Reproducible and Accurate Matrix Multiplication in ExBLAS for High-Performance Computing

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

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

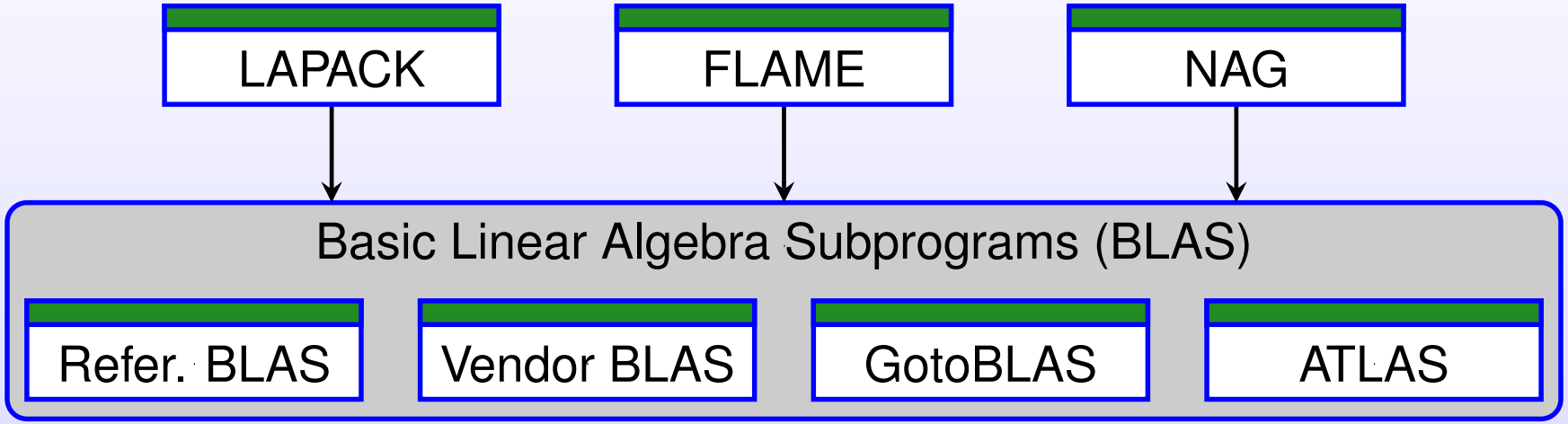
roman.iakymchuk@lip6.fr

SCAN 2014, 16th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics
Würzburg, Germany, September 21-26, 2014


SORBONNE UNIVERSITÉS
UPMC SORBONNE UNIVERSITÉS

ICS Institut du calcul et de la simulation
LIP6

Motivation



$\Downarrow$

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

Ultimate Goal

- To compute BLAS operations with floating-point numbers efficiently and with the best possible accuracy 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

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

Reproducibility at ExaScale (I)

- **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

Reproducibility at ExaScale (I)

- **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 (I)

- **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

Needs for Reproducibility

- Debugging
  - Look inside the code step-by-step and might need to rerun multiple times on the same input data
- Understanding the reliability of output
- Contractual reasons (for security, ...)
- etc.

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

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

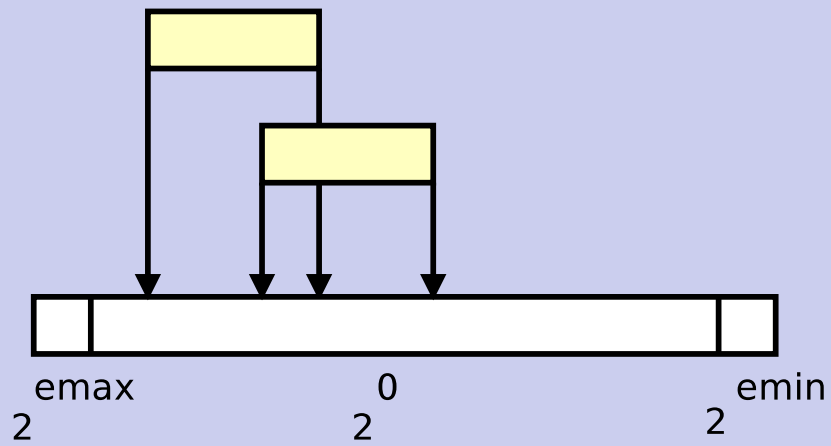
Our Approach

Algorithm 1 EFT of size 2 (Dekker and Knuth)

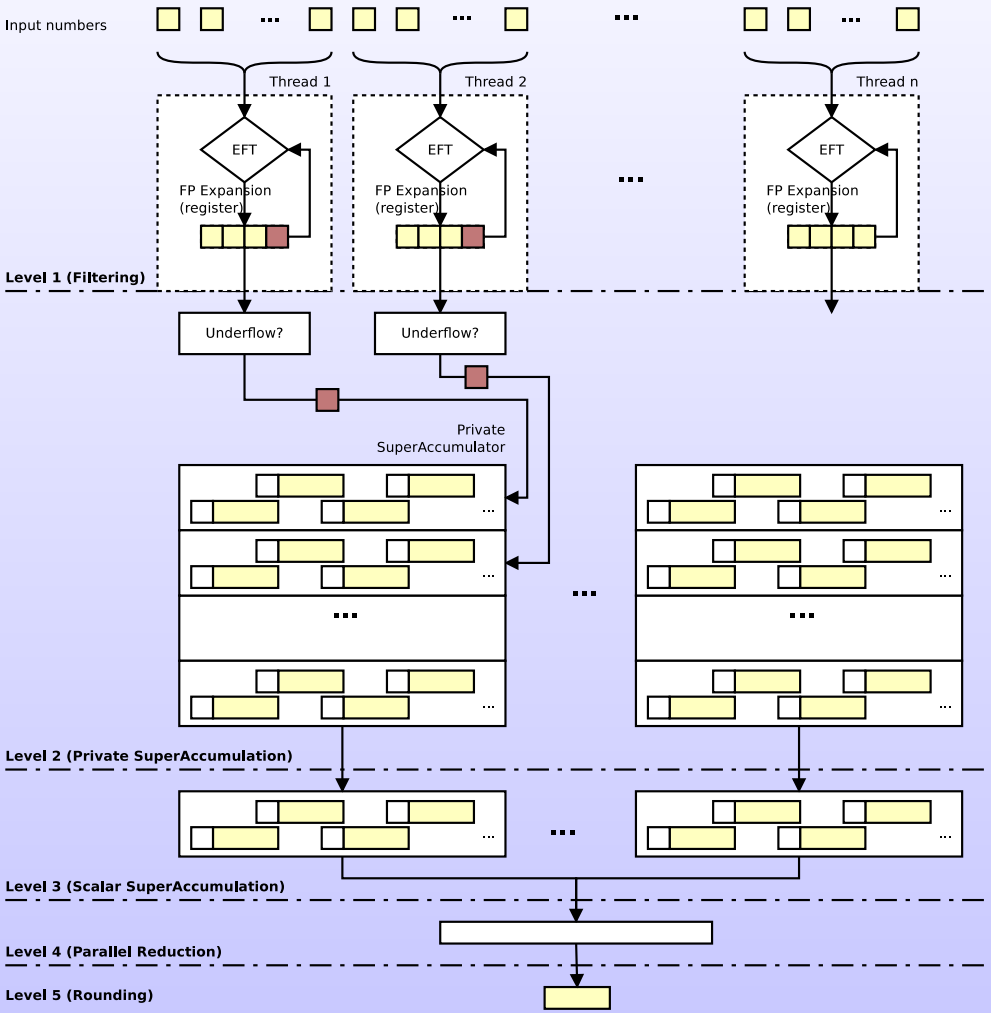
function$[r, s]$ = TwoSum(a, b)

1: $r \leftarrow a + b$
2: $z \leftarrow r - a$
3: $s \leftarrow (a - (r - z)) + (b - z)$

Kulisch long accumulator



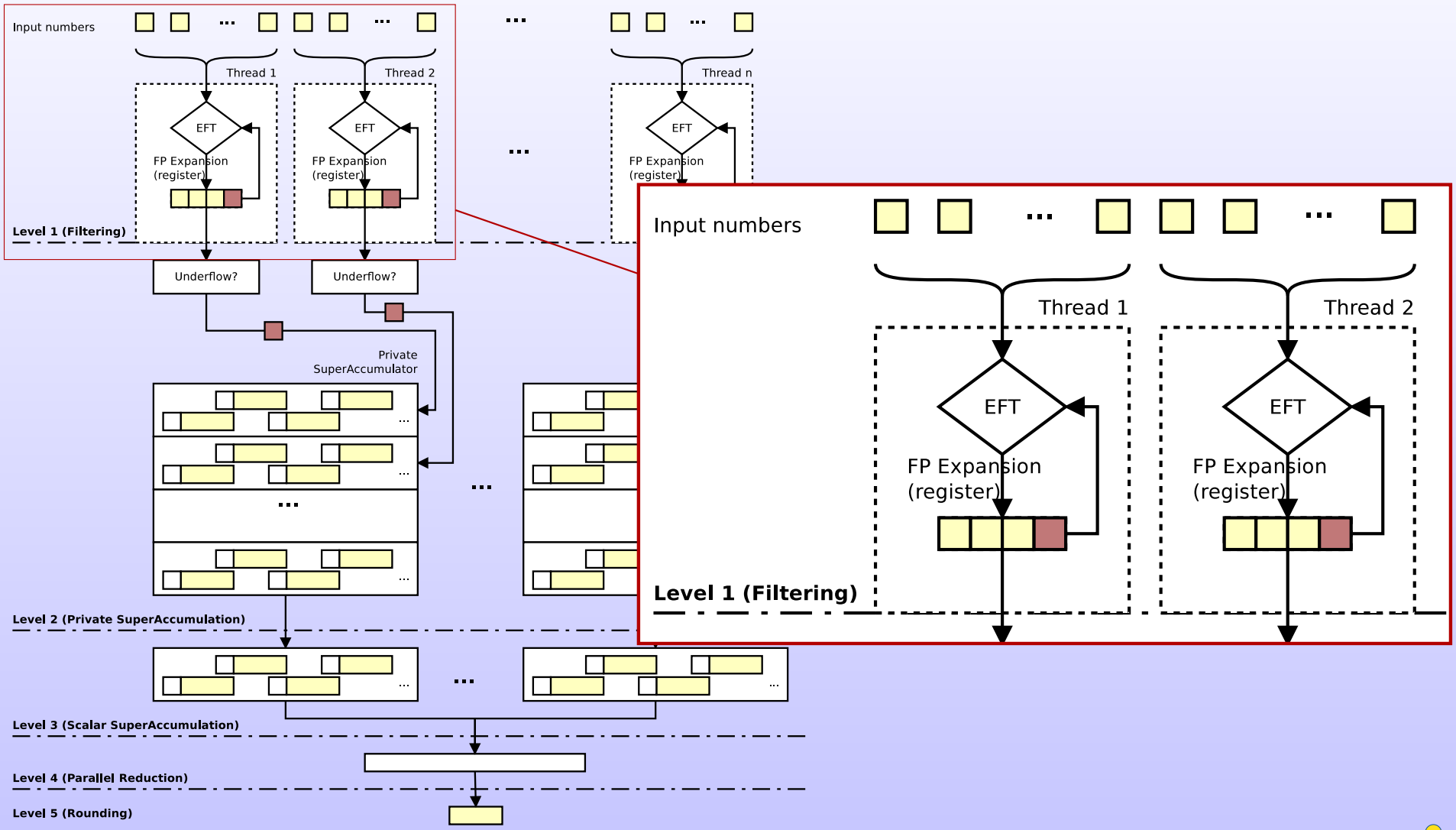
Multi-Level Reproducible Summation



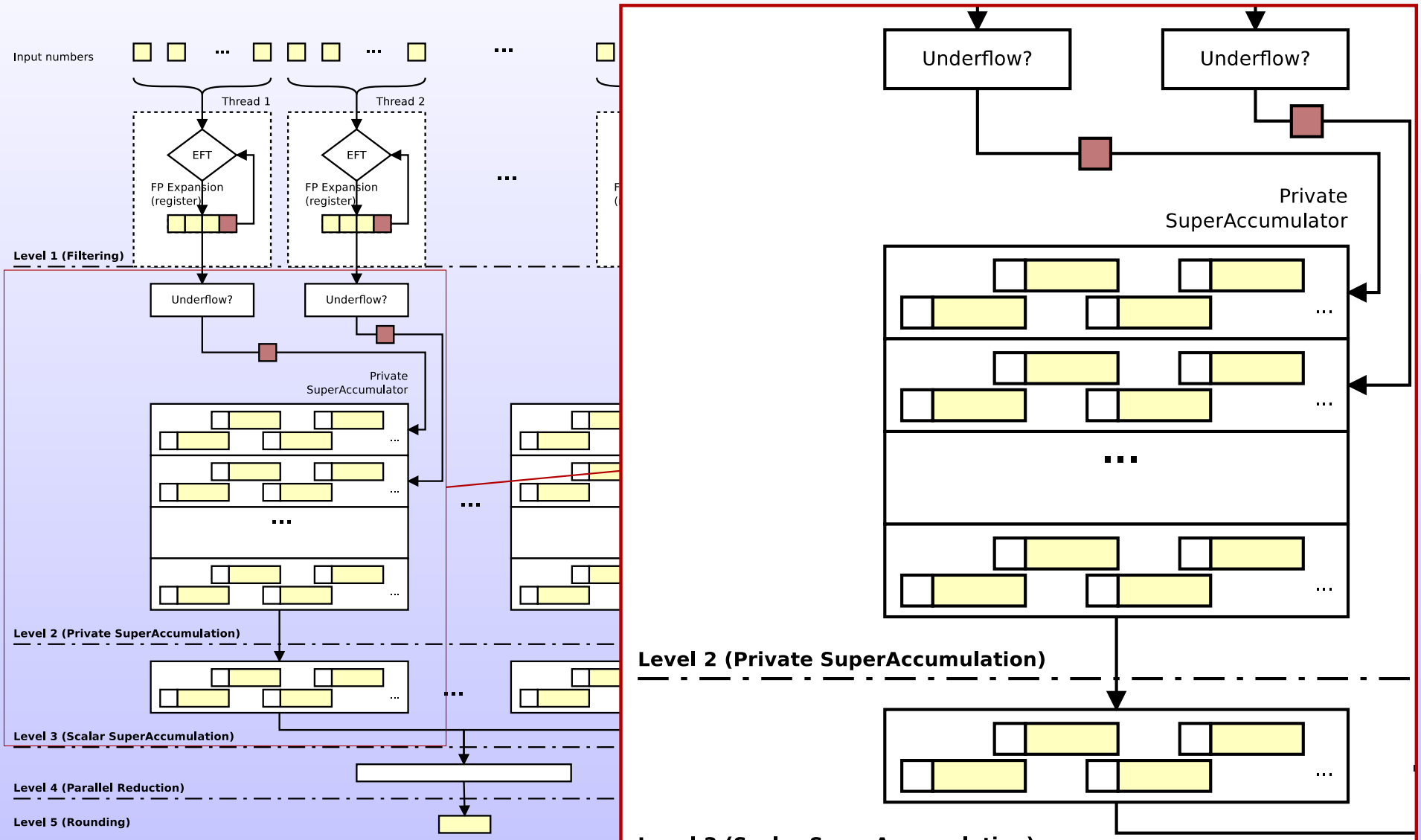
- Parallel algorithm with 5-levels
- Suitable for today's parallel architectures
- Based on FPE with EFT and Kulisch accumulator
- Guarantees "inf" precision

→ bit-wise reproductibility

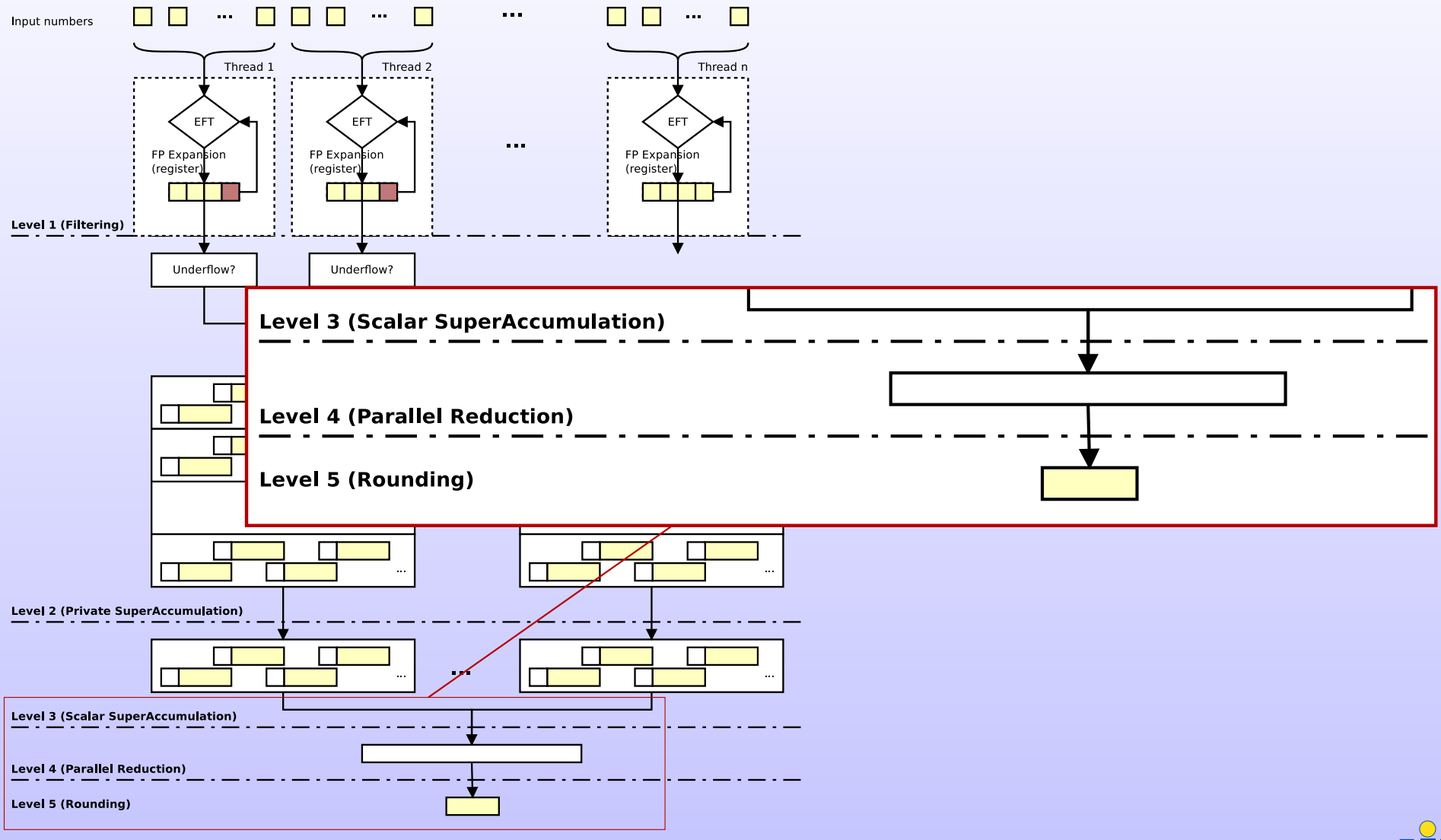
Level 1: Filtering



Level 2 and 3: Scalar Superaccumulator

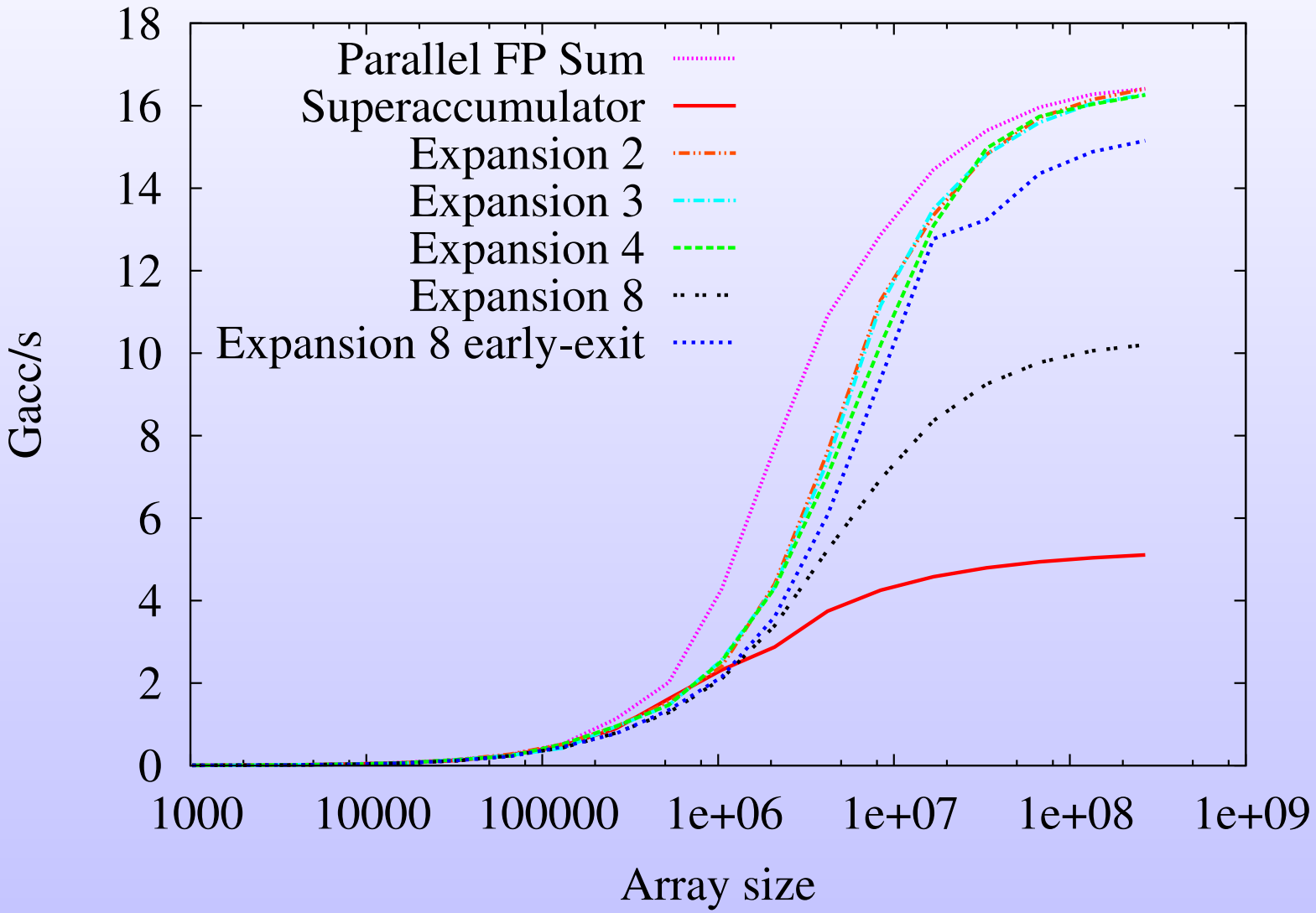


Level 4 and 5: Reduction and Rounding



Parallel Summation

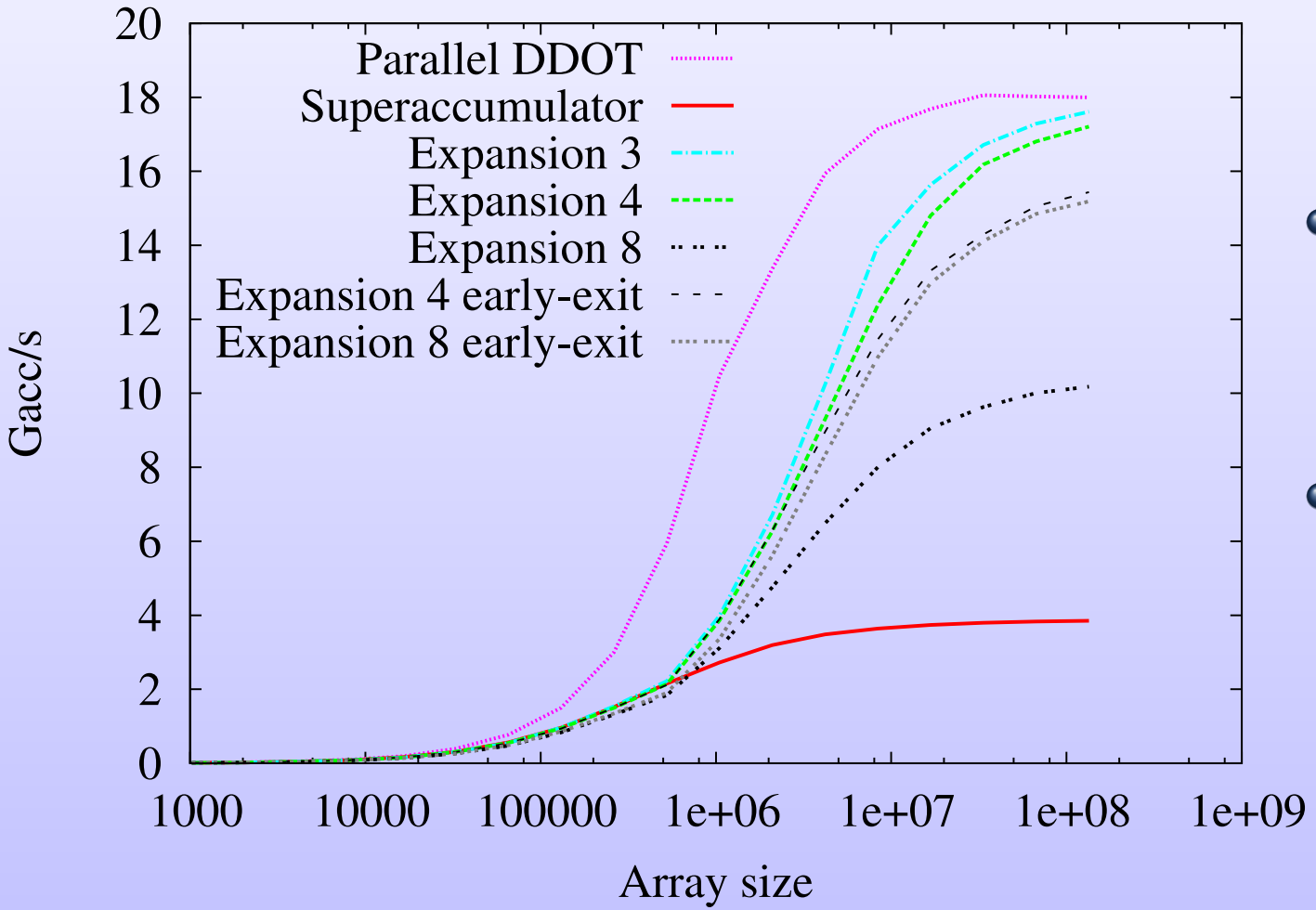
Performance Scaling on NVIDIA Tesla K20c



Parallel Dot Product

Performance Scaling on NVIDIA Tesla K20c

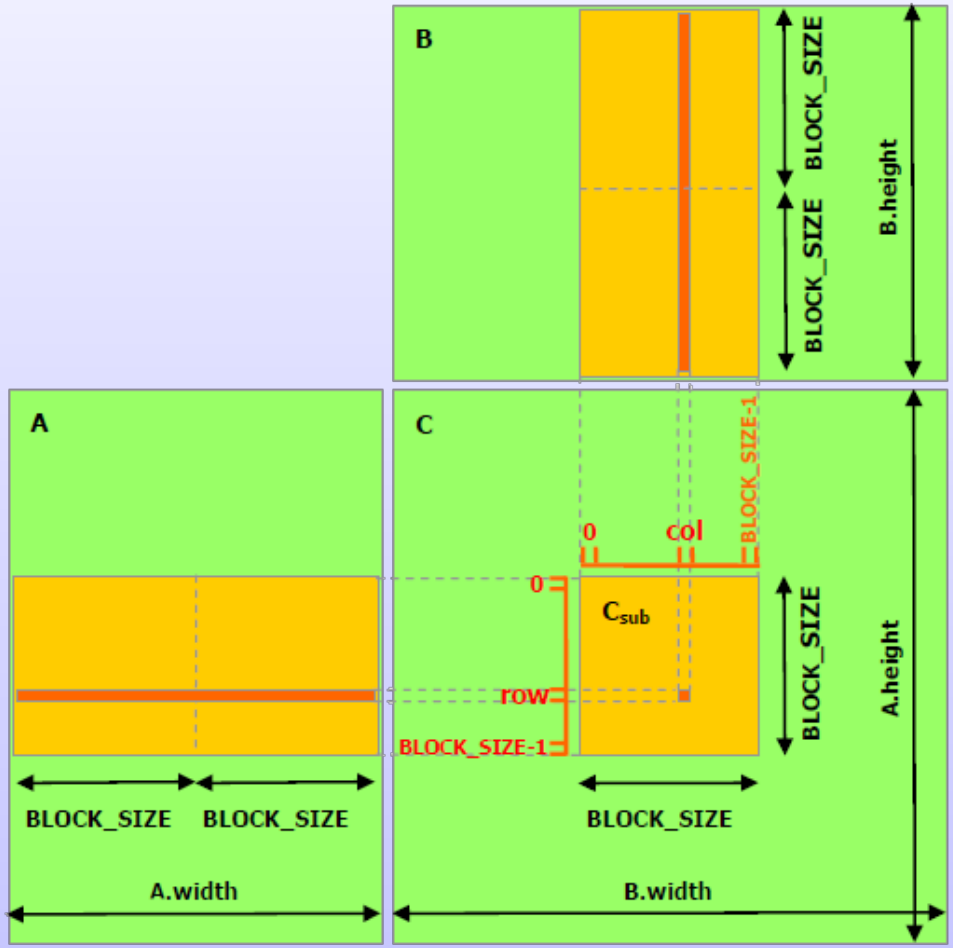
DDOT: $\alpha := x^T y = \sum_i^N x_i y_i$



- Based on TwoProduct and Reproducible Summation
- TwoProduct relies on FMA

Multi-Level Reproducible DGEMM

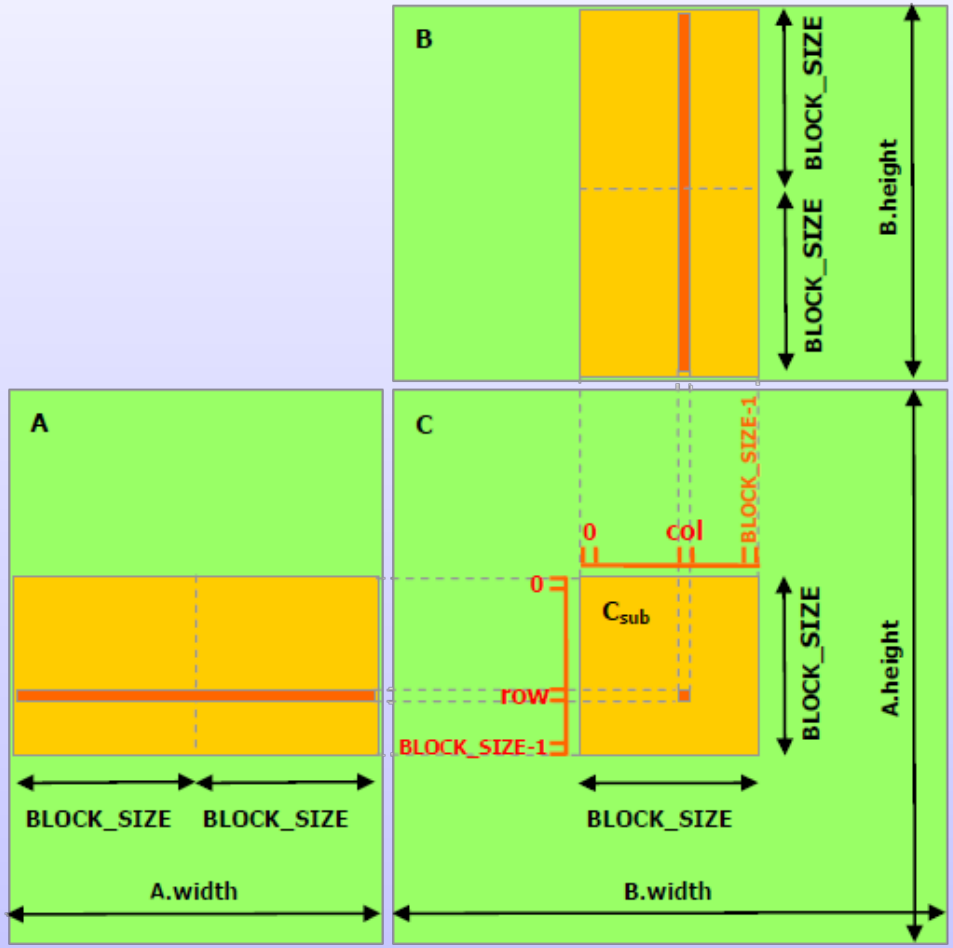
DGEMM: $C := \alpha AB + \beta C$



Source: CUDA C Programming Guide

Multi-Level Reproducible DGEMM

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



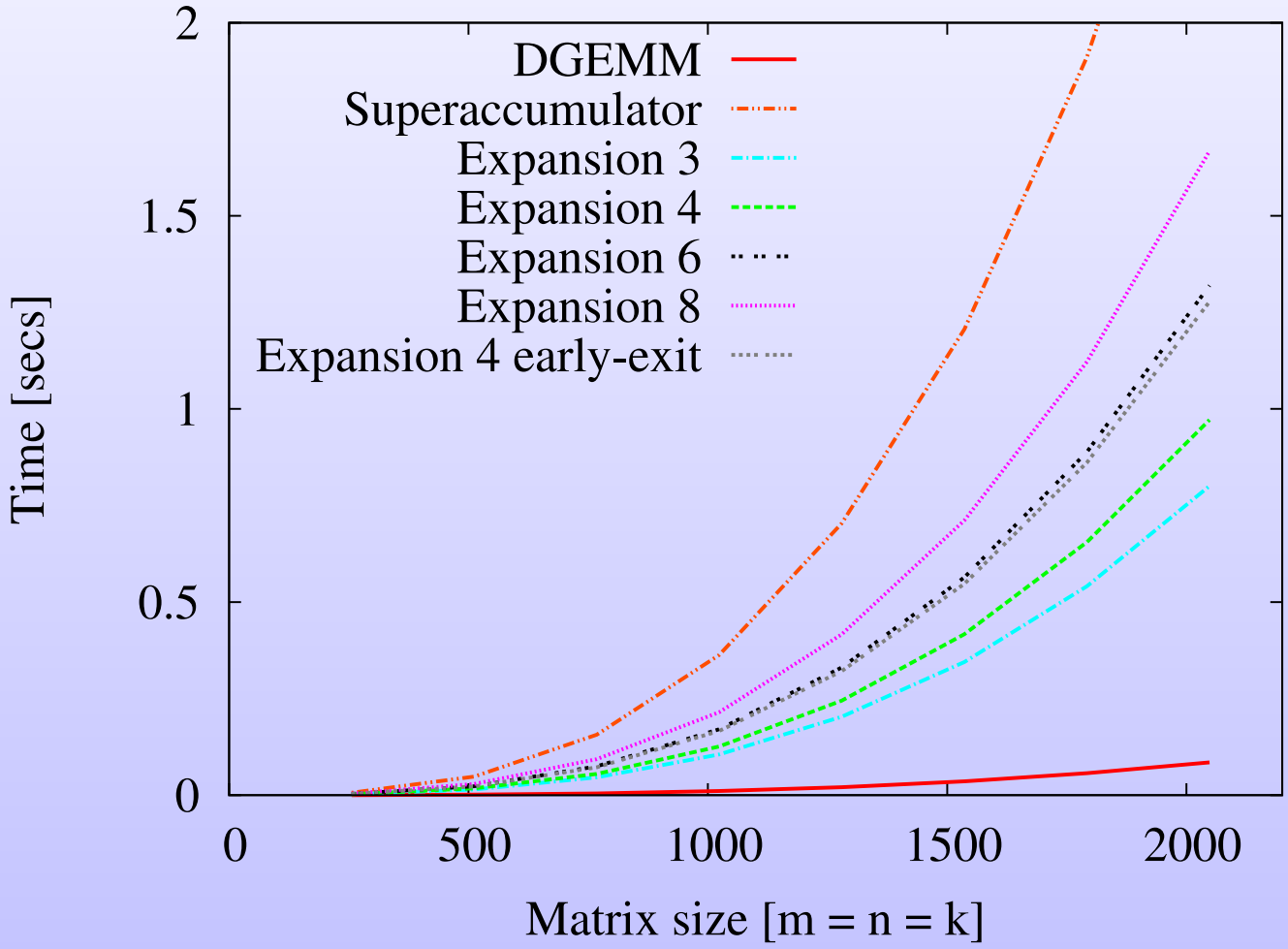
Source: CUDA C Programming Guide

- One FPE and Kulisch accumulator per thread
- Algorithm consists of 3 steps:
  - Filtering
  - Private SuperAccumulation
  - Rouding
- Each thread computes multiple elements of matrix C to reduce memory pressure

Parallel Matrix Multiplication

Performance Scaling on NVIDIA Tesla K20c

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



- $12dn^3$ Flops more, d is size of FPE
- Up to $76n^3$ more memory usage

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
- For DGEMM the performance needs to be enhanced

Future Work

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

→ Parallelization with MPI

ExBLAS – **Ex**act **BLAS**

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

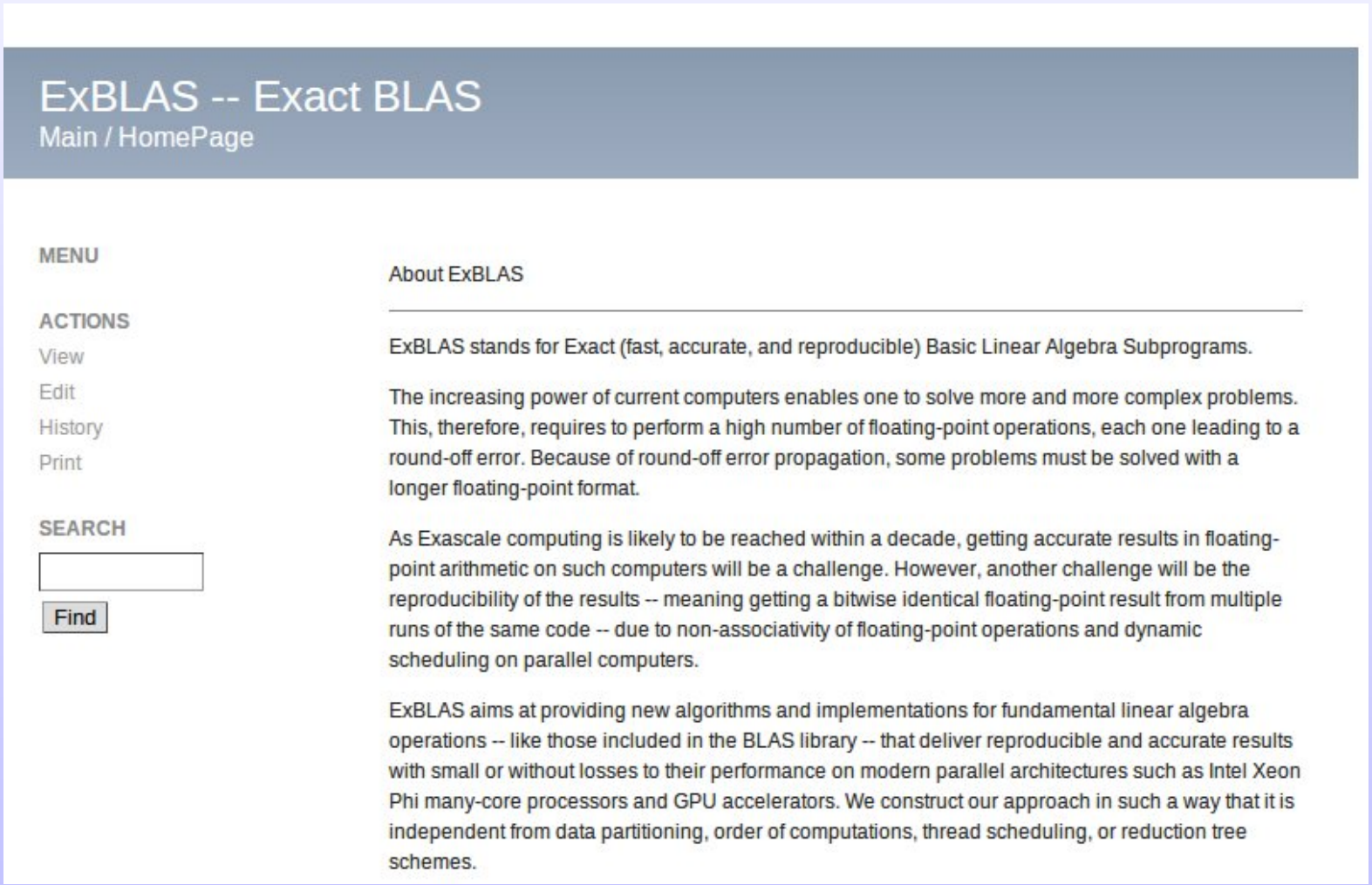
Acknowledgement

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).

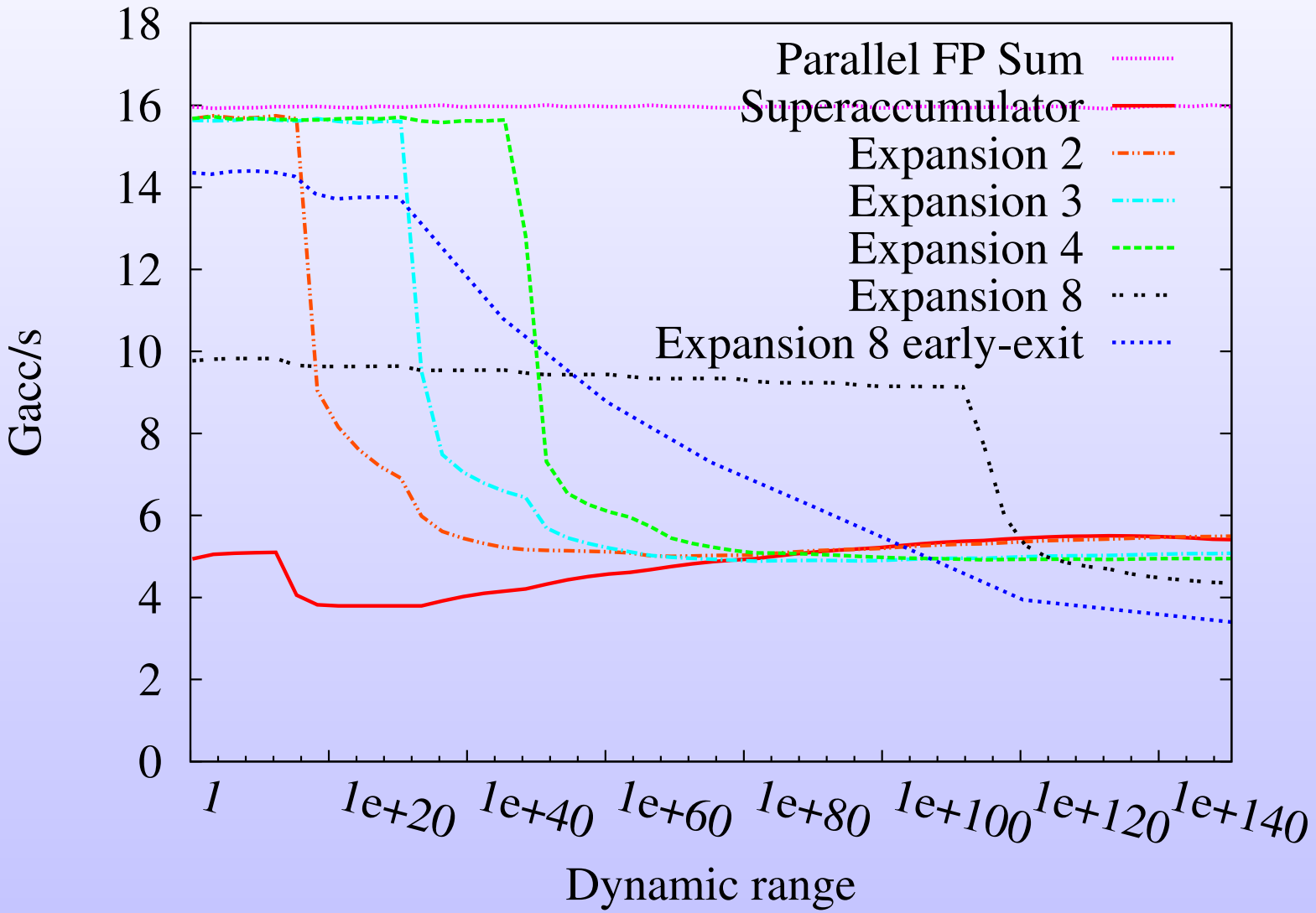
On the Web

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



Parallel Summation

Data-Dependent Performance on NVIDIA Tesla K20c



References

S. Collange, D. Defour, S. Graillat, and R. Iakymchuk

Numerical Reproducibility for the Summation Problem on Multi- and Many-Core Architectures. Submitted to the Parallel Computing Journal

S. Collange, D. Defour, S. Graillat, and R. Iakymchuk

Full-Speed Deterministic Bit-Accurate Parallel Floating-Point Summation on Multi- and Many-Core Architectures, Tech report, Feb, 2014. HAL-ID: hal-00949355

J. Demmel and H.D. Hguyen

Fast Reproducible Floating-Point Summation. Proceedings of the 21st IEEE Symposium on Computer Arithmetic, Austin, Texas, USA, 2013

J. Demmel and H.D. Hguyen

Parallel Reproducible Summation. To appear in IEEE Transactions on Computers, Special Section on Computer Arithmetic 2014

J. Demmel and H.D. Hguyen

Numerical reproducibility and accuracy at ExaScale (invited talk), the 21st IEEE Symposium on Computer Arithmetic, Austin, Texas, USA, 2013

A. Arteaga, O. Fuhrer, and T.Hoefler

Designing Bit-Reproducible Portable High-Performance Applications. Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2014

G. Michelogiannakis, X.S. Li, D.H. Bailey, and J. Shalf

Extending Summation Precision for Network Reduction Operations. Symposium on Computer Architecture and High Performance Computing, Porto de Galinhas, Brazil, 2013