 ? ? ? ? ?

= 0 , 0 0 0 0 0 0 0 0 0 0 0 0 ? ? ? ? ?

? ? ? ? ? 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Source of numerical errors

- Value rounding : 0,1 ≠ 0.100000001490116119384765625

- Absorption : $a + b = a \; and \; b \; \neq 0$

- Catastrophic cancelation : $c = a \; - b \; and \; a \approx b$

- Libraries : Libmath GCC ≠ Intel math library

- Hardware…

Float are not Reals

Things that do not hold anymore

- **Associativity**
- Distributivity
- Factorization
- Any trigonometric formula
- Unicity of neutral elements
- …

Things that still hold

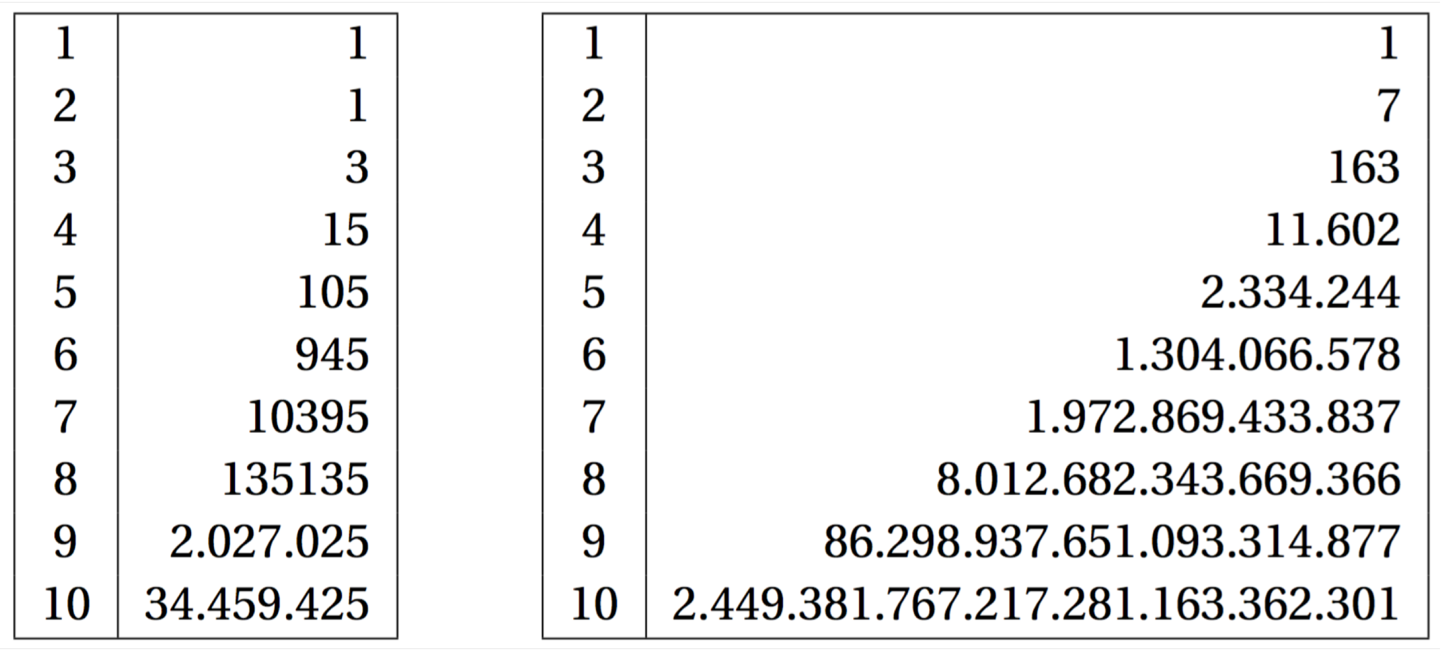
- Commutativity

Equivalence is gone !

Search space of expressions is intractable

# sum

polynomial

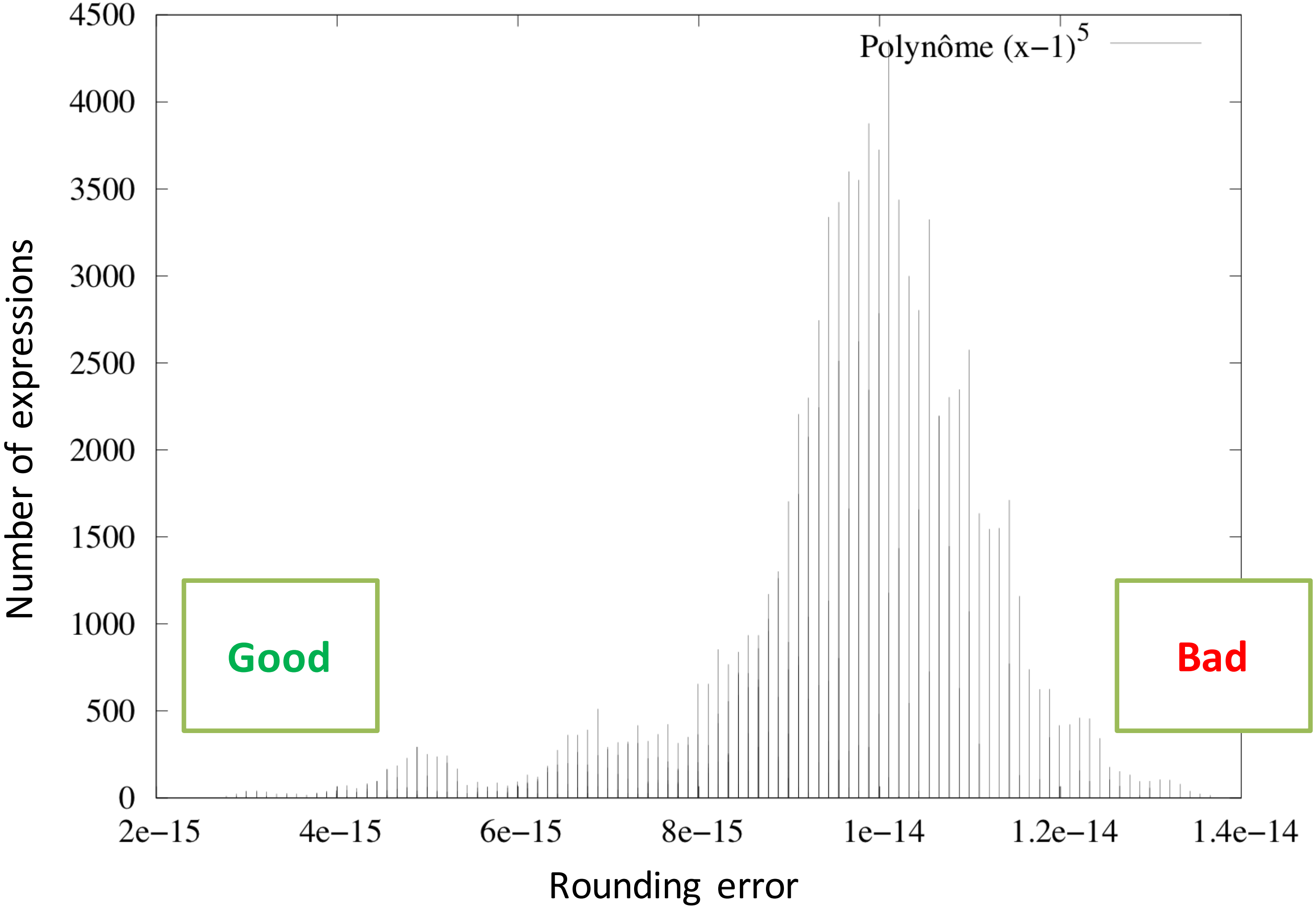


1	1
2	1
3	3
4	15
5	105
6	945
7	10395
8	135135
9	2.027.025
10	34.459.425

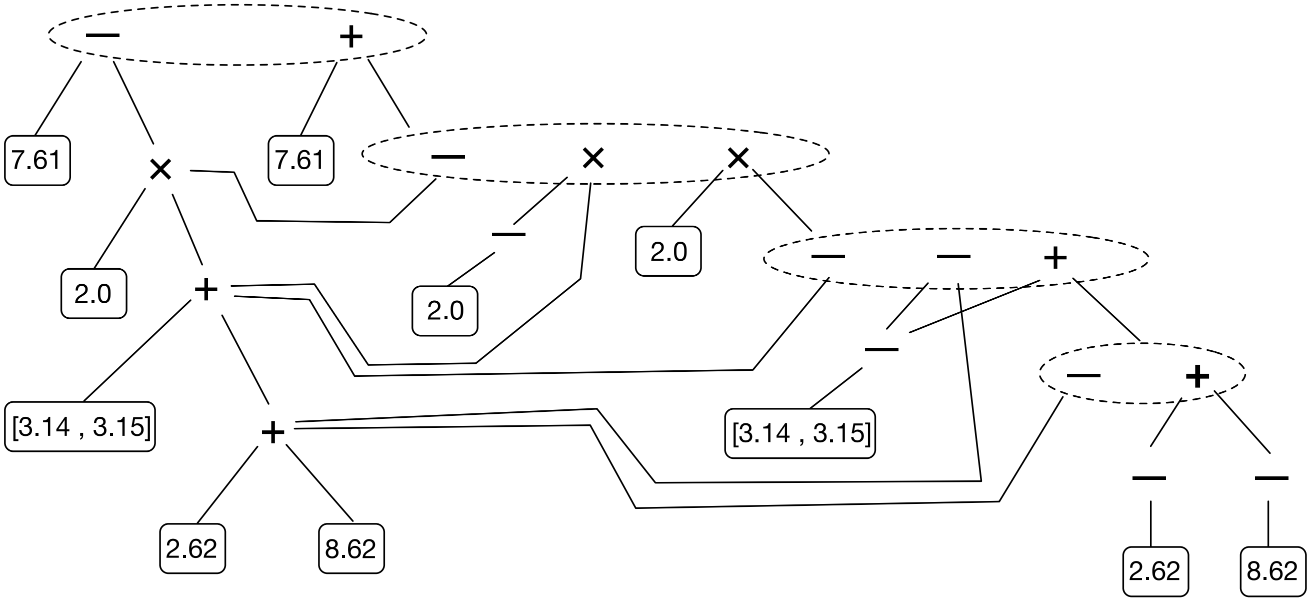
1	1
2	7
3	163
4	11.602
5	2.334.244
6	1.304.066.578
7	1.972.869.433.837
8	8.012.682.343.669.366
9	86.298.937.651.093.314.877
10	2.449.381.767.217.281.163.362.301

But what about its ”density”?

Few good ones, few bad ones, but how to find them?

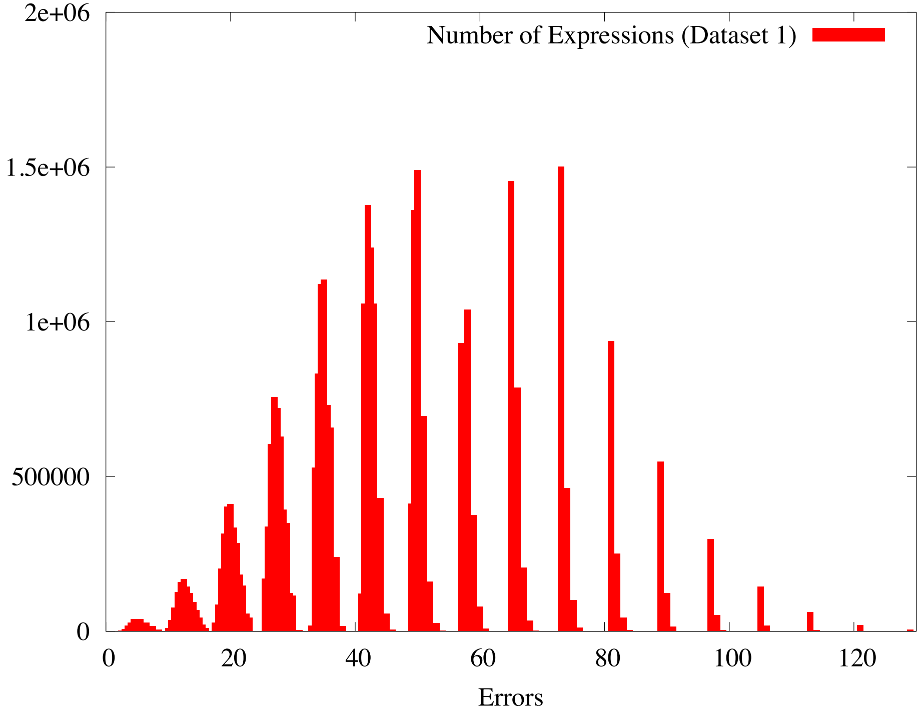


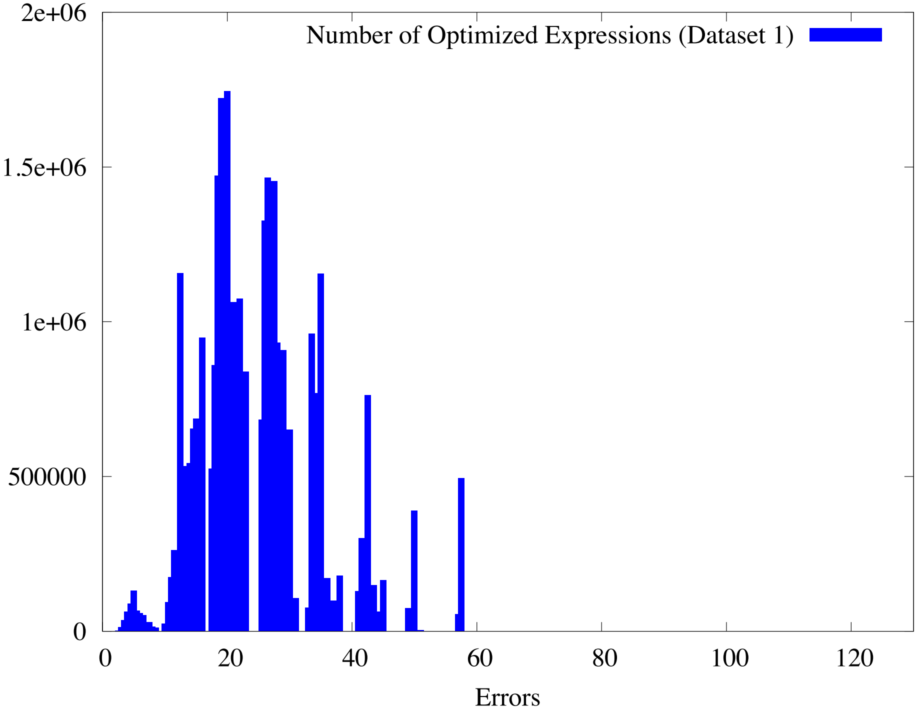
Abstract interpretation is here to help



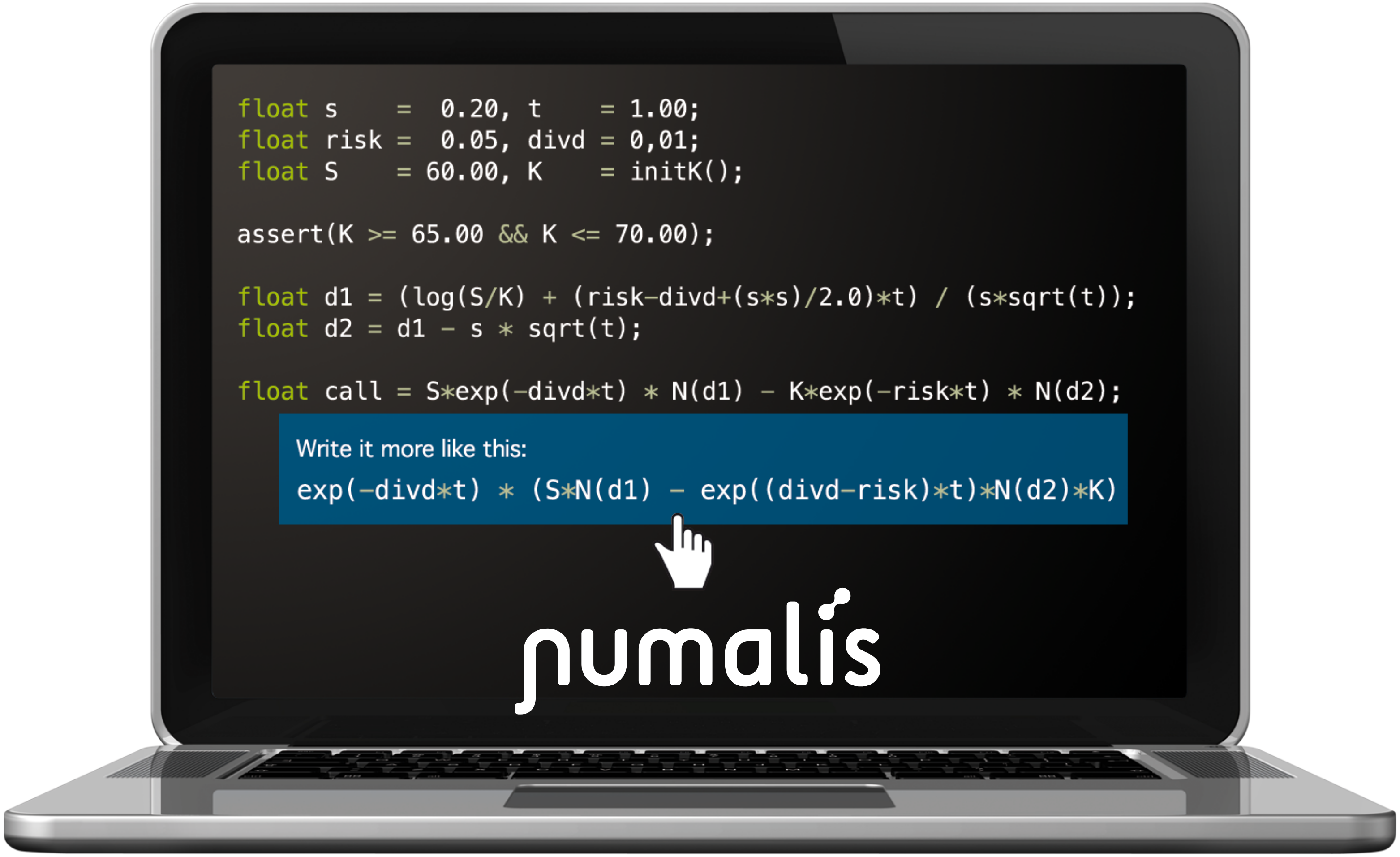
- Keep the size polynomial
- Represent an exponential number of expressions

Shifting toward the better ones automatically





Our solution:
Detect & correct automatically from the start



```
float s    =  0.20, t    = 1.00;
float risk =  0.05, divd = 0,01;
float S    = 60.00, K    = initK();

assert(K >= 65.00 && K <= 70.00);

float d1 = (log(S/K) + (risk-divd+(s*s)/2.0)*t) / (s*sqrt(t));
float d2 = d1 - s * sqrt(t);

float call = S*exp(-divd*t) * N(d1) - K*exp(-risk*t) * N(d2);
```

Write it more like this:

```
exp(-divd*t) * (S*N(d1) - exp((divd-risk)*t)*N(d2)*K)
```

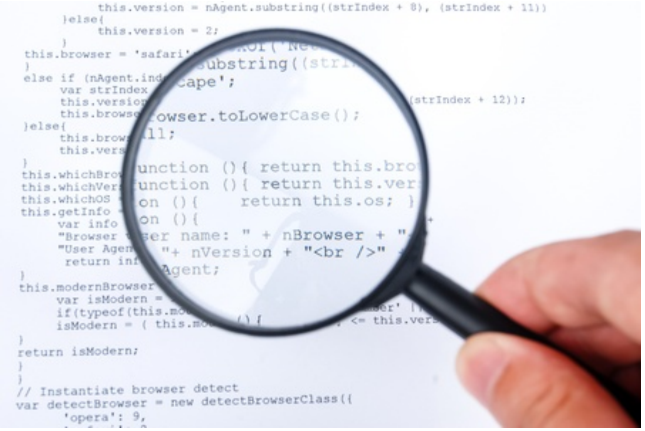
At the beginning there are **Bugs**

… **Pain** comes at the end

INVOICE

Description	Quantity	Price
Failed final test	1	
	TOTAL :	**$ 90.0 k** **4 months delay**

Inefficient bug correction process



- Spoat-Vulnerability

  Find every numerical flaws

- Spoat-Trust

  Determines the necessary amount of testing

- Wizoat-Stability

  Reduce the accumulating error

- Wizoat-Arbitrage

  Find the right accuracy/performance tradeoff

Our products

But why talk about this at Teratec?

Parallelism, vectorization

Restrained the evaluation order

Impact :

- Hard to anticipate

GPU

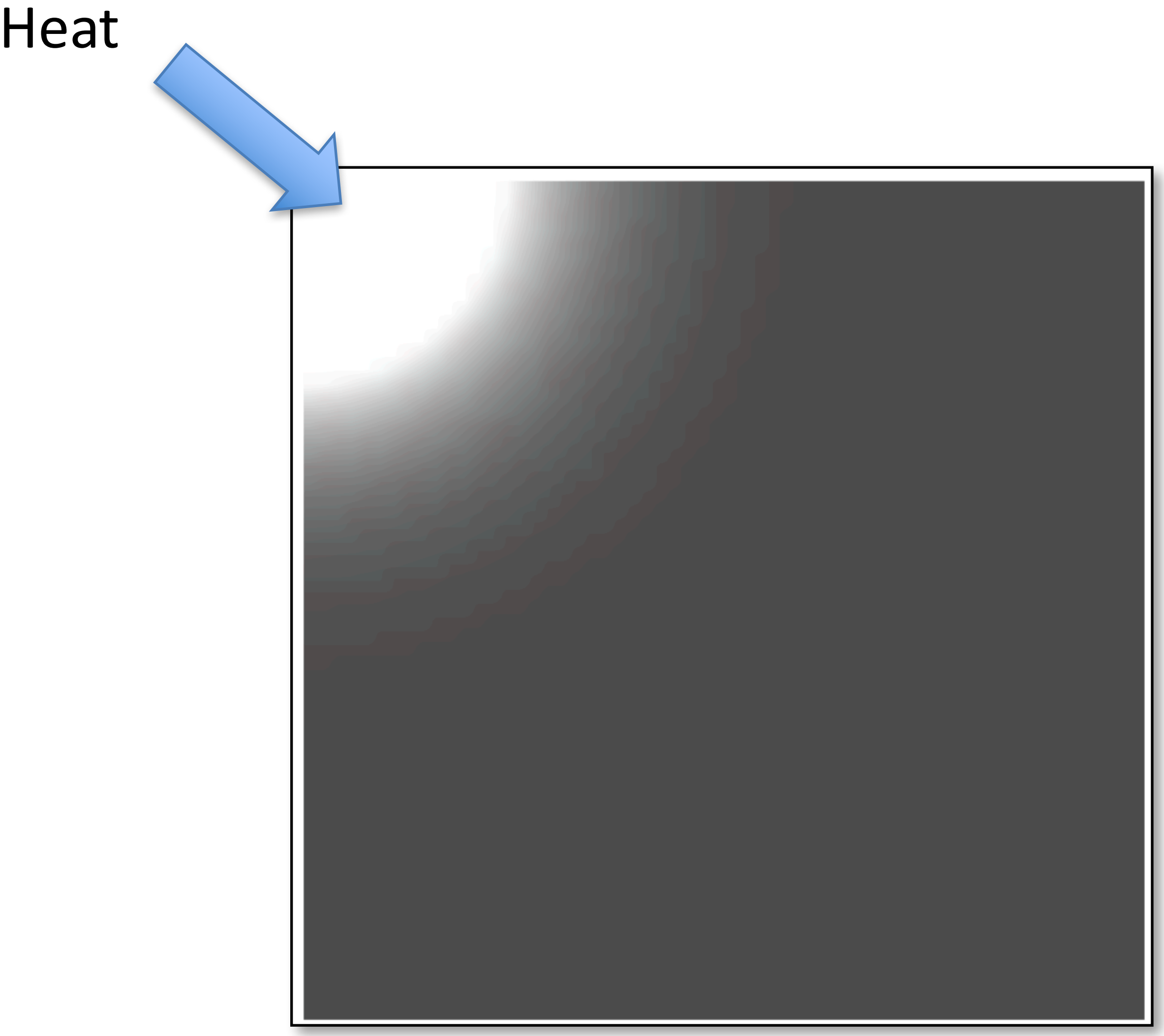
Evaluation order is not known

Smaller formats are often used

Impact:

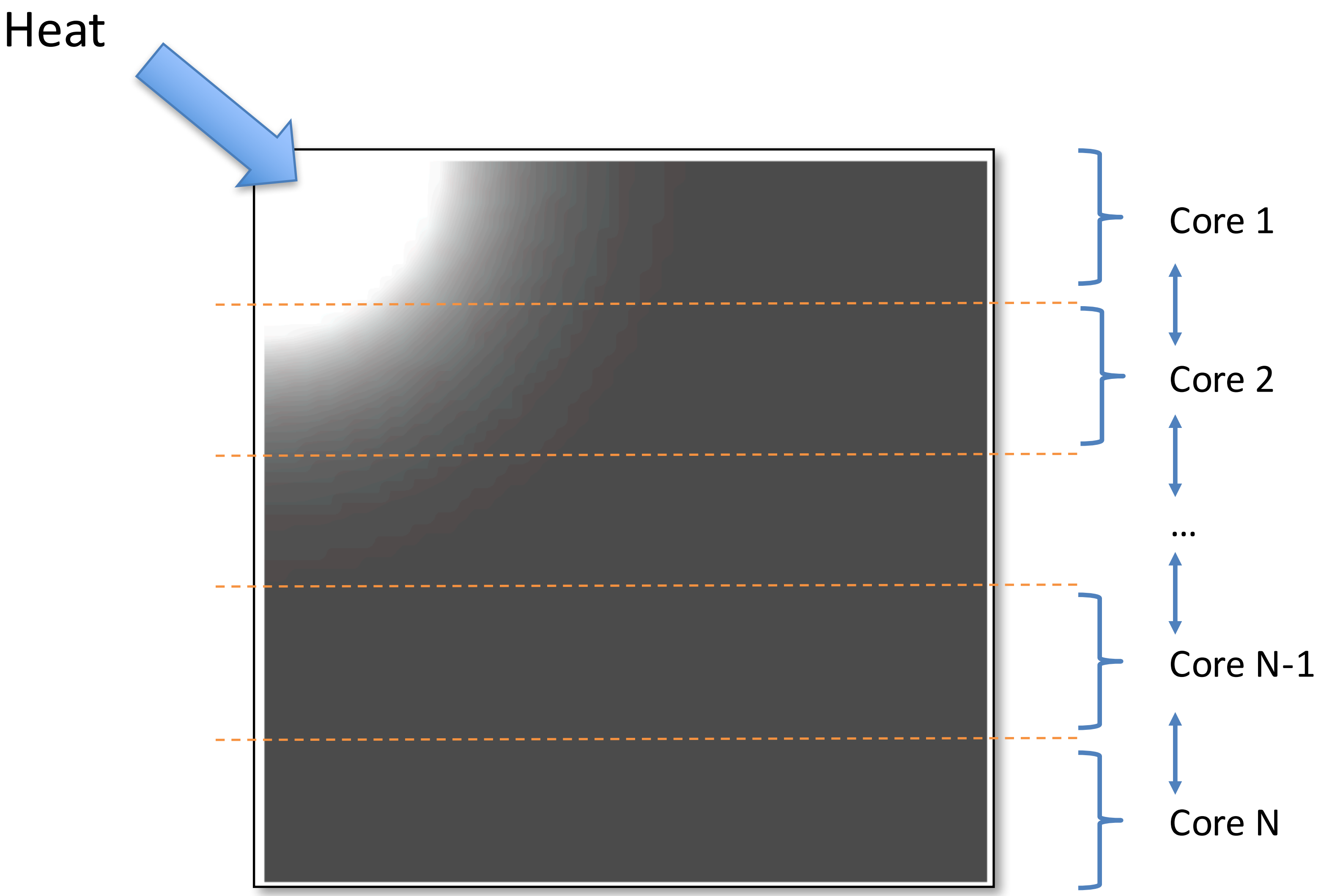
- “Almost” unpredictable

A simple simulation problem

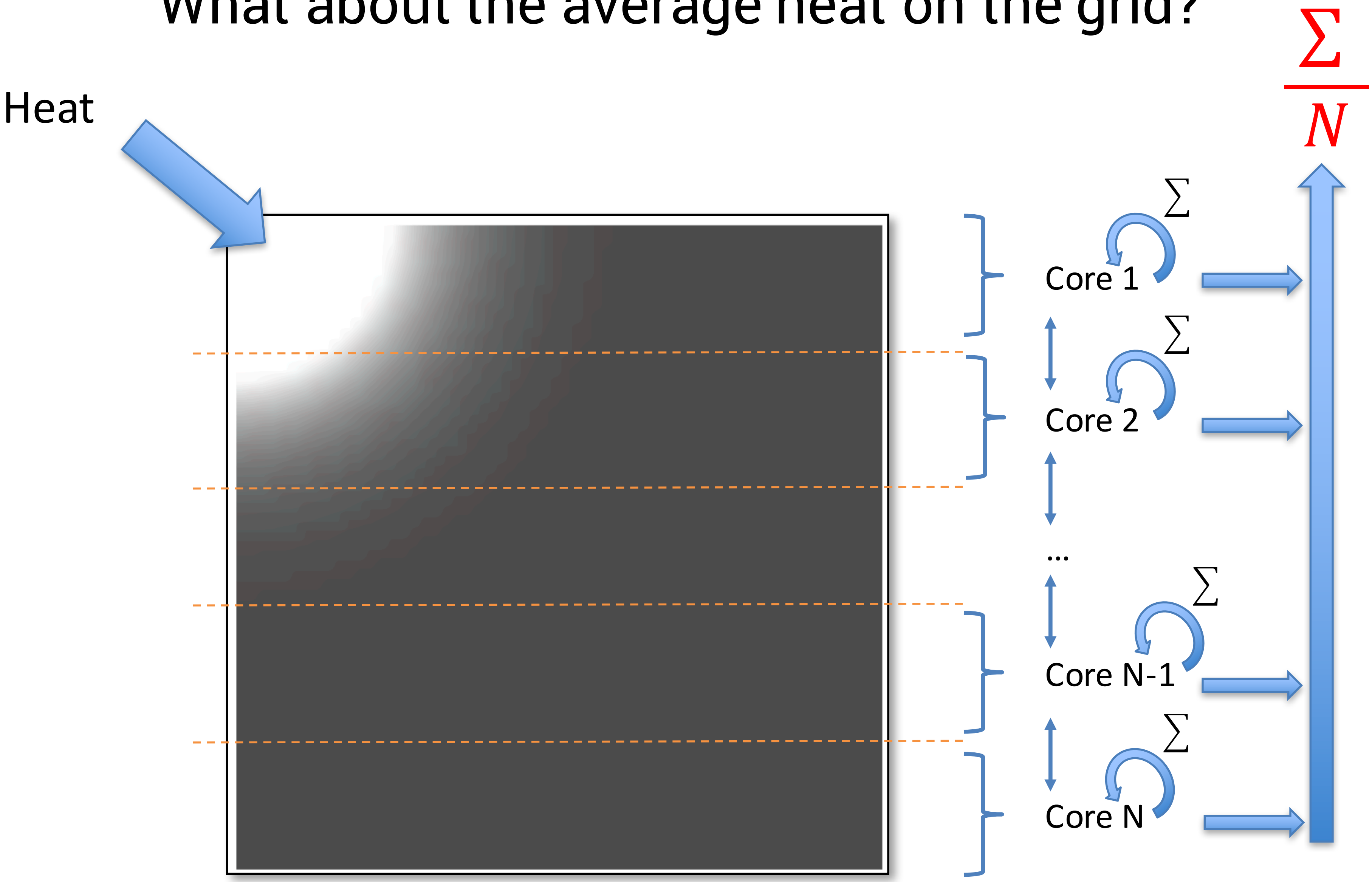


Fully sequential process

A parallelized version

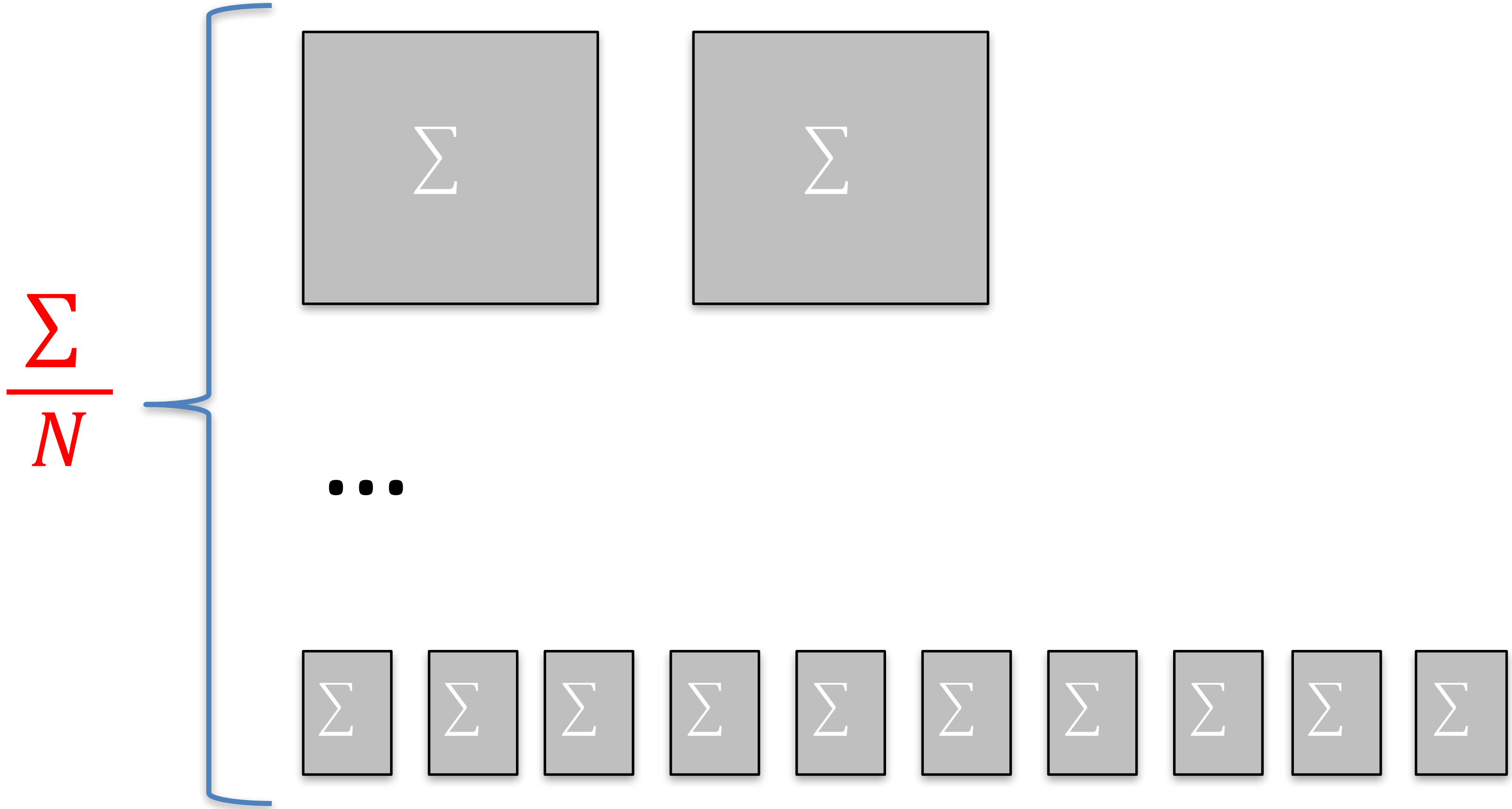


What about the average heat on the grid?



Need for an aggregator

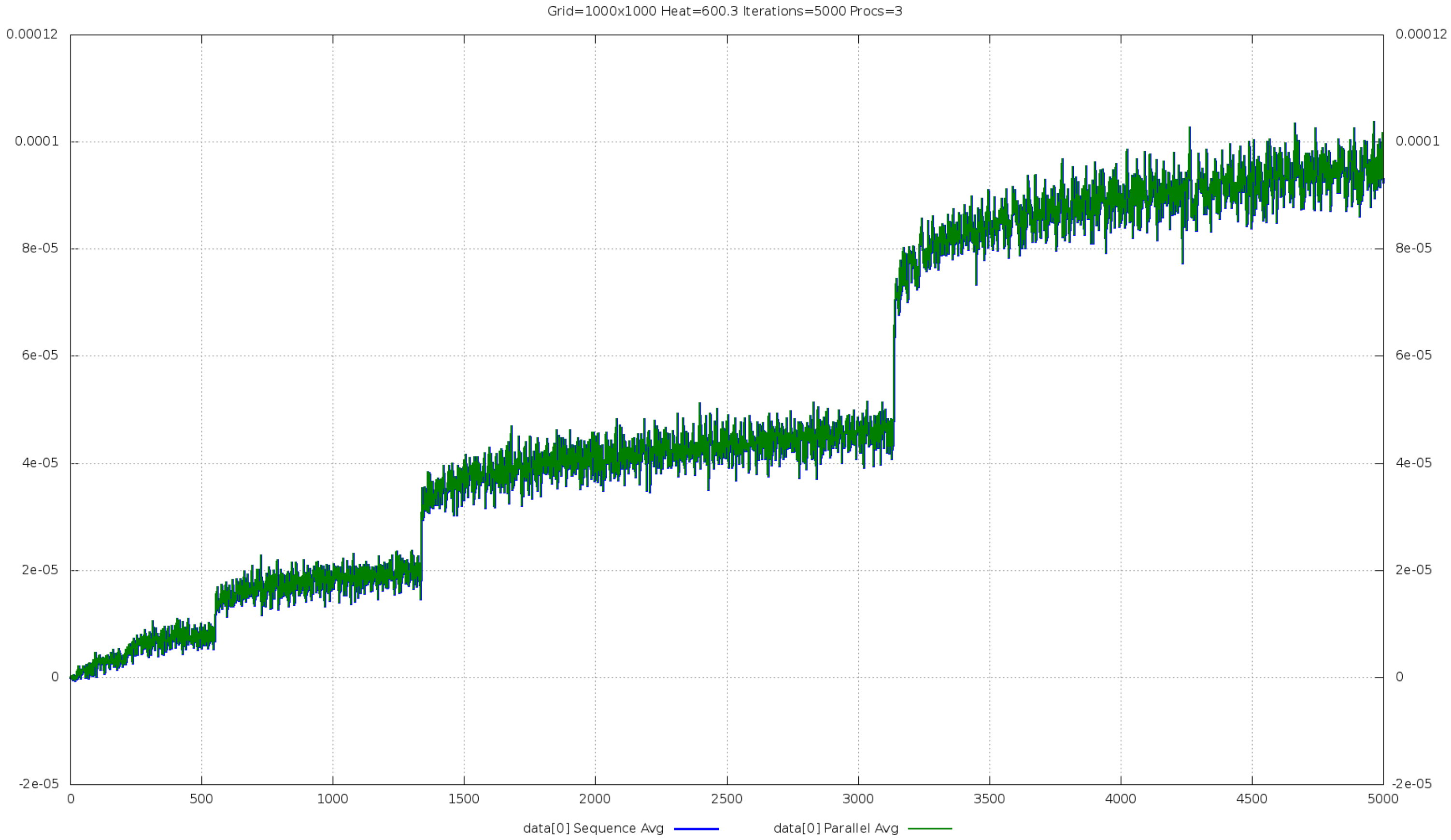
Number of cores will change the output



Different sum, different results

3 cores

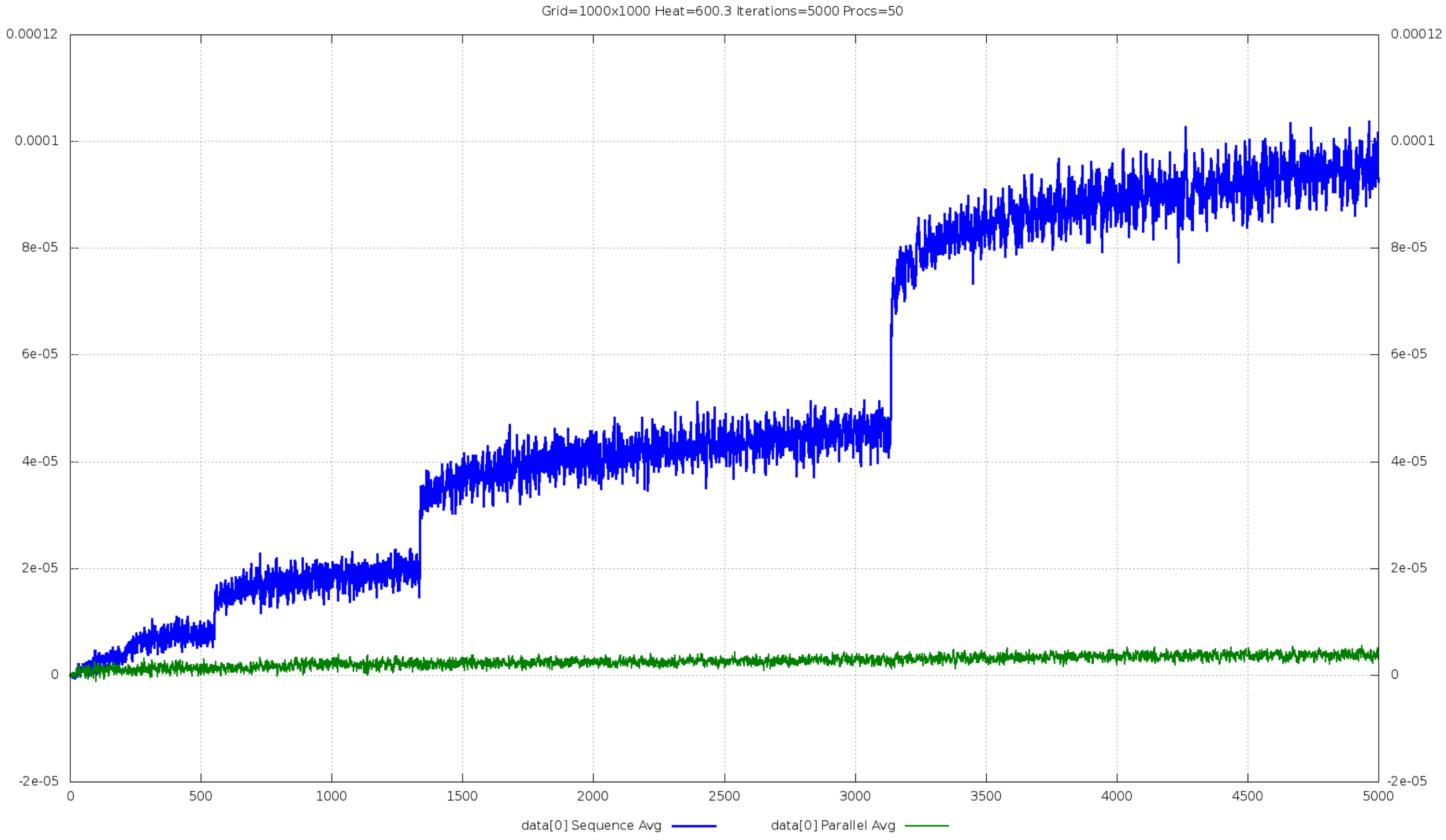
600° heat, 5k iterations, 1500^2 grid, single precision



0.01 %

Sequential and parallel are similar

50 cores
600° heat, 5k iterations, 1500^2 grid, single precision



0.01 %

0.0005 %

Parallel is 20x more accurate

Parallelism vs. accuracy

- Parallelism impacts accuracy (in good or bad)
- Performance and precision can mix… with caution
- Iterative methods can be affected by accuracy
  - What about parallel ones?
- Impact of local specialization on each core
  - Numalis and UPVD tools to optimize accuracy

Thanks for listening

numalis

UPVD
Université de Perpignan Via Domitia

Arnault Ioualalen, CEO
ioualalen@numalis.com

Nicolas Normand, CTO
nnormand@numalis.com

Farah Benmouhoub, Student
farah.benmouhoub@etudiant.univ-perp.com

Nasrine Damouche, Assistant professor
nasrine.damouche@univ-perp.fr

Matthieu Martel, Professor
matthieu.martel@univ-perp.fr