Reproducibility and Accuracy for High-Performance Computing

Roman Iakymchuk[1,2], Sylvain Collange[3], David Defour[4], and Stef Graillat[1]

[1] Sorbonne Universités, UPMC Univ Paris VI, UMR 7606, LIP6
[2] Sorbonne Universités, UPMC Univ Paris VI, ICS
[3] INRIA – Centre de recherche Rennes – Bretagne Atlantique
[4] DALI–LIRMM, Université de Perpignan

roman.iakymchuk@lip6.fr

RAIM 2015, April 7-9
Rennes, France



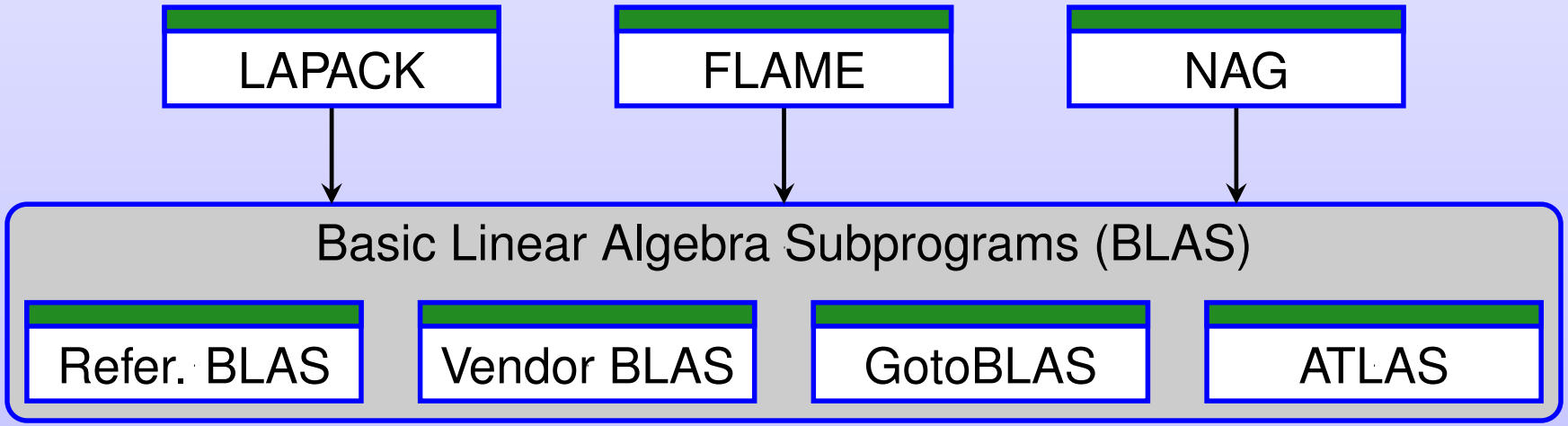


Motivation

BLAS-1 [1979]:	$y := y + \alpha x$ $\alpha := \alpha + x^T y$	$\alpha \in \mathbb{R}; x, y \in \mathbb{R}^n$	$2/3$
BLAS-2 [1988]:	$A := A + xy^T$ $y := A^{-1}x$	$A \in \mathbb{R}^{n \times n}; x, y \in \mathbb{R}^n$	2
BLAS-3 [1990]:	$C := C + AB$ $C := A^{-1}B$	$A, B, C \in \mathbb{R}^{n \times n}$	$n/2$

Motivation

BLAS-1 [1979]:	$y := y + \alpha x$ $\alpha := \alpha + x^T y$	$\alpha \in \mathbb{R}; x, y \in \mathbb{R}^n$	$2/3$
BLAS-2 [1988]:	$A := A + xy^T$ $y := A^{-1}x$	$A \in \mathbb{R}^{n \times n}; x, y \in \mathbb{R}^n$	2
BLAS-3 [1990]:	$C := C + AB$ $C := A^{-1}B$	$A, B, C \in \mathbb{R}^{n \times n}$	$n/2$



Ultimate Goal

- To compute BLAS operations with floating-point numbers fast and precise, ensuring their reproducibility, on a wide range of architectures

ExBLAS – **Ex**act **BLAS**

- ExBLAS-1: `ExSCAL`, `ExDOT`, `ExAXPY`, ...
- ExBLAS-2: `ExGER`, `ExGEMV`, `ExTRSV`, `ExSYR`, ...
- ExBLAS-3: `ExGEMM`, `ExTRSM`, `ExSYR2K`, ...

Outline

Problems

- Floating-point arithmetic suffers from rounding errors
- Floating-point operations $(+, \times)$ are commutative but non-associative

$$(-1+1)+2^{-53} \neq -1+(1+2^{-53}) \quad \text{in double precision}$$

Problems

- Floating-point arithmetic suffers from rounding errors
- Floating-point operations $(+, \times)$ are commutative but non-associative

$$2^{-53} \neq 0 \quad \text{in double precision}$$

Problems

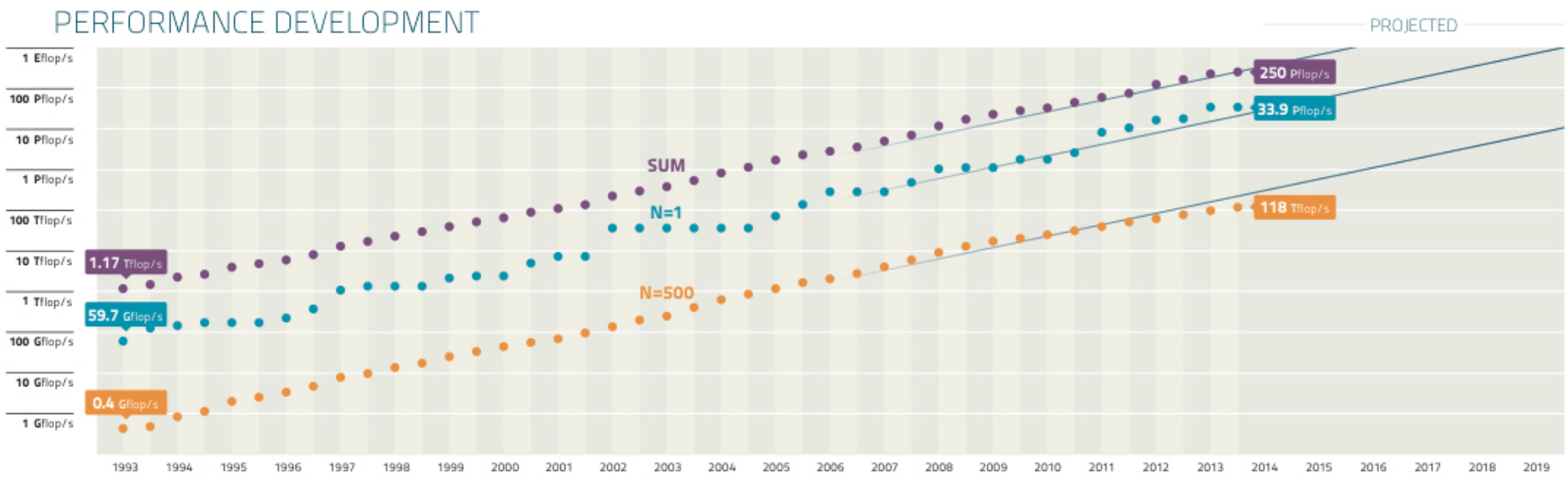
- Floating-point arithmetic suffers from rounding errors
- Floating-point operations $(+, \times)$ are commutative but non-associative

  $(-1 + 1) + 2^{-53} \neq -1 + (1 + 2^{-53})$ in double precision

- Consequence: results of floating-point computations depend on the order of computation
- Results computed by performance-optimized parallel floating-point libraries may be often inconsistent: each run returns a different result

TOP500: Nov 2014

	NAME	SPECS	SITE	COUNTRY	CORES	R_{MAX} PFLOP/S	POWER MW
1	**Tianhe-2 (Milkyway-2)**	NUDT, Intel Ivy Bridge (12C, 2.2 GHz) & Xeon Phi (57C, 1.1 GHz), Custom interconnect	NSCC Guangzhou	China	3,120,000	**33.9**	17.8
2	**Titan**	Cray XK7, Operon 6274 (16C 2.2 GHz) + Nvidia Kepler GPU, Custom interconnect	DOE/SC/ORNL	USA	560,640	**17.6**	8.2
3	**Sequoia**	IBM BlueGene/Q, Power BQC (16C 1.60 GHz), Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	**17.2**	7.9
4	**K computer**	Fujitsu SPARC64 VIIIfx (8C, 2.0GHz), Custom interconnect	RIKEN AICS	Japan	705,024	**10.5**	12.7
5	**Mira**	IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect	DOE/SC/ANL	USA	786,432	**8.59**	3.95



Source: www.top500.org

Reproducibility at ExaScale

- **Reproducibility** – ability to obtain bit-wise identical results from run-to-run on the same input data on the same or different architectures
- **ExaScale** – ability to perform exaflops (10^{18} floating-point operations) per second

Challenges

- Increasing power of current computers
  - $\rightarrow$ GPU accelerators, Intel Phi processors, etc.
- Enable to solve more complex problems
  - $\rightarrow$ Quantum field theory, supernova simulation, etc.
- A high number of floating-point operations performed
  - $\rightarrow$ Each of them leads to round-off error

Reproducibility at ExaScale

- **Reproducibility** – ability to obtain bit-wise identical results from run-to-run on the same input data on the same or different architectures
- **ExaScale** – ability to perform exaflops (10^{18} floating-point operations) per second

Challenges

- Increasing power of current computers
  - $\rightarrow$ GPU accelerators, Intel Phi processors, etc.
- Enable to solve more complex problems
  - $\rightarrow$ Quantum field theory, supernova simulation, etc.
- A high number of floating-point operations performed
  - $\rightarrow$ Each of them leads to round-off error

$\Downarrow$

Difficult to obtain accurate and reproducible results

Sources of Non-Reproducibility

Performance-optimized floating-point libraries are prone to non-reproducibility for various reasons:

- Changing Data Layouts:
  - Data partitioning
  - Data alignment

Sources of Non-Reproducibility

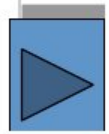
Performance-optimized floating-point libraries are prone to non-reproducibility for various reasons:

- Changing Data Layouts:
  - Data partitioning
  - Data alignment
- Changing Hardware Resources
  - Number of threads
  - Fused Multiply-Add support
  - Intermediate precision (64 bits, 80 bits, 128 bits, etc)
  - Data path (SSE, AVX, GPU warp, etc)
  - Cache line size
  - Number of processors
  - Network topology

Top 10 Challenges to Exascale

3 Hardware, 4 Software, 3 Algorithms/Math Related

- **Energy efficiency:**
  - Creating more energy efficient circuit, power, and cooling technologies.
- **Interconnect technology:**
  - Increasing the performance and energy efficiency of data movement.
- **Memory Technology:**
  - Integrating advanced memory technologies to improve both capacity and bandwidth.
- **Scalable System Software:**
  - Developing scalable system software that is power and resilience aware.
- **Programming systems:**
  - Inventing new programming environments that express massive parallelism, data locality, and resilience
- **Data management:**
  - Creating data management software that can handle the volume, velocity and diversity of data that is anticipated.
- **Scientific productivity:**
  - Increasing the productivity of computational scientists with new software engineering tools and environments.
- **Exascale Algorithms:**
  - Reformulating science problems and refactoring their solution algorithms for exascale systems.
- **Algorithms for discovery, design, and decision:**
  - Facilitating mathematical optimization and uncertainty quantification for exascale discovery, design, and decision making.
- **Resilience and correctness:**
  - Ensuring correct scientific computation in face of faults, reproducibility, and algorithm verification challenges.



Source: Algorithmic and Software Challenges at Extreme Scales by J.Dongarra at SCAN14

Existing Solutions

- **Fix the Order of Computations**
  - Sequential mode: intolerably costly at large-scale systems
  - Fixed reduction trees: substantial communication overhead

  → Example: Intel **C**onditional **N**umerical **R**eproducibility (slow, no accuracy guarantees)

Existing Solutions

- **Fix the Order of Computations**
  - Sequential mode: intolerably costly at large-scale systems
  - Fixed reduction trees: substantial communication overhead
  - → Example: Intel **C**onditional **N**umerical **R**eproducibility (slow, no accuracy guarantees)
- **Eliminate/Reduce the Rounding Errors**
  - Fixed-point arithmetic: limited range of values
  - Fixed FP expansions with Error-Free Transformations (EFT)
  - → Example: double-double or quad-double (Briggs, Bailey, Hida, Li) (work well on a set of relatively close numbers)
  - "Infinite" precision: reproducible independently from the inputs
  - → Example: Kulisch accumulator (considered inefficient)

Existing Solutions

- **Fix the Order of Computations**
  - Sequential mode: intolerably costly at large-scale systems
  - Fixed reduction trees: substantial communication overhead

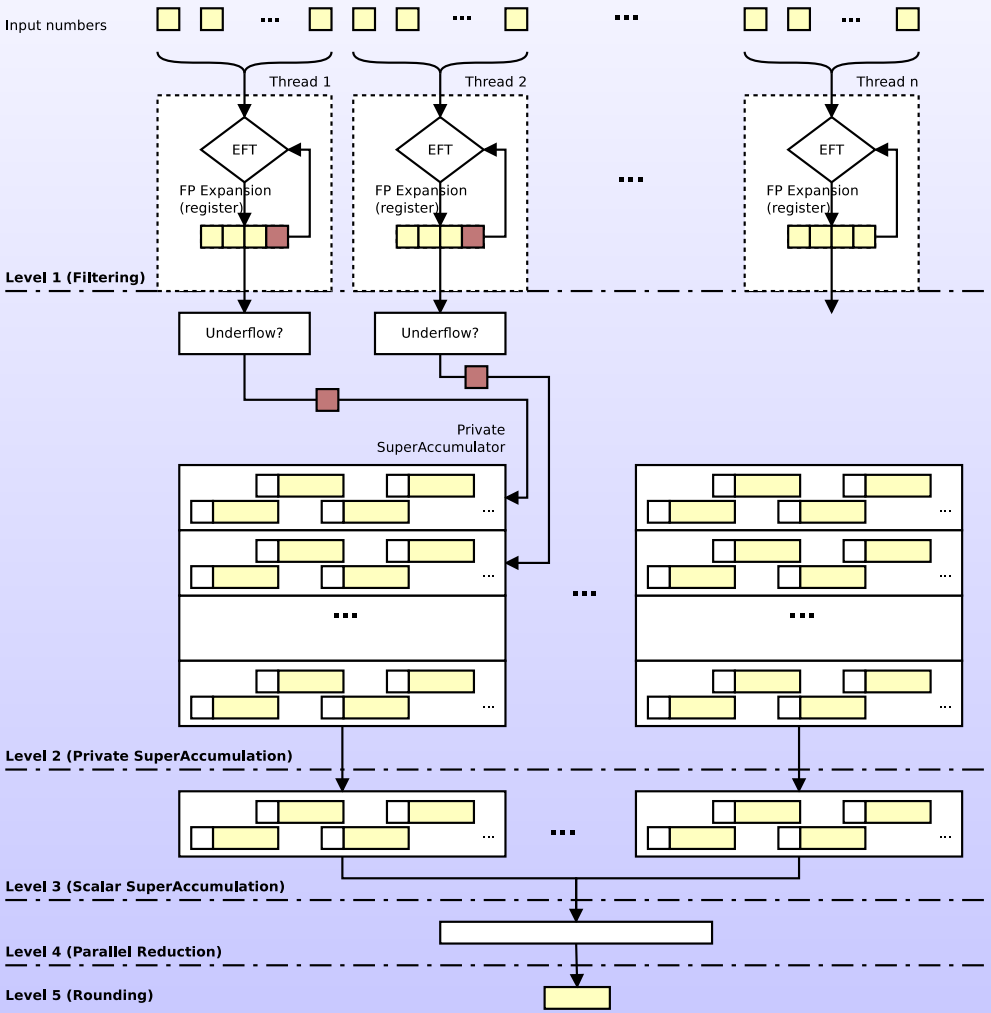
  → Example: Intel **C**onditional **N**umerical **R**eproducibility (slow, no accuracy guarantees)
- **Eliminate/Reduce the Rounding Errors**
  - Fixed-point arithmetic: limited range of values
  - Fixed FP expansions with Error-Free Transformations (EFT)

  → Example: double-double or quad-double (Briggs, Bailey, Hida, Li) (work well on a set of relatively close numbers)
  - "Infinite" precision: reproducible independently from the inputs

  → Example: Kulisch accumulator (considered inefficient)
- **Libraries**
  - ReproBLAS: Reproducible BLAS (Demmel and Nguyen)

  → For BLAS-1 on CPUs only

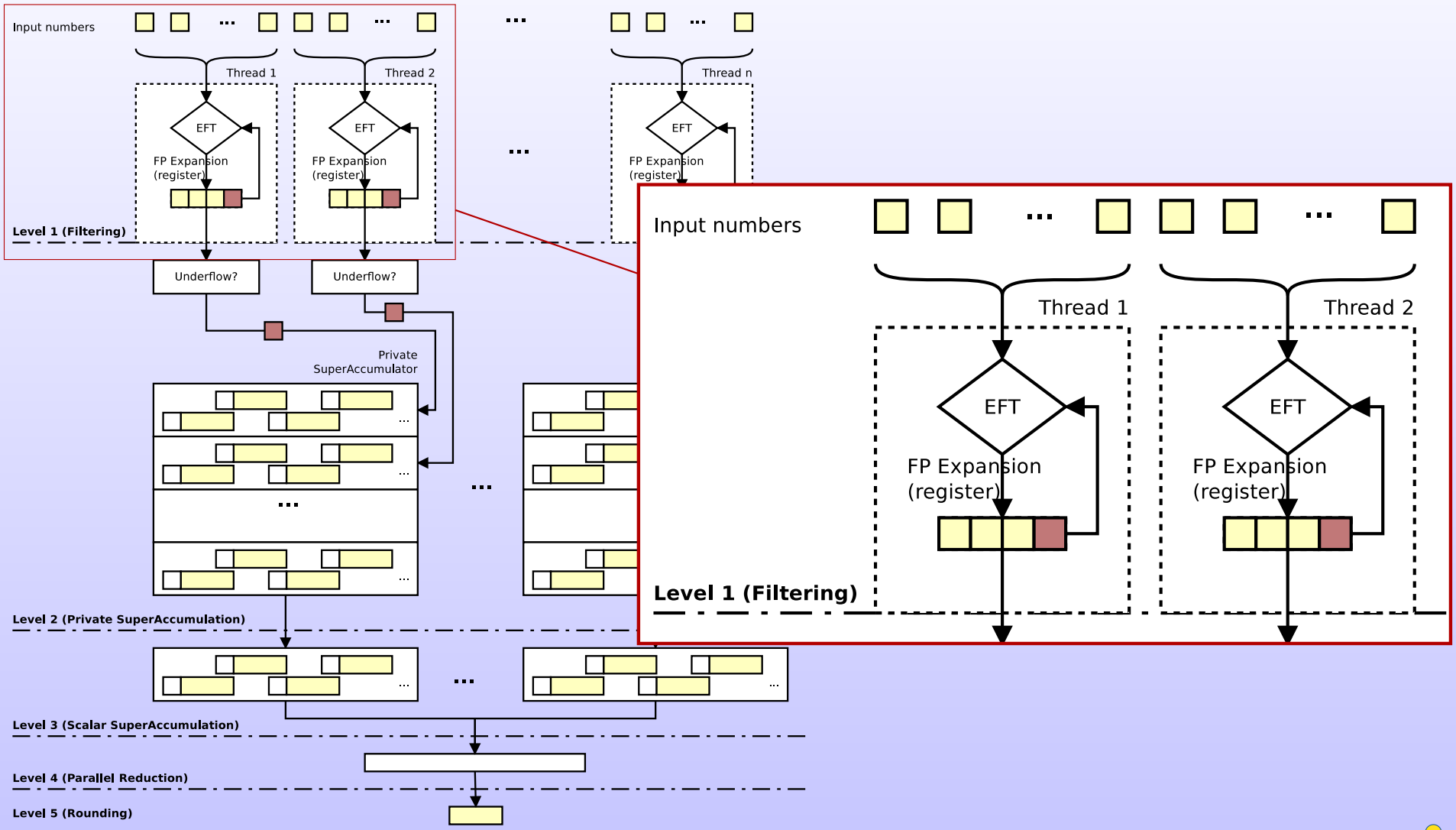
Our Multi-Level Reproducible Summation



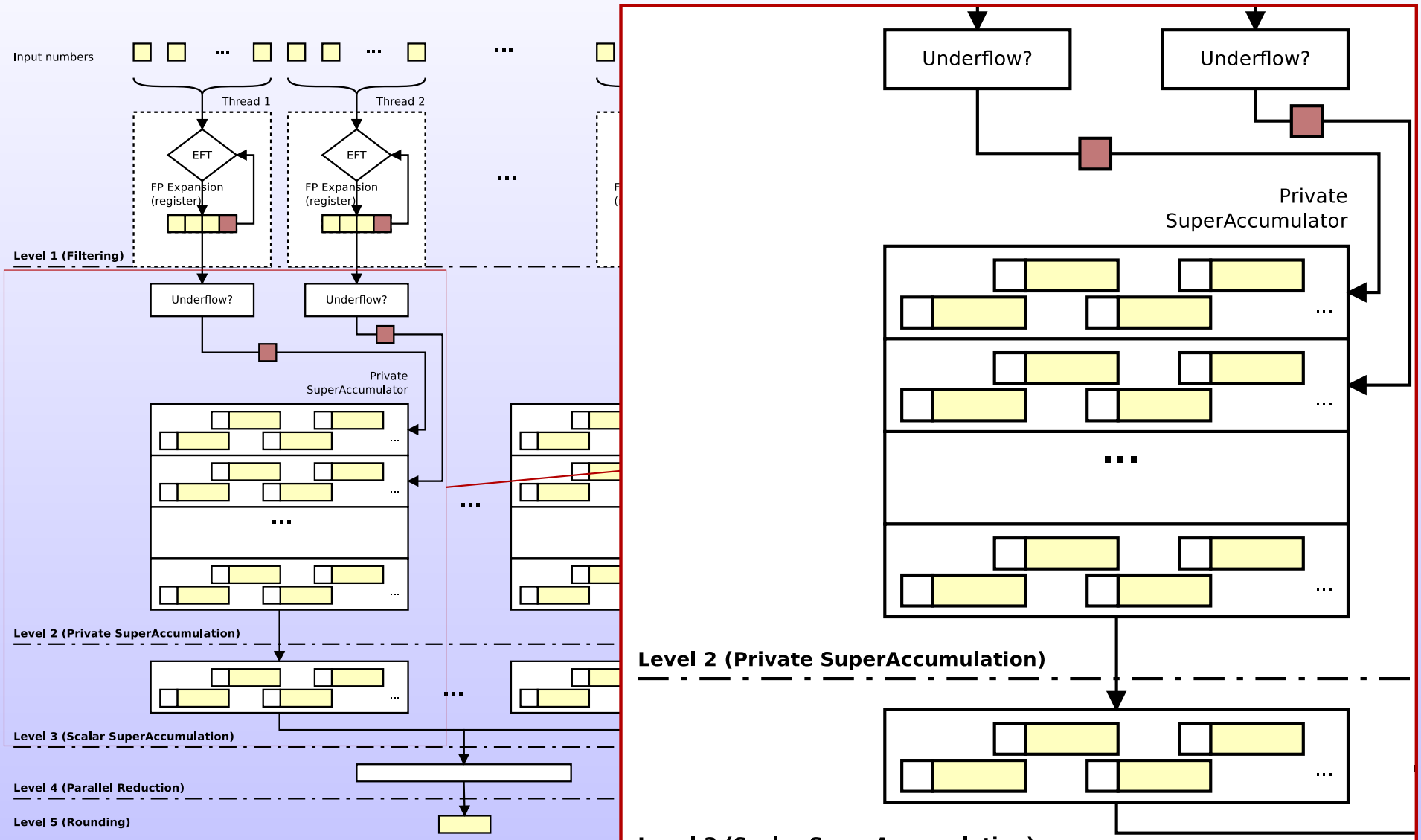
- Parallel algorithm with 5-levels
- Suitable for today's parallel architectures
- Based on FPE with EFT and Kulisch accumulator
- Guarantees "inf" precision

→ bit-wise reproductibility

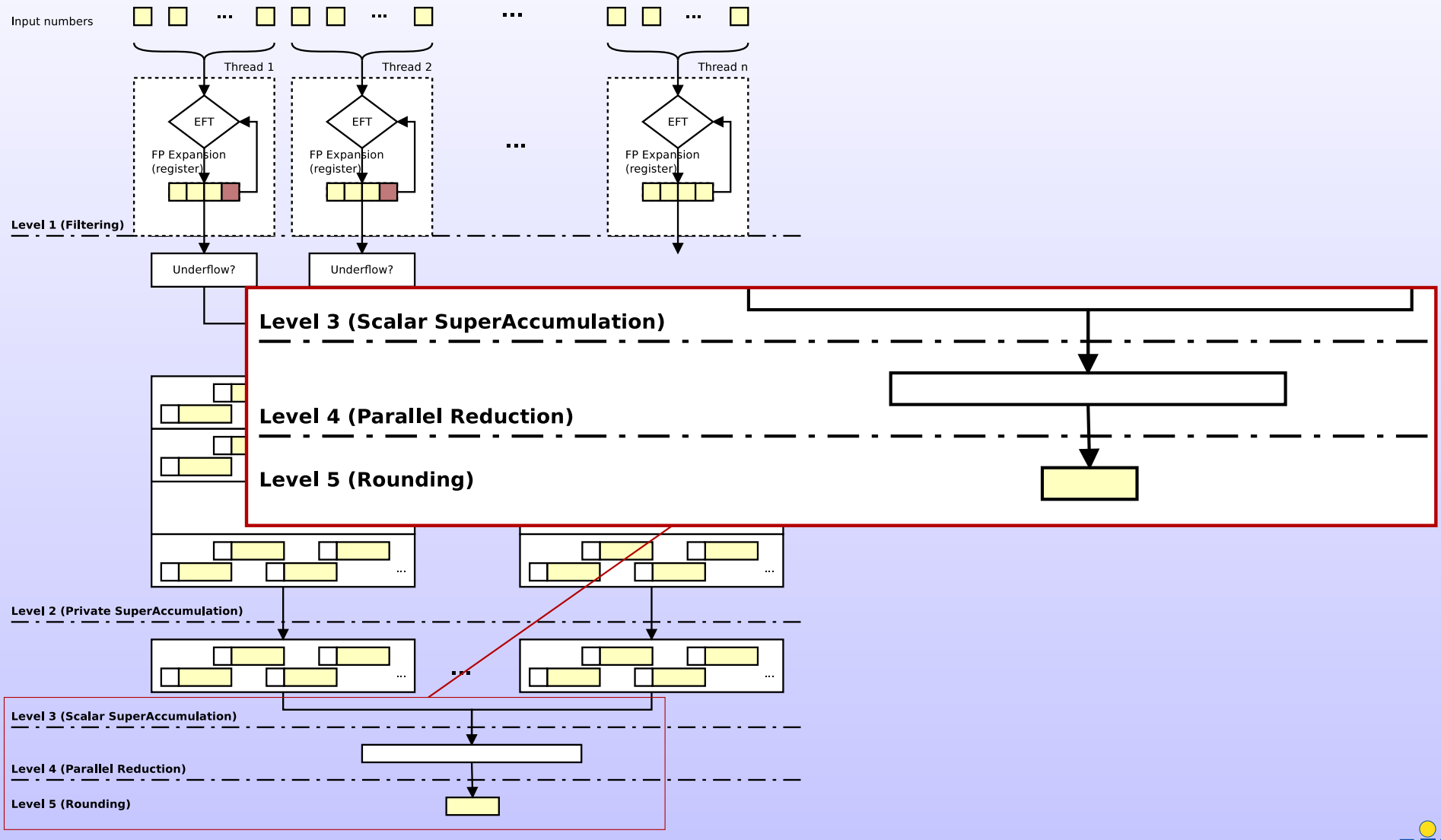
Level 1: Filtering



Level 2 and 3: Scalar Superaccumulator



Level 4 and 5: Reduction and Rounding



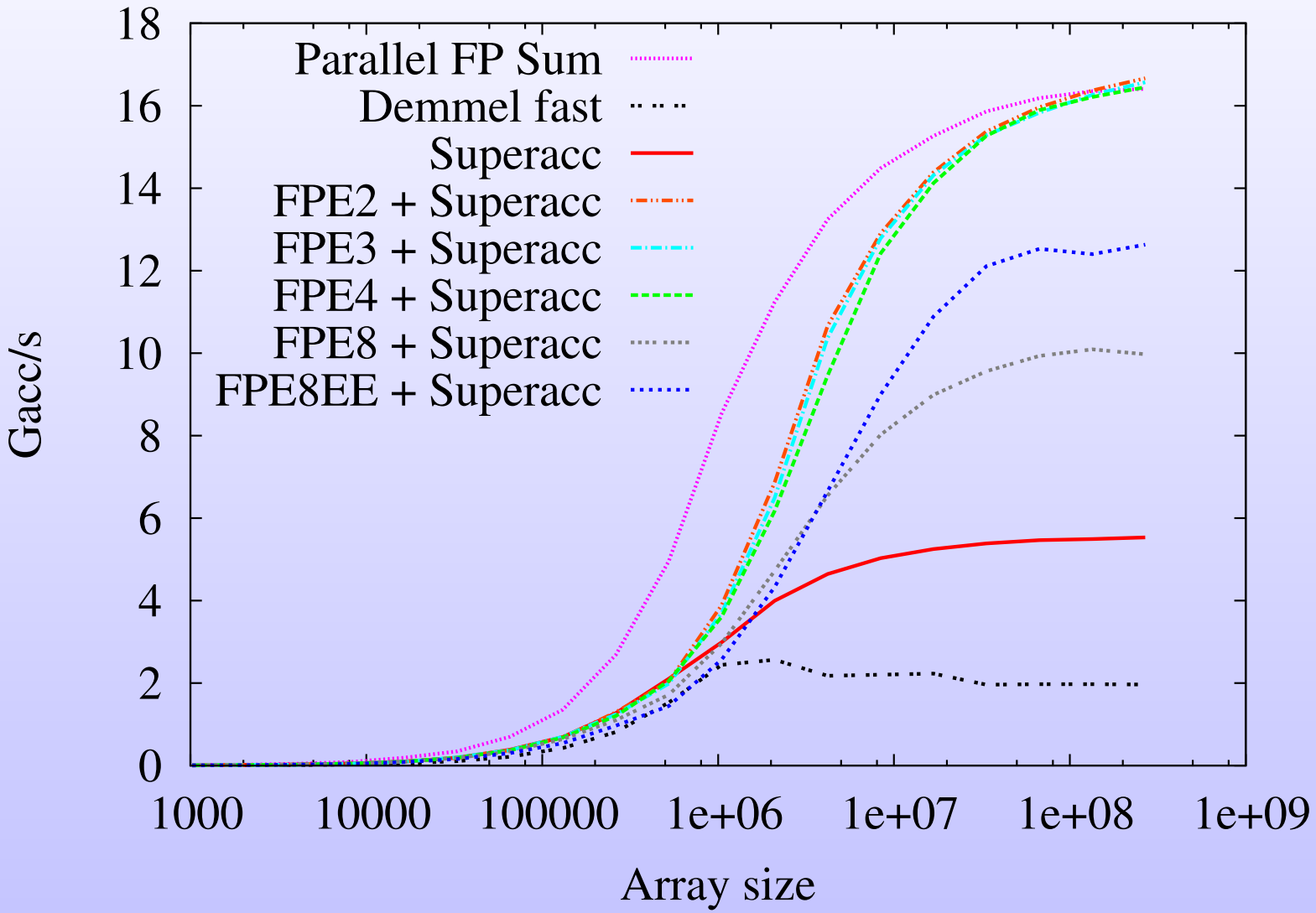
Experimental Environments

Table : Hardware platforms employed in the experimental evaluation

Intel Core i7-4770 (Haswell)	4 cores with HT
Mesu cluster (Intel Sandy Bridge)	$64 \times 2 \times 8$ cores
Intel Xeon Phi 3110P	60 cores $\times$ 4-way MT
NVIDIA Tesla K20c	13 SMs $\times$ 192 CUDA cores
NVIDIA Quadro K5000	8 SMs $\times$ 192 CUDA cores
AMD Radeon HD 7970	32 CUs $\times$ 64 units

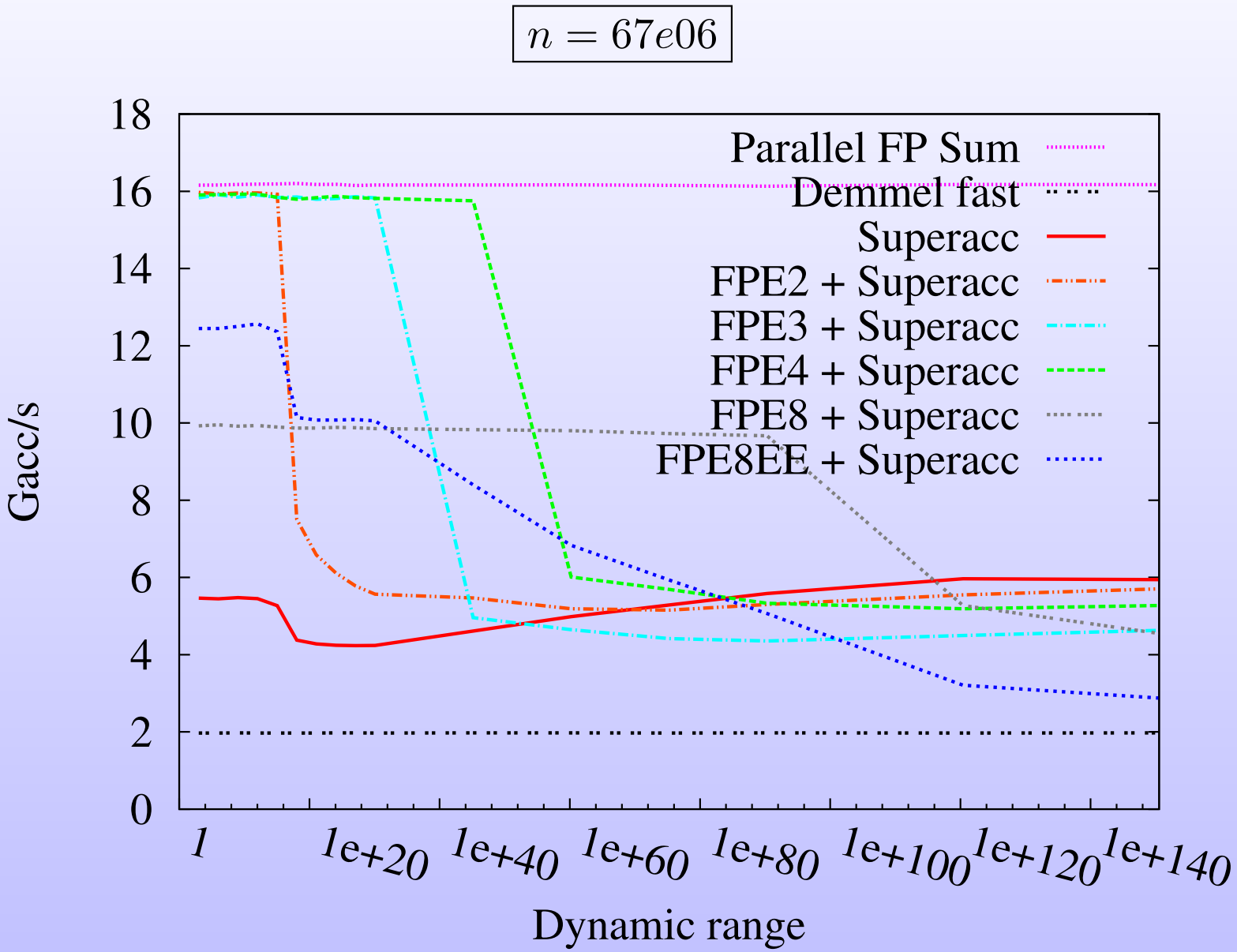
Parallel Summation

Performance Scaling on NVIDIA Tesla K20c



Parallel Summation

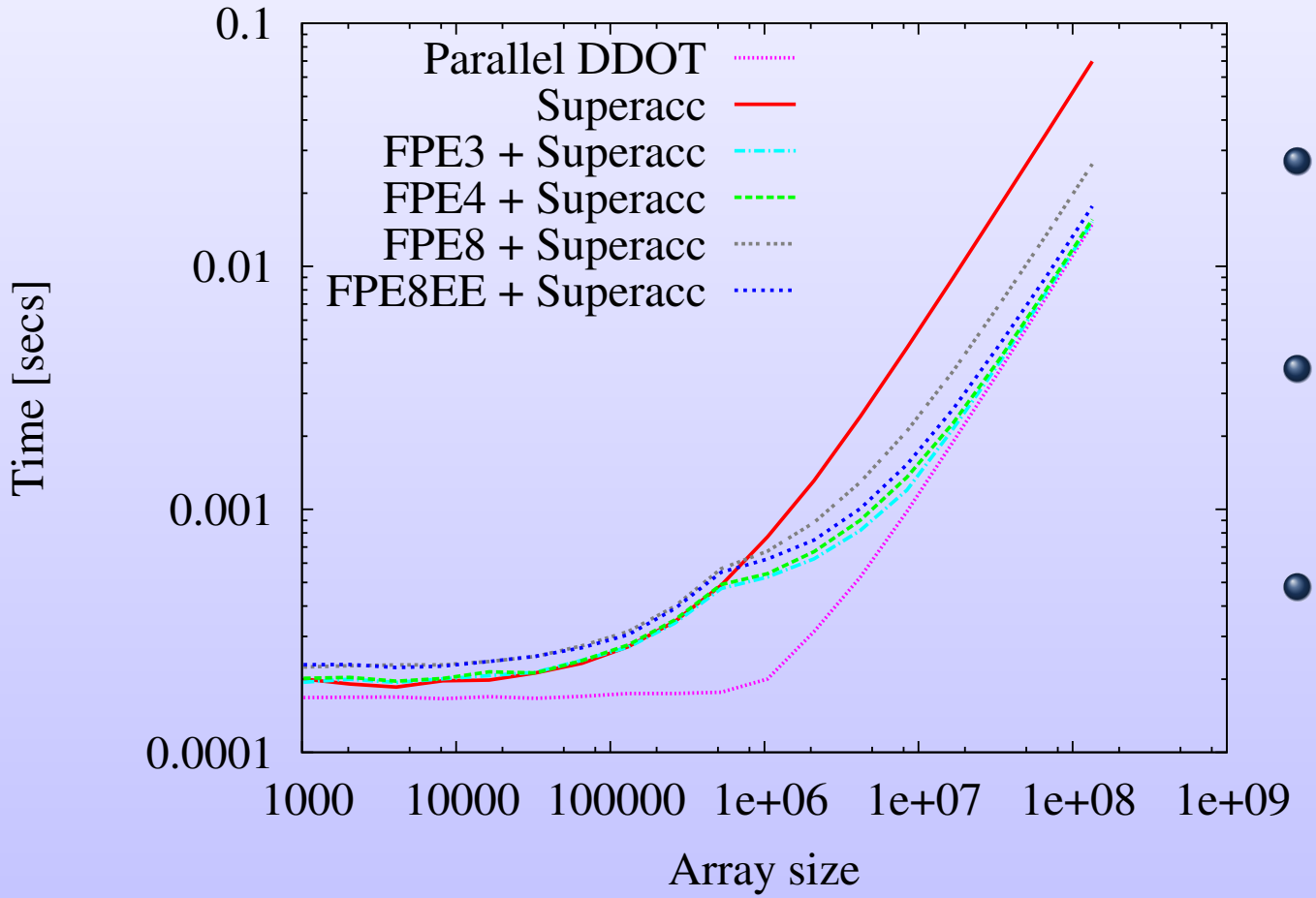
Data-Dependent Performance on NVIDIA Tesla K20c



Parallel Dot Product

Performance Scaling on NVIDIA Tesla K20c

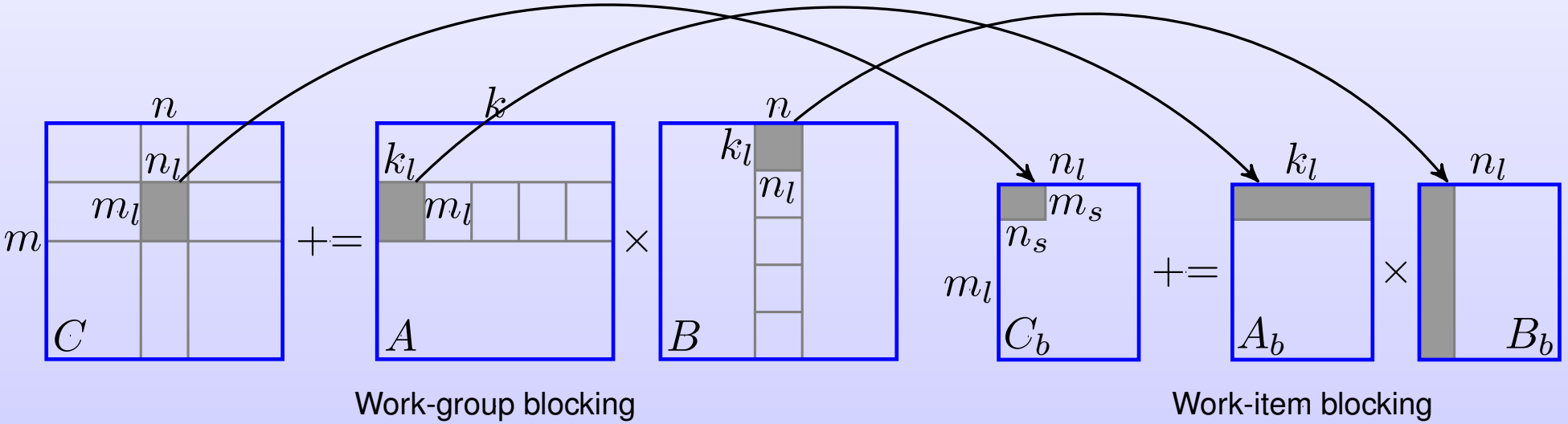
DDOT: $\alpha := x^T y = \sum_i^N x_i y_i$



- Based on TwoProduct and Reproducible Summation
- TwoProduct(a, b)
  1: $r \leftarrow a * b$
  2: $s \leftarrow fma(a, b, -r)$
- $fma(a, b, c) = a * b + c$

Matrix-Matrix Multiplication

GEMM (General matrix multiplication): $C := \alpha AB + \beta C$

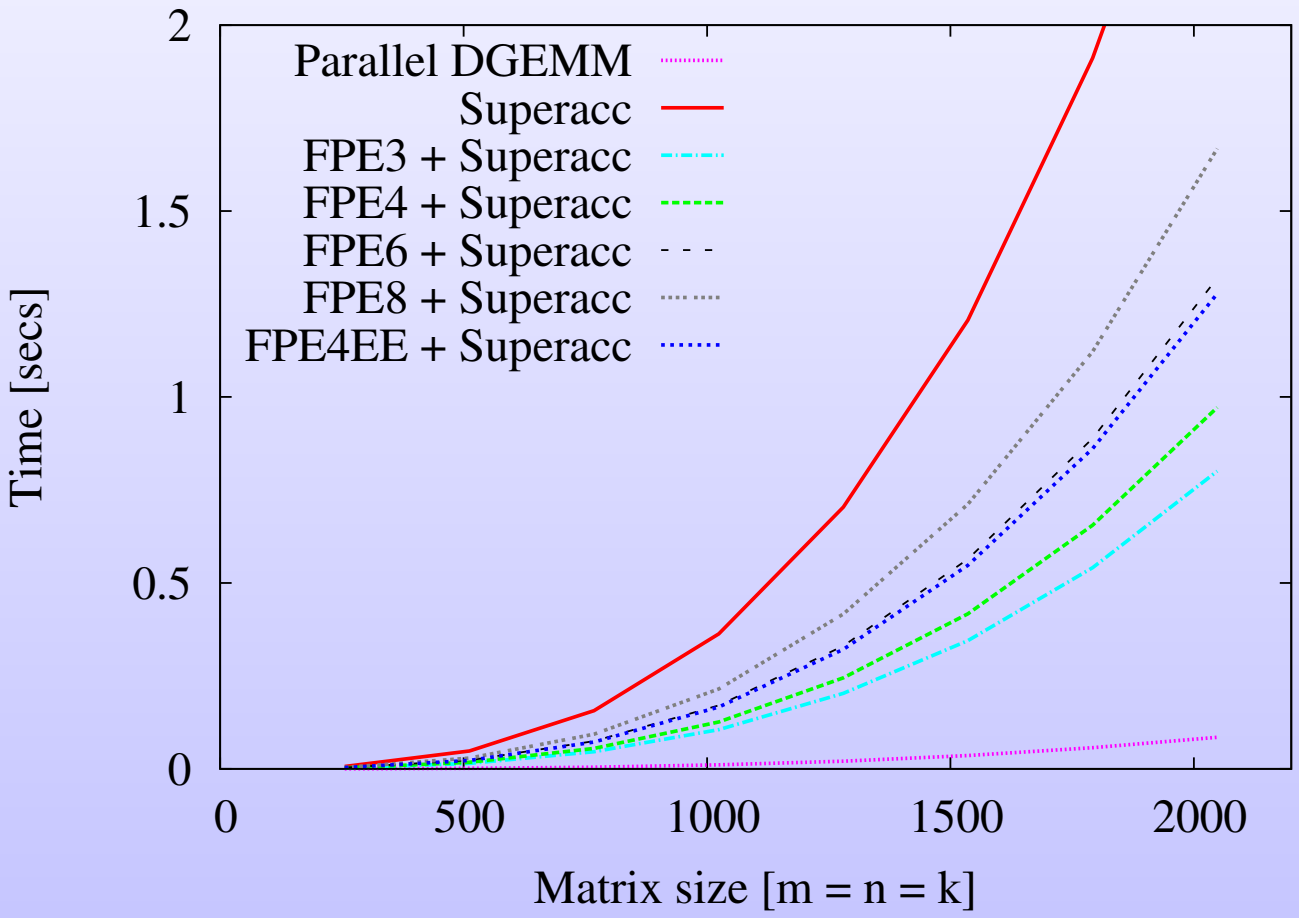


Partitioning of matrix-matrix multiplication

Parallel Matrix Product

Performance Scaling on NVIDIA Tesla K20c

DGEMM: $C := \alpha AB + \beta C$



- Extensive usage of memory → lower performance

The Proposed Multi-Level Algorithm

- Computes the results with no errors due to rounding
- Provides bit-wise identical reproducibility, regardless of
  - Data permutation, data assignment
  - Thread scheduling, etc.

The Proposed Multi-Level Algorithm

- Computes the results with no errors due to rounding
- Provides bit-wise identical reproducibility, regardless of
  - Data permutation, data assignment
  - Thread scheduling, etc.
- Is efficient – delivers comparable performance to the standard parallel summation and dot product
- Scales perfectly with the increase of the problem size or the number of cores

Conclusions

The Proposed Multi-Level Algorithm

- Computes the results with no errors due to rounding
- Provides bit-wise identical reproducibility, regardless of
  - Data permutation, data assignment
  - Thread scheduling, etc.
- Is efficient – delivers comparable performance to the standard parallel summation and dot product
- Scales perfectly with the increase of the problem size or the number of cores
- The GEMM performance needs to be enhanced

Future Work

- ExGEMM on Intel Phi and Intel CPUs
- The algorithms are suitable for large scale systems (ExaScale) with one more reduction step between nodes

ExBLAS – **Ex**act **BLAS**

- ExBLAS-1: `ExSCAL`, `ExDOT`, `ExAXPY`, ...
- ExBLAS-2: `ExGER`, `ExGEMV`, `ExTRSV`, ...
- ExBLAS-3: `ExGEMM`, `ExTRMM`, `ExSYR2K`, ...

Acknowledgement

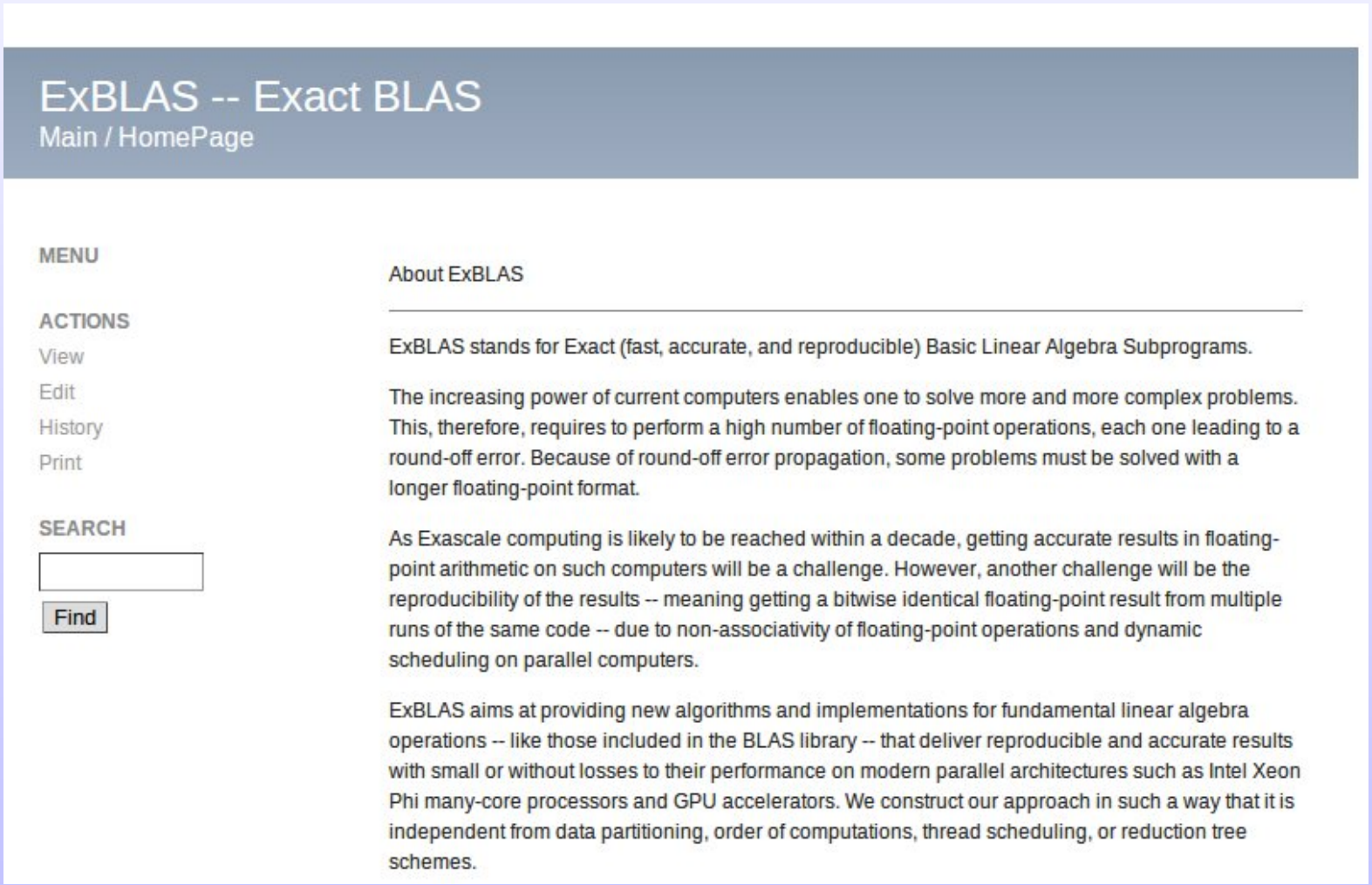
Thank you for your attention!

→ This work undertaken (partially) in the framework of CALSIMLAB is supported by the public grant ANR-11-LABX-0037-01 overseen by the French National Research Agency (ANR) as part of the “Investissements d’Avenir” program (reference: ANR-11-IDEX-0004-02)

→ This work was granted access to the HPC resources of The Institute for scientific Computing and Simulation financed by Region Île-de-France and the project Equip@Meso (reference ANR-10-EQPX-29-01) overseen by the French National Research Agency (ANR) as part of the “Investissements d’Avenir” program

On the Web

URL: `https://exblas.lip6.fr`



ExBLAS -- Exact BLAS

Main / HomePage

MENU

ACTIONS

View

Edit

History

Print

SEARCH

Find

About ExBLAS

ExBLAS stands for Exact (fast, accurate, and reproducible) Basic Linear Algebra Subprograms.

The increasing power of current computers enables one to solve more and more complex problems. This, therefore, requires to perform a high number of floating-point operations, each one leading to a round-off error. Because of round-off error propagation, some problems must be solved with a longer floating-point format.

As Exascale computing is likely to be reached within a decade, getting accurate results in floating-point arithmetic on such computers will be a challenge. However, another challenge will be the reproducibility of the results -- meaning getting a bitwise identical floating-point result from multiple runs of the same code -- due to non-associativity of floating-point operations and dynamic scheduling on parallel computers.

ExBLAS aims at providing new algorithms and implementations for fundamental linear algebra operations -- like those included in the BLAS library -- that deliver reproducible and accurate results with small or without losses to their performance on modern parallel architectures such as Intel Xeon Phi many-core processors and GPU accelerators. We construct our approach in such a way that it is independent from data partitioning, order of computations, thread scheduling, or reduction tree schemes.