Reproducible level 1 BLAS on massively parallel systems

Chemseddine Chohra
Philippe Langlois and David Parello

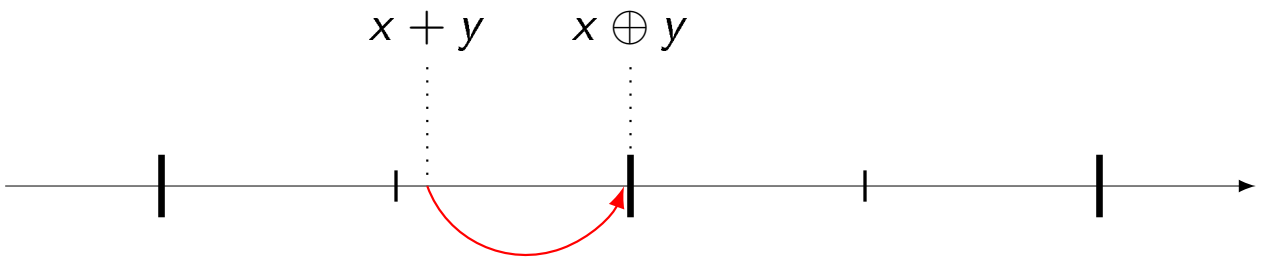
University of Perpignan Via Domitia

RAIM - April 07^{th} 2015

Introduction and problematic

Limited machine precision

- Using floating point numbers as approximation.
- $x \longrightarrow X = fl(x)$ if $x \notin F$ or x if $x \in F$.
- $X + Y \neq X \oplus Y = fl(X + Y)$.
- IEEE-754 standard defines several rounding modes.



Introduction and problematic

Non-associativity of addition

- $A \oplus (B \oplus C) \neq (A \oplus B) \oplus C$.
- Catastrophic cancelation : $M = 2^{53}$; $0 = -M \oplus (M \oplus 1) \neq (-M \oplus M) \oplus 1 = 1$.

Non-reproducibility of summation

- For a sum $(\sum_{i=1}^{n} X_i)$, the final result depends on the order of the computations.
- Why should operations order be different.
  - Different micro-architectures (different instruction sets).
  - Different compilers (or even versions of compilers).
  - Data alignment.
  - Dynamic scheduling.
  - Non-deterministic reduction.

Introduction and problematic

Is numerical reproducibility really important ?

- Important for debugging.
- Important for validating results.
- Reproducibility : One of top 10 exascale research challenges (U.S. Department of Energy [DOE], 2014).
  - 10^{18} flop/s.
  - Millions of cores.

"Reproducibility will be expensive if not impossible to achieve on exascale"

How to fix the numerical reproducibility problem ?

Parallel libraries solutions

- Static scheduling.
- Deterministic reduction.
  - Available on OpenMP and TBB.
  - Recommended in MPI 3.0 standard.
- MKL (Conditional Numerical Reproducibility).

Algorithmic solutions

- Deterministic error (Demmel and Nguyen, 2013).
  - ReprodSum.
  - FastReprodSum.
  - OneReduction.
- Enhanced precision.
  - Higher precision (quadruple precision for instance).
    - Improve accuracy and consistency (Villa and al., 2009).
    - Reproducibility is not always guaranteed.
  - Correctly rounded arithmetic.
    - FP expansions + Super accumulators (S. Collange and al., 2014).

Our aim

Guarantee the numerical reproducibility for BLAS (Basic Linear Algebra Subroutines)

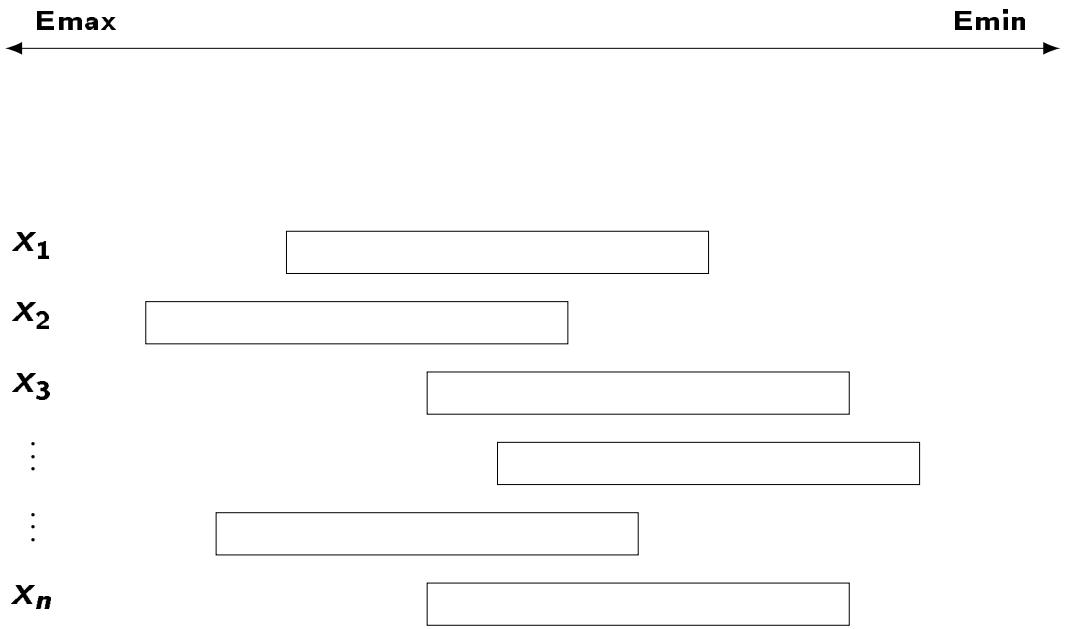
- Level 1 : max, min, scal, axpy, norm, asum, dot.
- dot can be transformed to a sum $\sum_{i=1}^{n} X_i \cdot Y_i = \sum_{i=1}^{2n} Z_i$.

Compute an accurate sum

- Several algorithms available.
- Is the cost acceptable ?

Table of contents

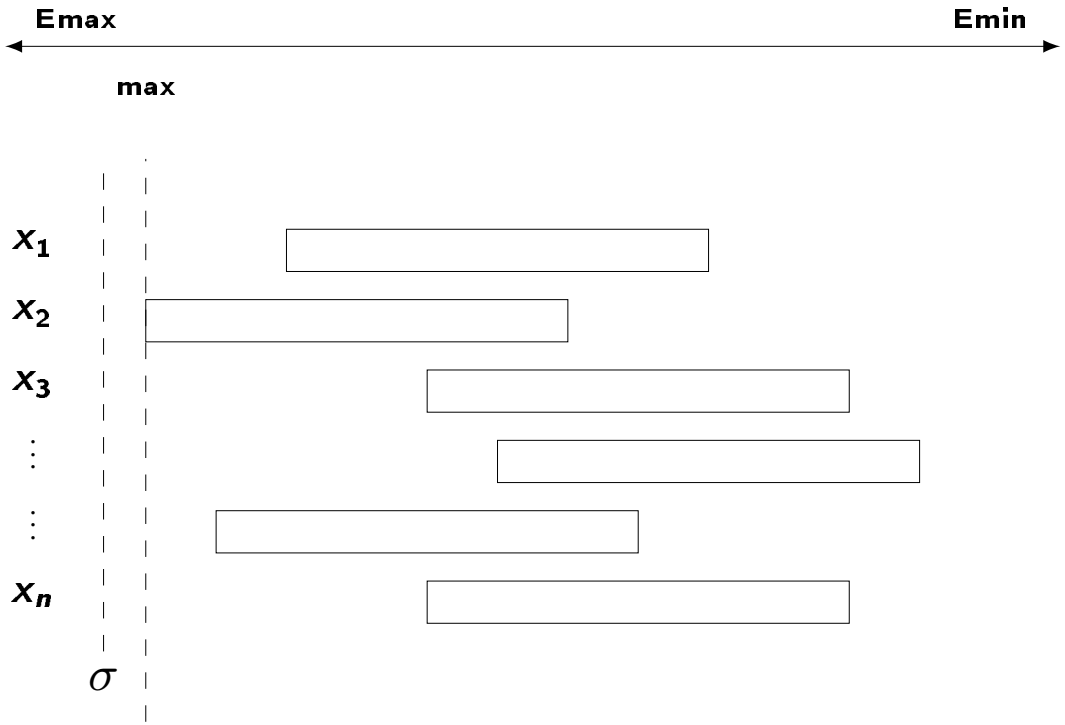
ReprodSum and FastReprodSum (Demmel and Nguyen, 2013)



Steps for sequential

- Compute max.

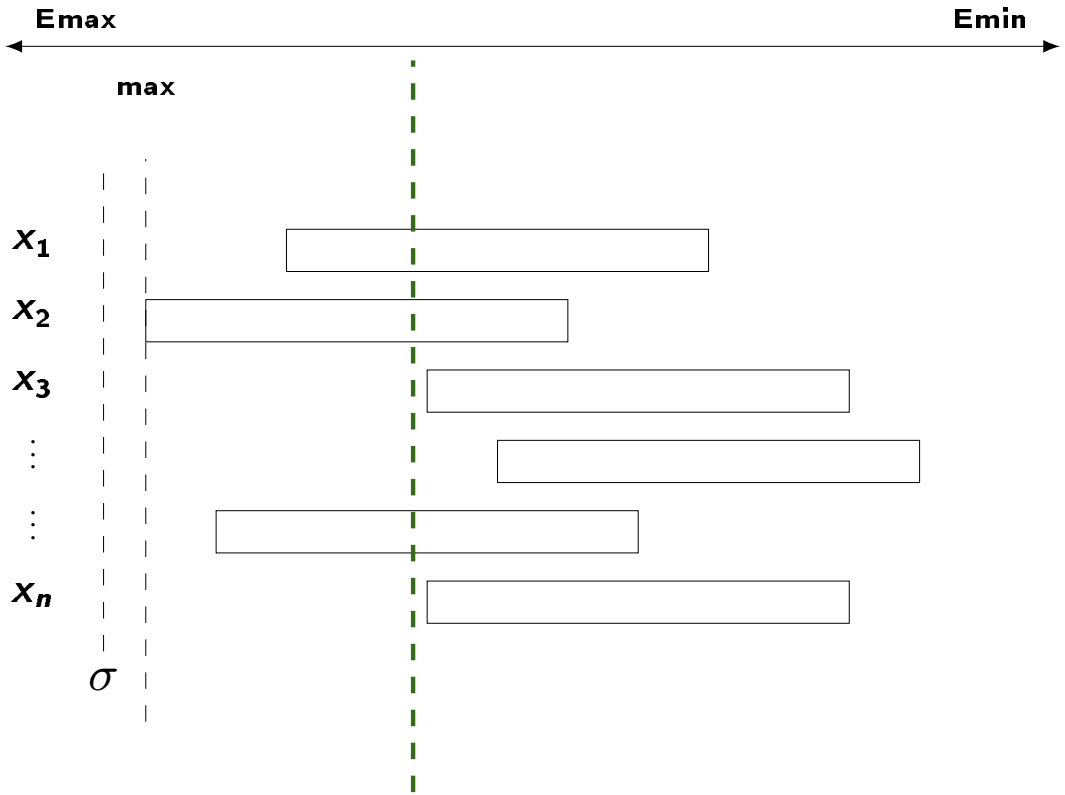
ReprodSum and FastReprodSum (Demmel and Nguyen, 2013)



Steps for sequential

- Compute max.

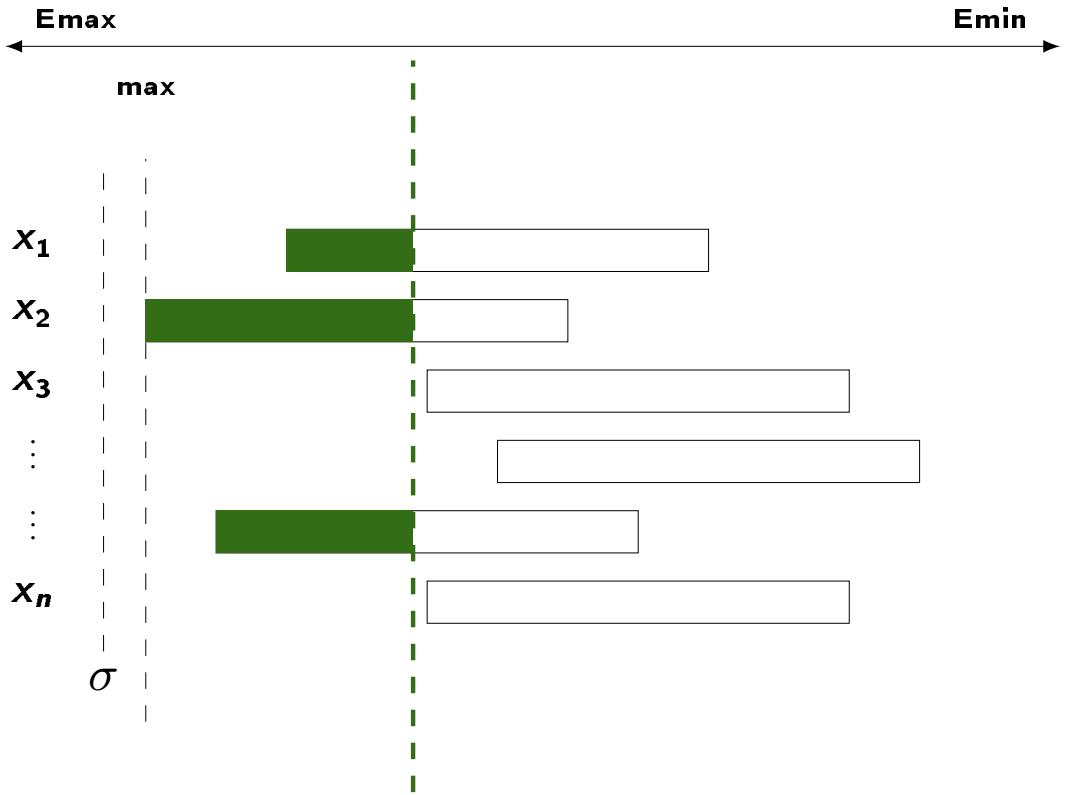
ReprodSum and FastReprodSum (Demmel and Nguyen, 2013)



Steps for sequential

- Compute max.

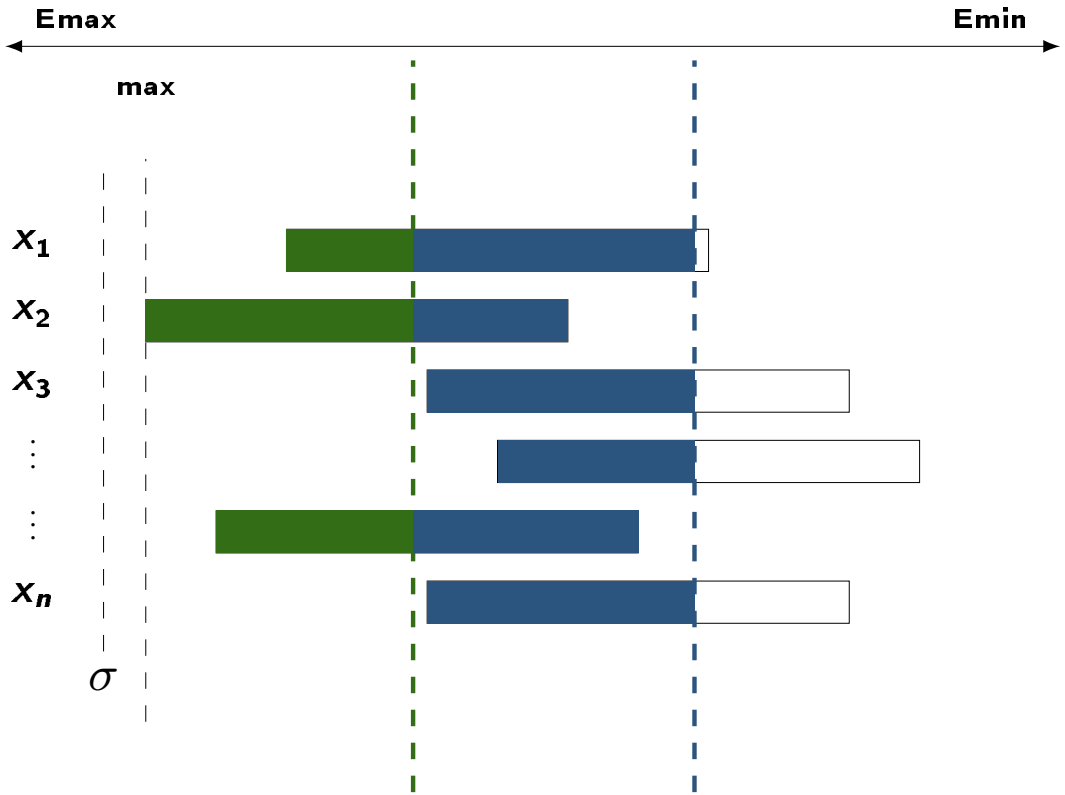
ReprodSum and FastReprodSum (Demmel and Nguyen, 2013)



Steps for sequential

- Compute max.
- Sum each fold.

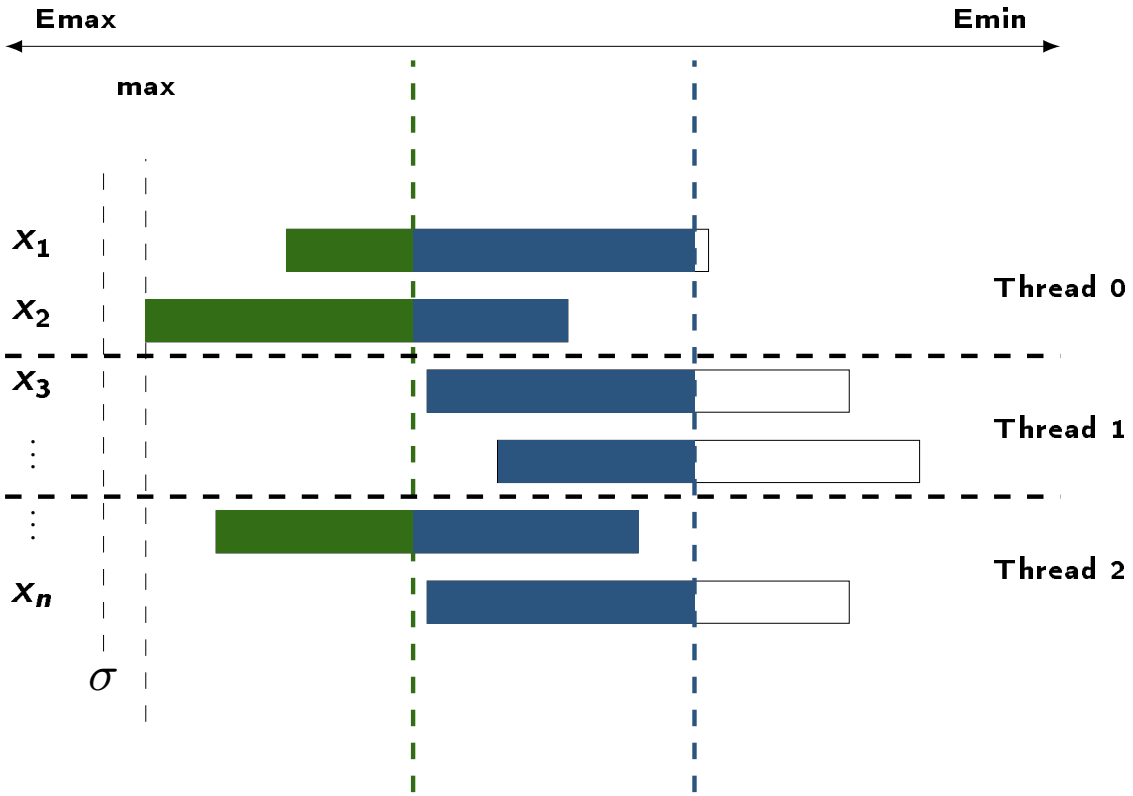
ReprodSum and FastReprodSum (Demmel and Nguyen, 2013)



Steps for sequential

- Compute max.
- Sum each fold.

ReprodSum and FastReprodSum (Demmel and Nguyen, 2013)



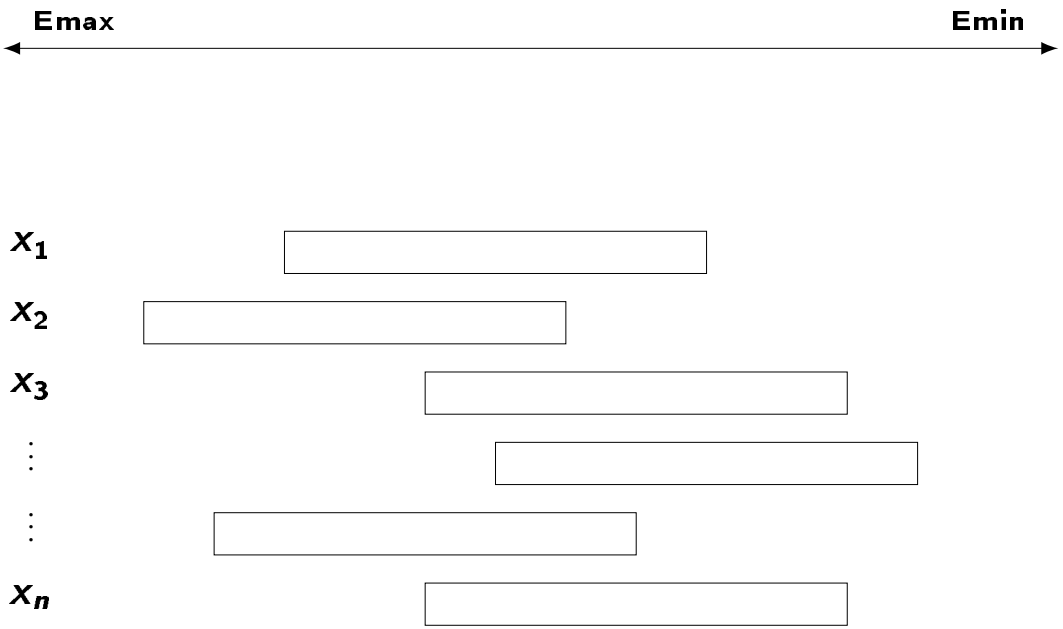
Steps for sequential

- Compute max.
- Sum each fold.

Steps for parallel

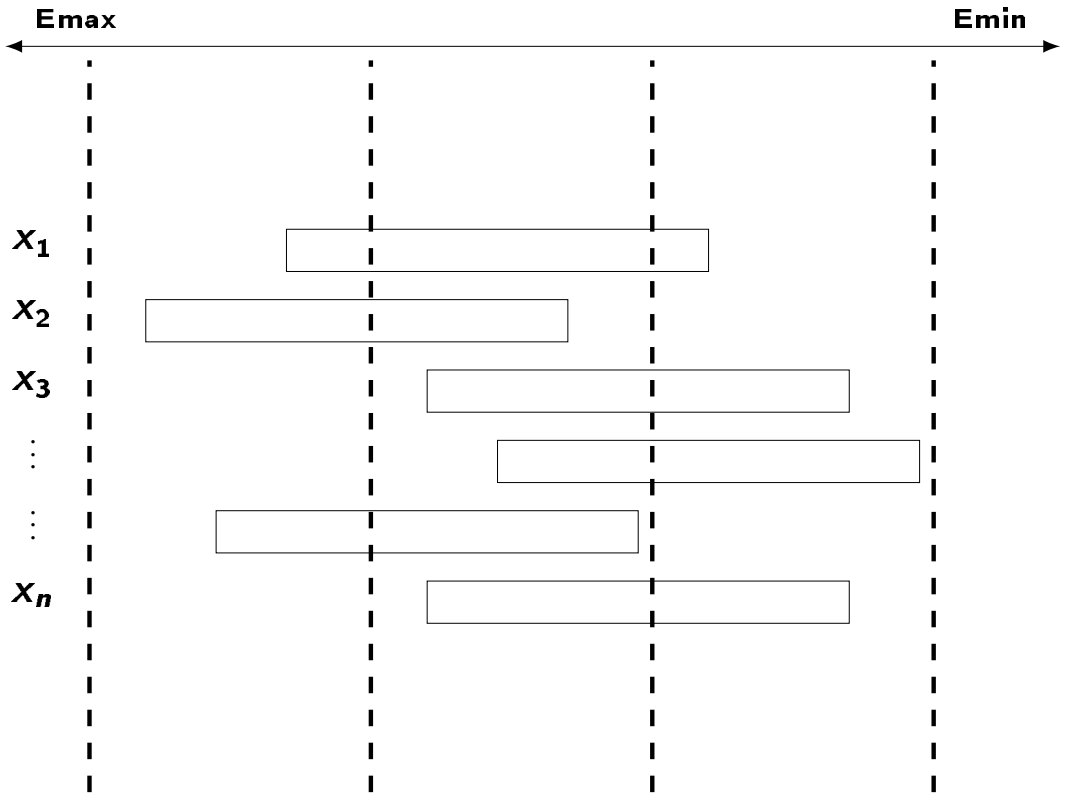
- Max (Compute).
- Max (Reduce).
- Sum (Compute).
- Sum (Reduce).

OneReduction (Demmel and Nguyen, 2013)



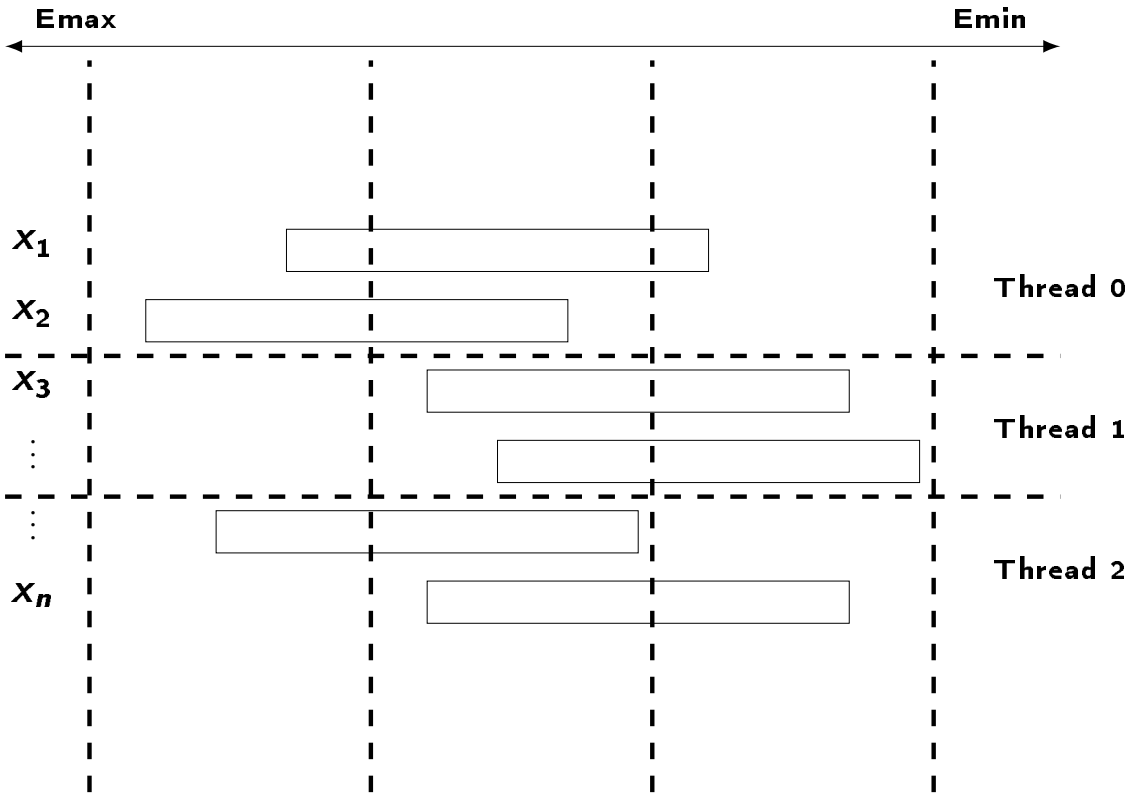
Steps for parallel

OneReduction (Demmel and Nguyen, 2013)



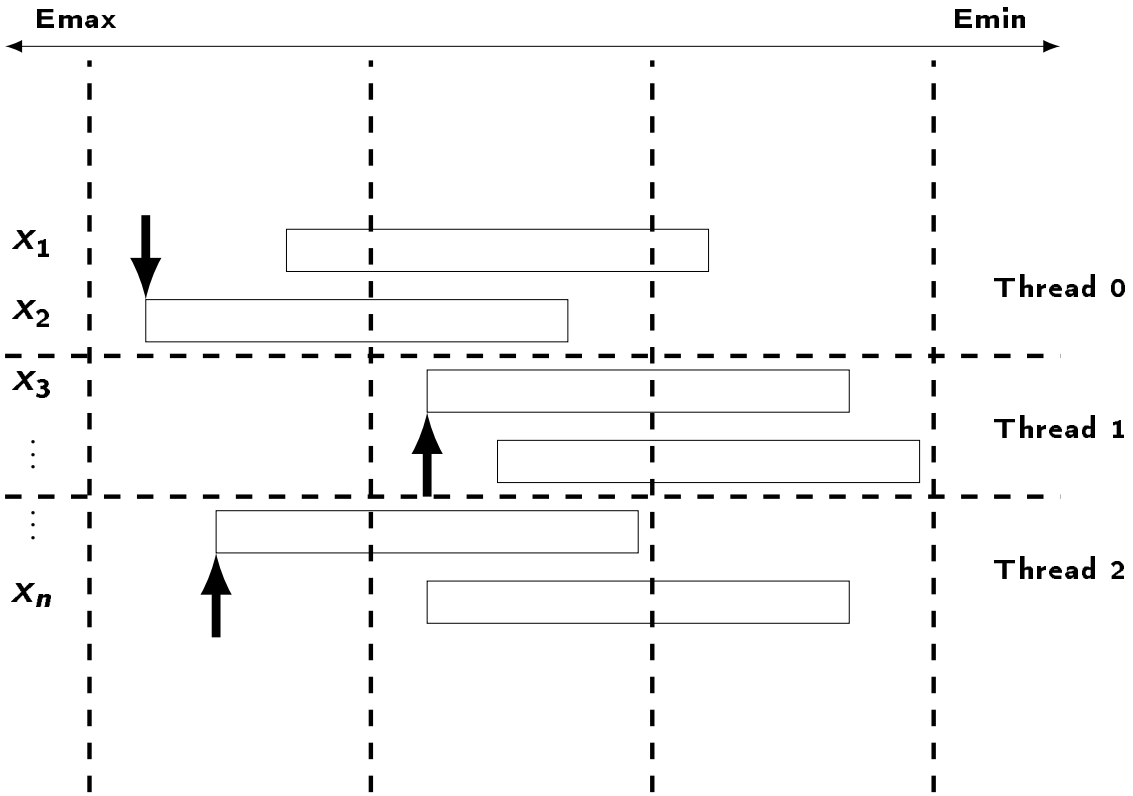
Steps for parallel

OneReduction (Demmel and Nguyen, 2013)



Steps for parallel

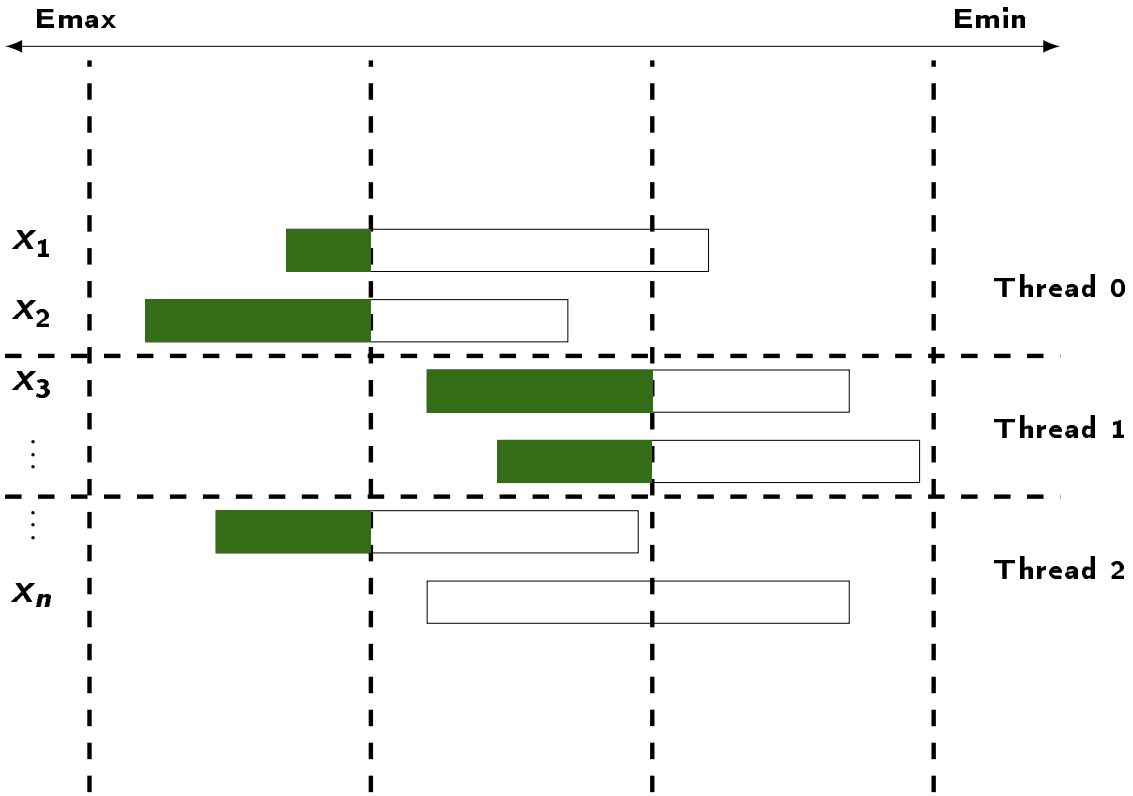
OneReduction (Demmel and Nguyen, 2013)



Steps for parallel

- Max (Compute).

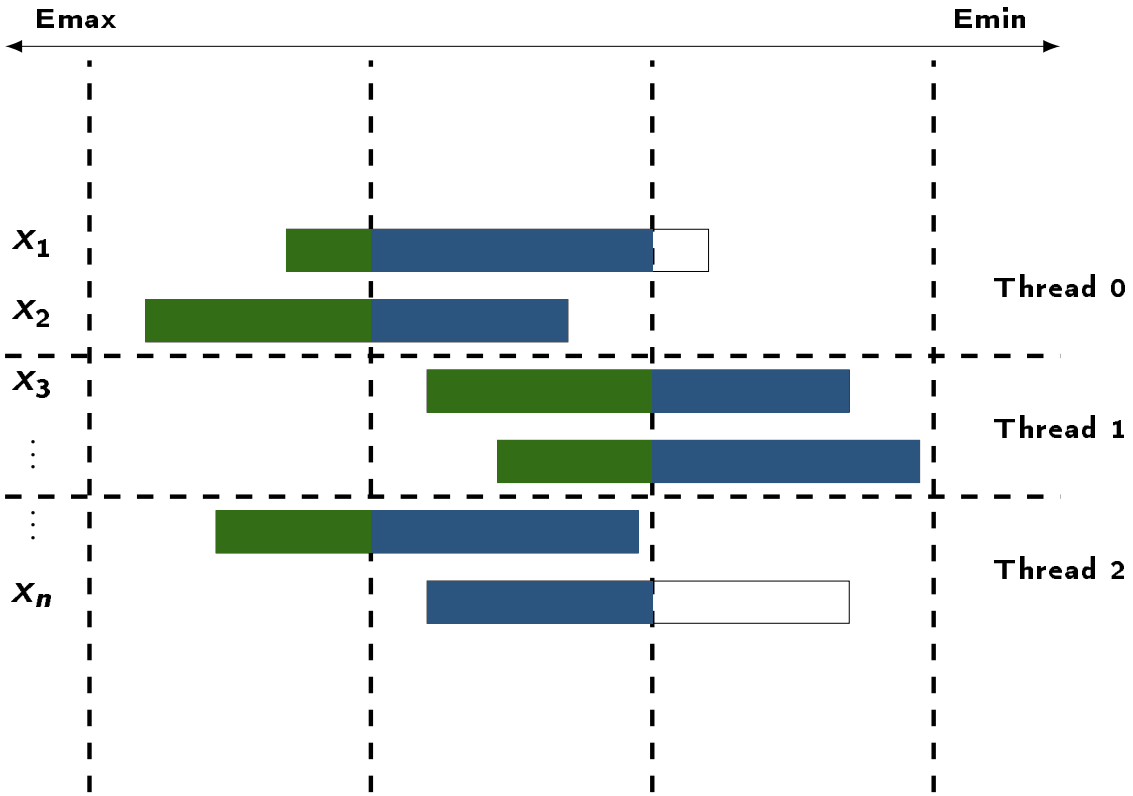
OneReduction (Demmel and Nguyen, 2013)



Steps for parallel

- Max (Compute).
- Sum (Compute).

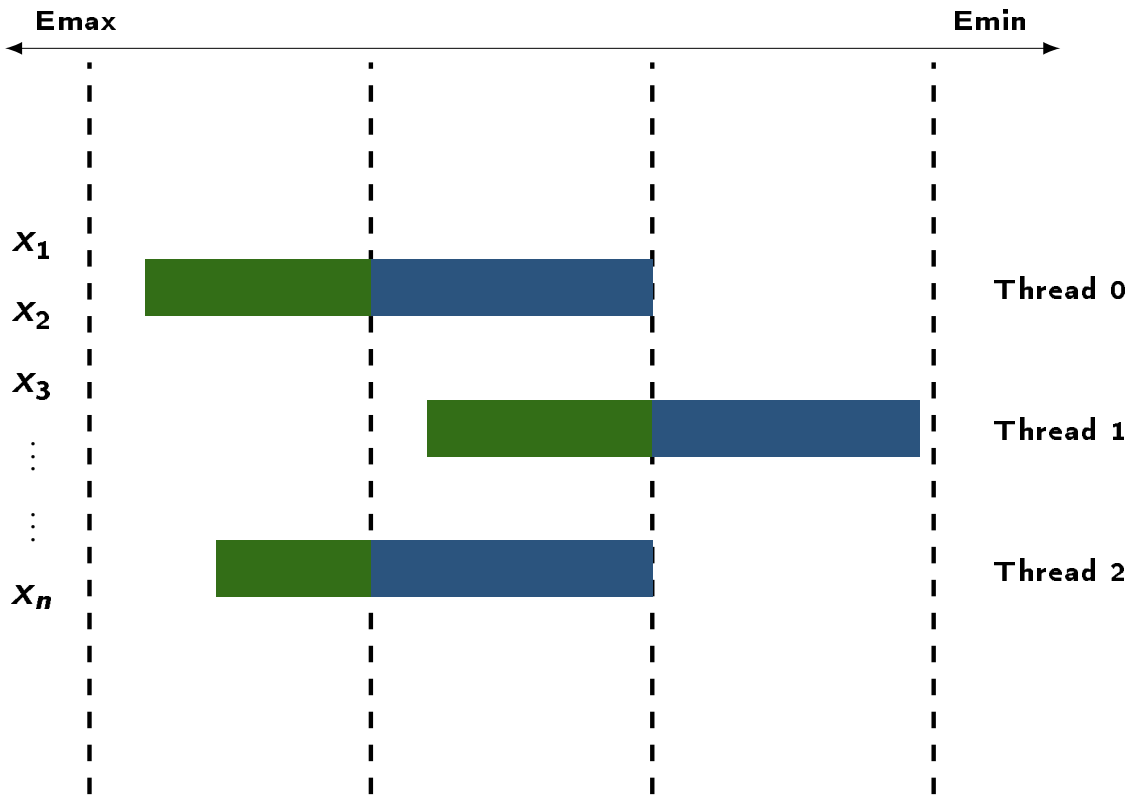
OneReduction (Demmel and Nguyen, 2013)



Steps for parallel

- Max (Compute).
- Sum (Compute).

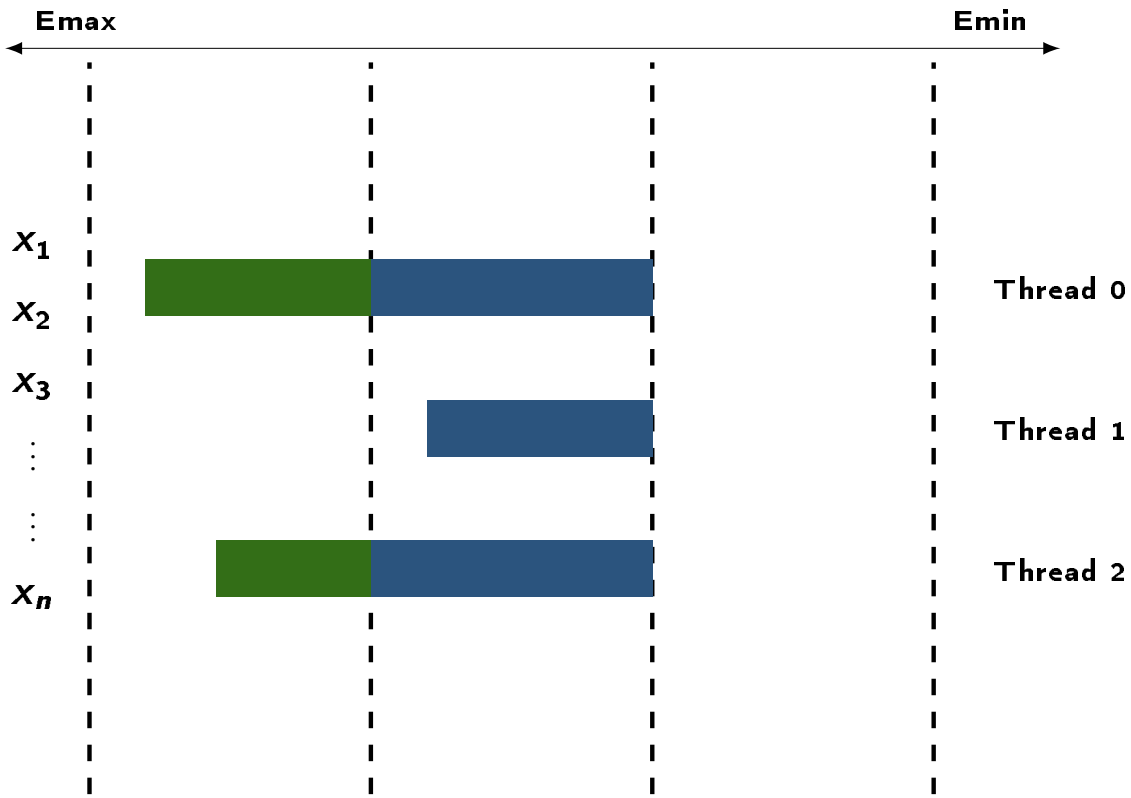
OneReduction (Demmel and Nguyen, 2013)



Steps for parallel

- Max (Compute).
- Sum (Compute).

OneReduction (Demmel and Nguyen, 2013)



Steps for parallel

- Max (Compute).
- Sum (Compute).
- Sum (Reduce).

Table of contents

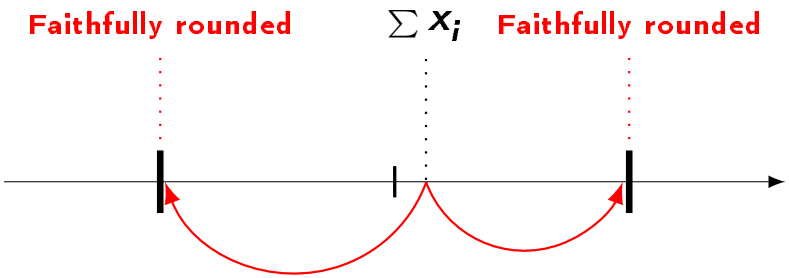
Precise summation algorithms

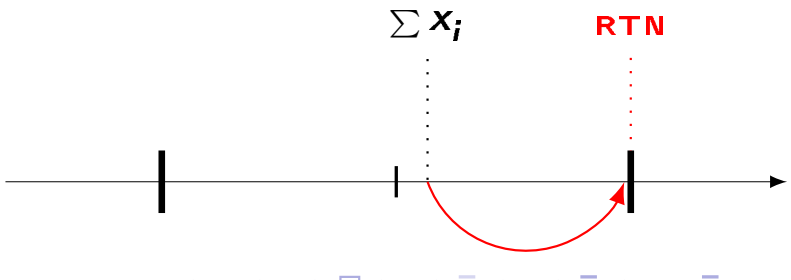
Faithfully rounded (one of the floating-point neighbors)

- FastSum (Zhu and Hayes, 2005).
- AccSum (Rump and al., 2008).
- FastAccSum (Rump, 2008).

Correctly rounded (according to the rounding mode)

- NearSum (Rump and al., 2008).
- iFastSum (Zhu and Hayes, 2009).
- HybridSum (Zhu and Hayes, 2009).
- OnlineExact (Zhu and Hayes, 2010).





Experimental framework

Implementation

- Implemented using C language.

Compiler

- Intel ICC 14.0.2.
- Options : -O3 -xHost -fp-model double -fp-model strict -funroll-all-loops.

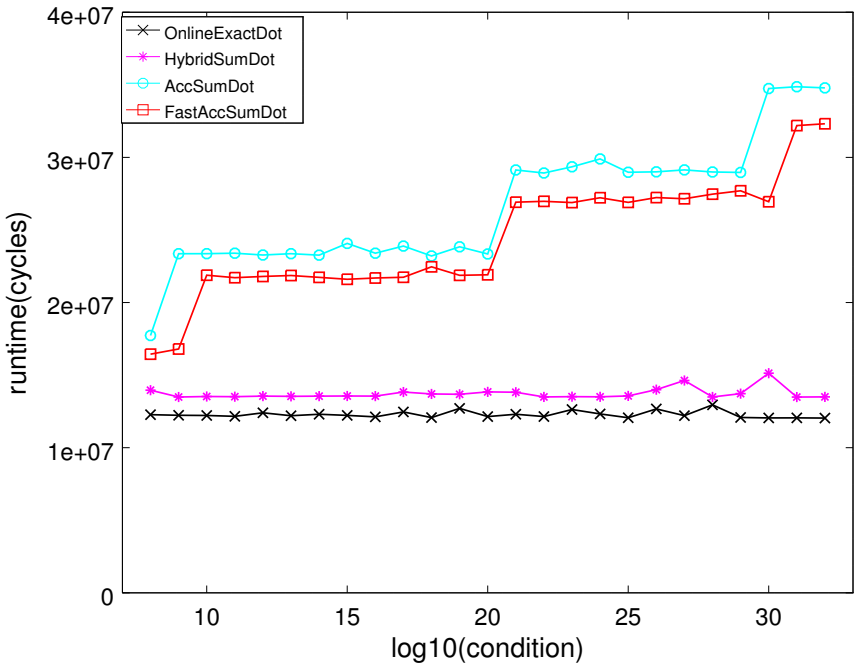
Hardware

- i7-3540M at 3 GHz.
- Turbo boost turned off.

HybridSum and OnlineExact do not depend on the condition number

Implementation

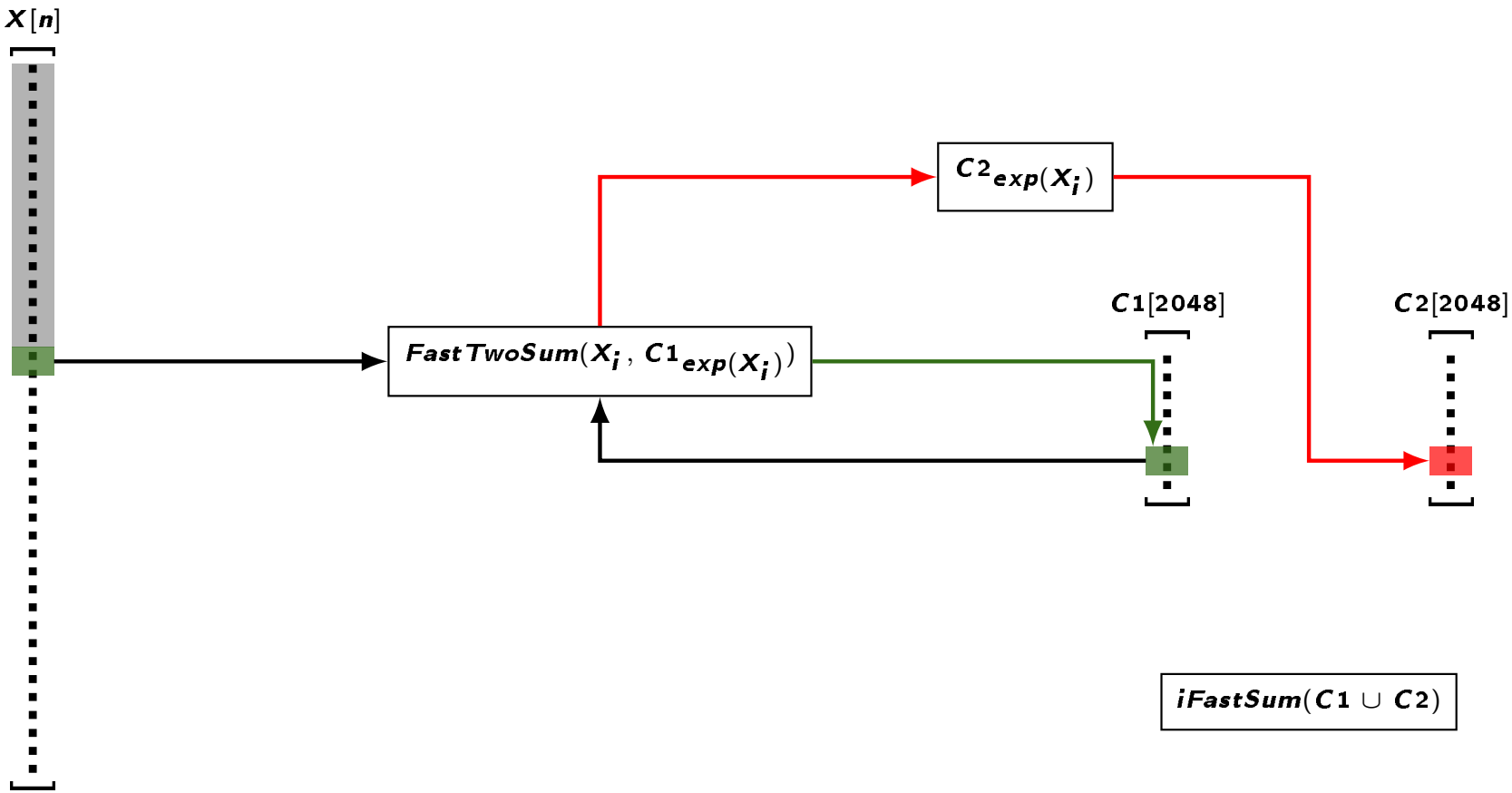
- Manually optimized version for all algorithms.
- Entry vectors size = 10^6.



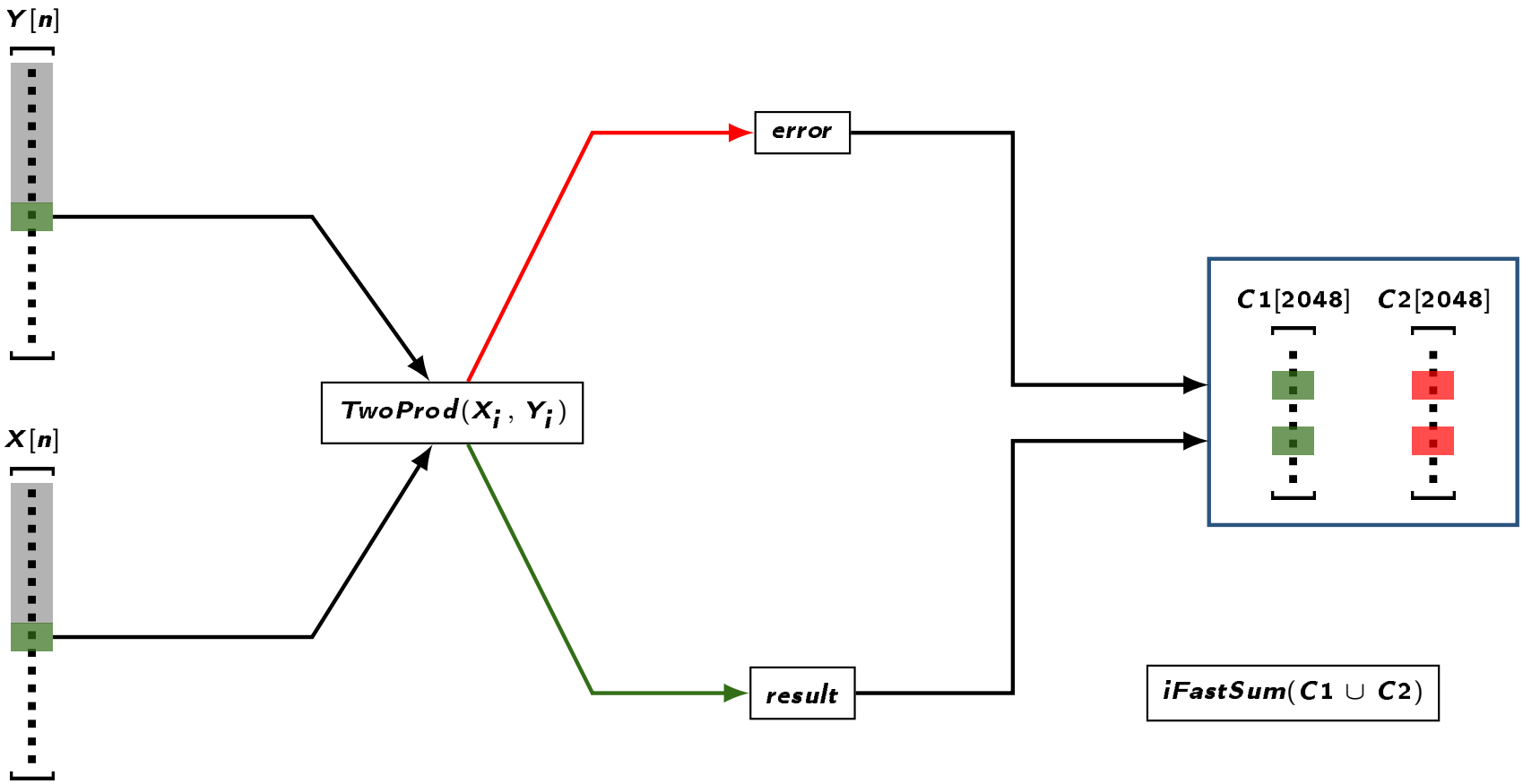
Note

- HS and OLE : condition number independents.

Algorithm OnlineExact (Zhu and Hayes, 2010)



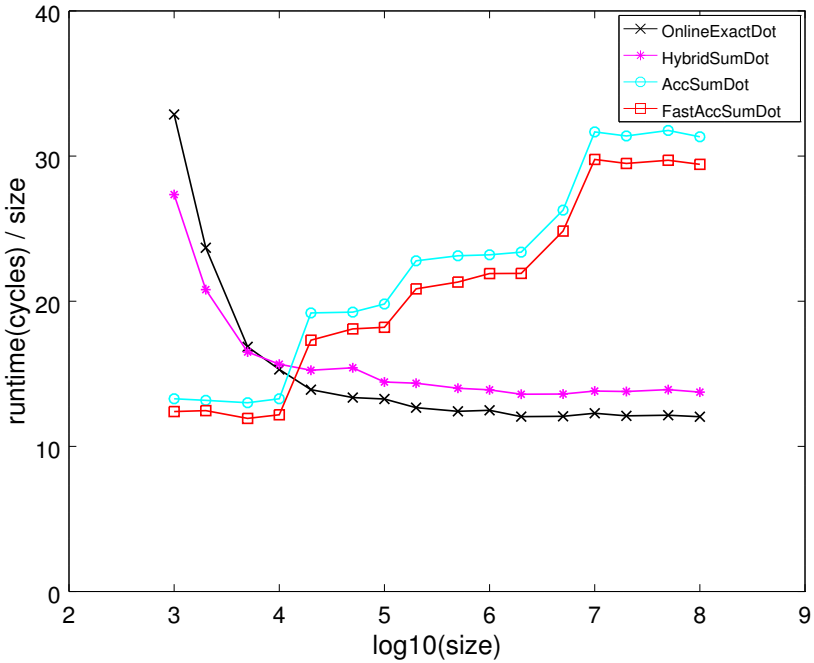
Dot with OnlineExact (Zhu and Hayes, 2010)



HybridSum and OnlineExact are not efficient for small vectors

Implementation

- Manually optimized version for all algorithms.
- Entry vectors condition = 10^{16}.

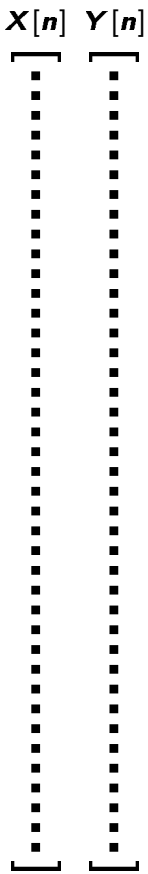


Note

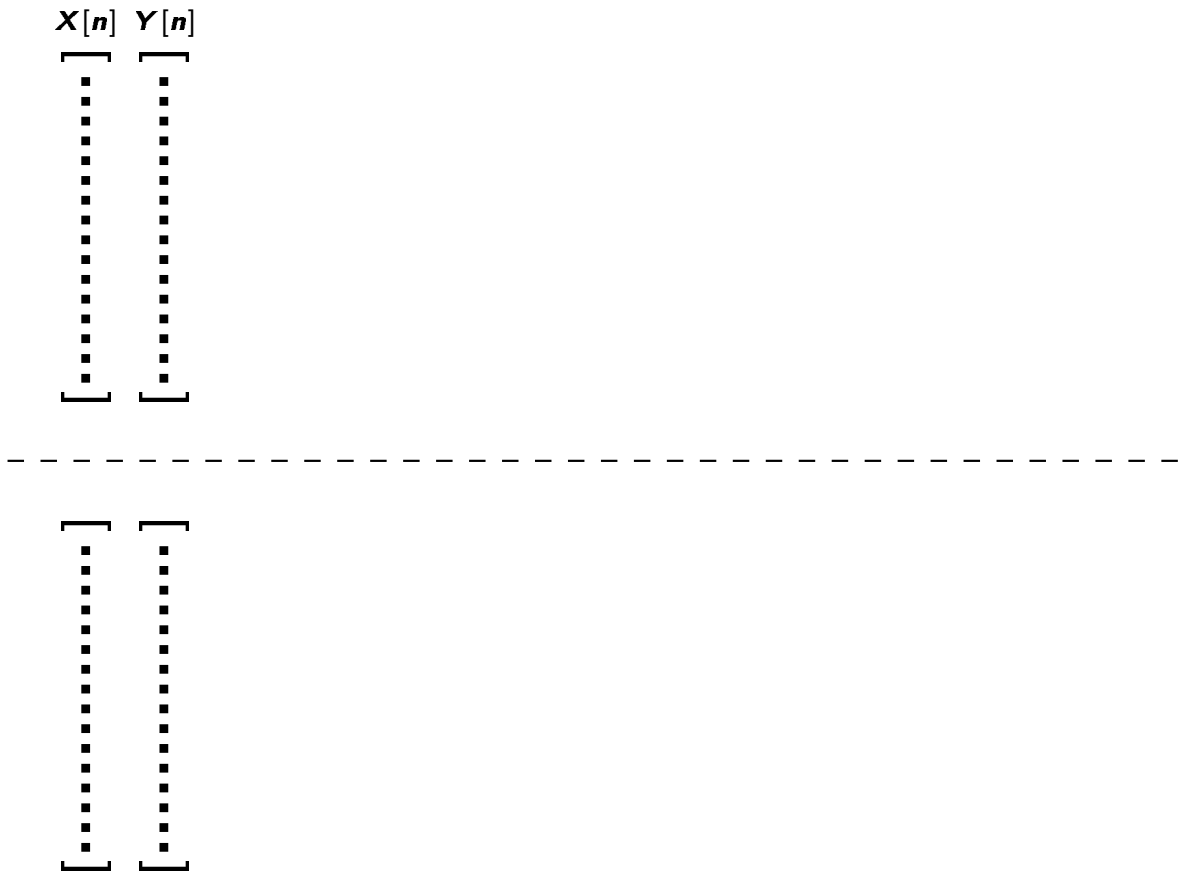
- HS and OLE : more efficient for large vectors.
- Mixed solution is possible.

Table of contents

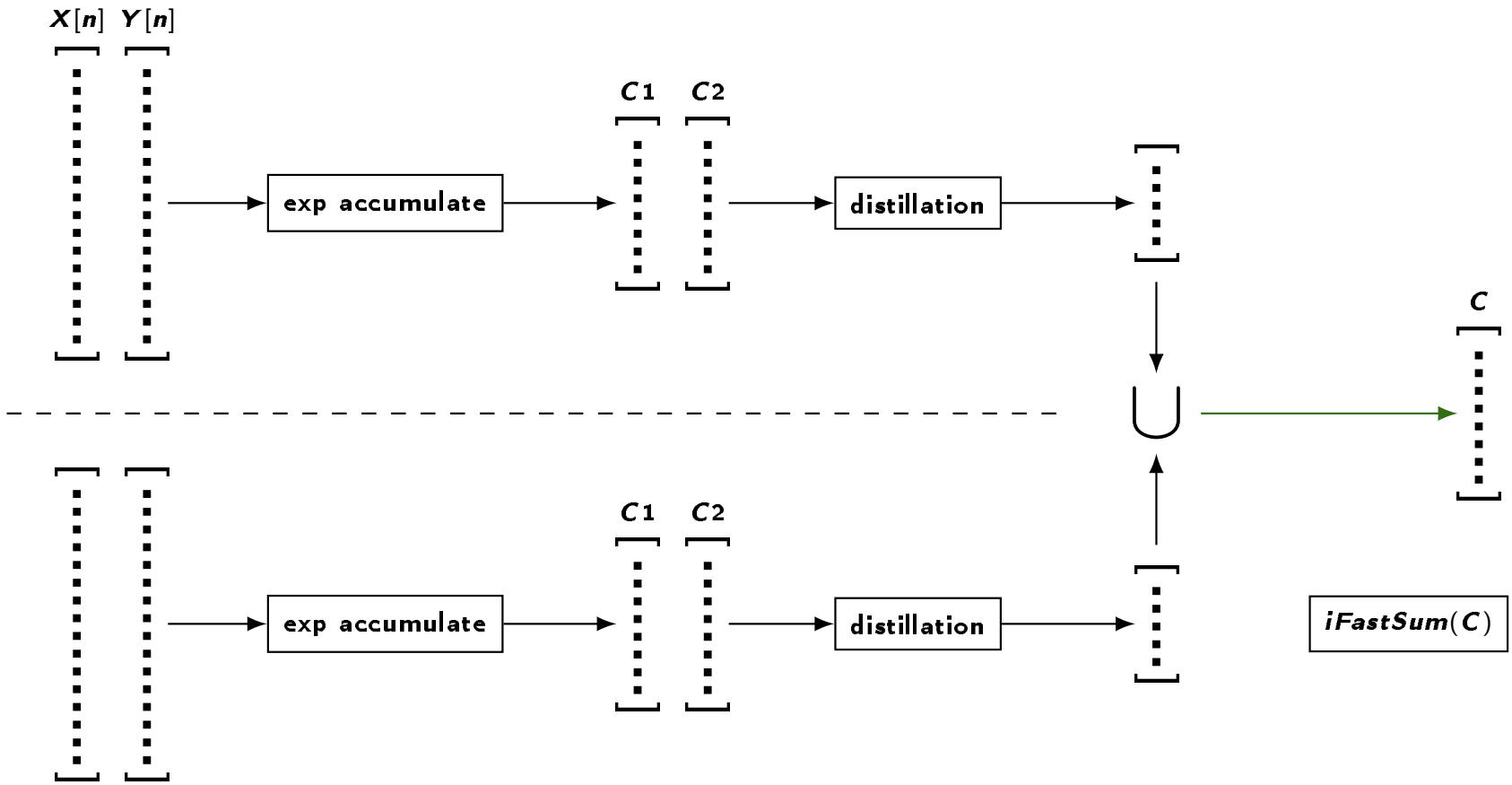
Parallel OnlineExact (2 processors case)



Parallel OnlineExact (2 processors case)



Parallel OnlineExact (2 processors case)



Experimental framework

Hardware

- OCCIGEN (26^{th} supercomputer in top500 list).
- 4212 Xeon E5-2690 v3 socket (L3 cache = 30 M).
- 12 cores on each socket.

Software

- Intel ICC 15.0.0.
- OpenMP 4.0 (Intra socket parallelism).
- OpenMPI (Inter socket communications).

Algorithms

- Classic dot (MKL implementation + MPI reduction).
- Reproducible dot (ReprodDot, FastReprodDot, OneReduction).
- Correctly rounded dot (Parallel version of OnlineExact and HybridSum).

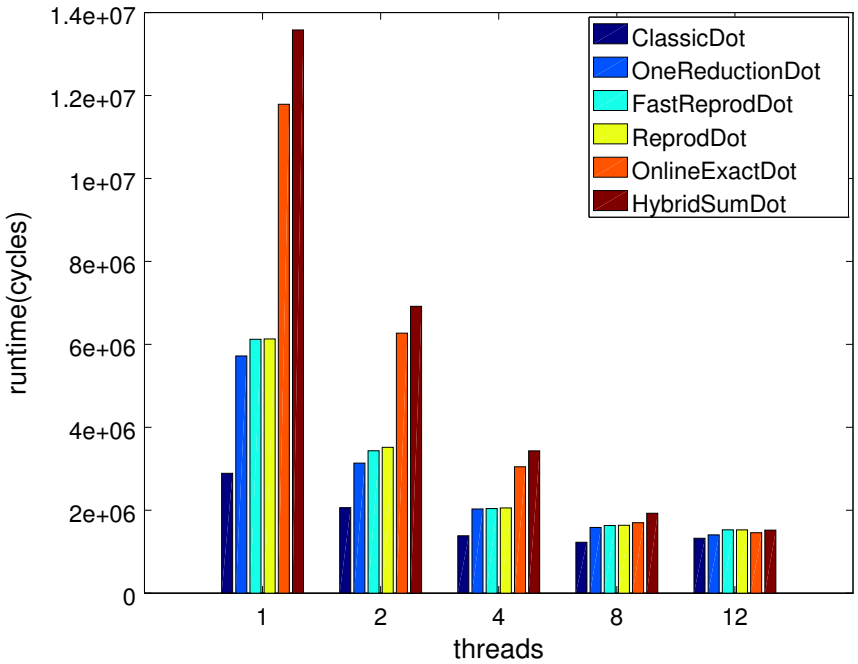
Single socket results

configurations

- #sockets = 1.
- #threads = 1 .. 12 on single socket.

dataset

- Entry vectors size = 10^6.
- Condition number = 10^{32}.



Note

- Classic dot is memory bounded.
- Data hold in level 3 cache.
- ReprodDot and FastReprodDot avoid second memory access.

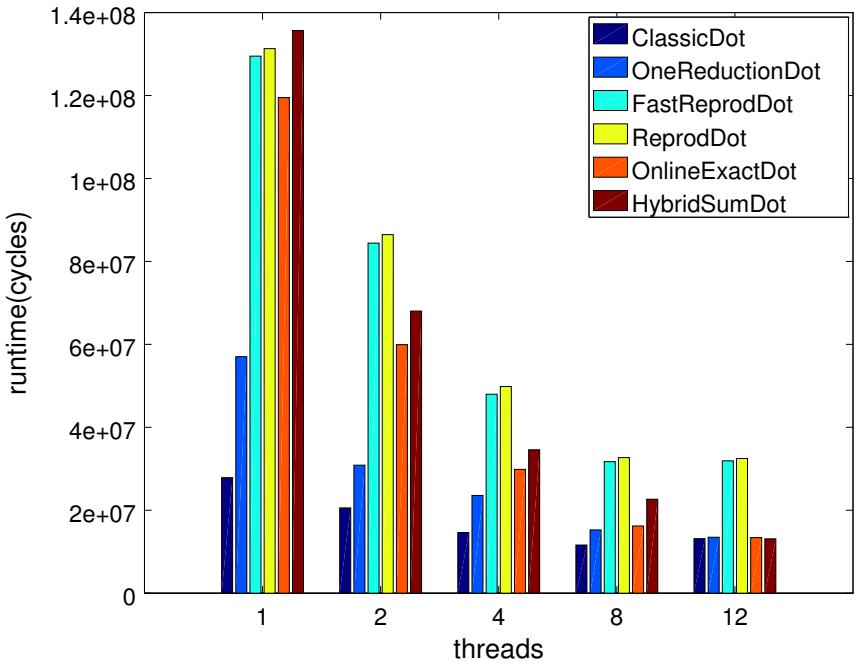
Single socket results

configurations

- #sockets = 1.
- #threads = 1 .. 12 on single socket.

dataset

- Entry vectors size = 10^7.
- Condition number = 10^{32}.



Note

- Data do not hold in cache.
- Memory cost is more important for ReprodDot and FastReprodDot.

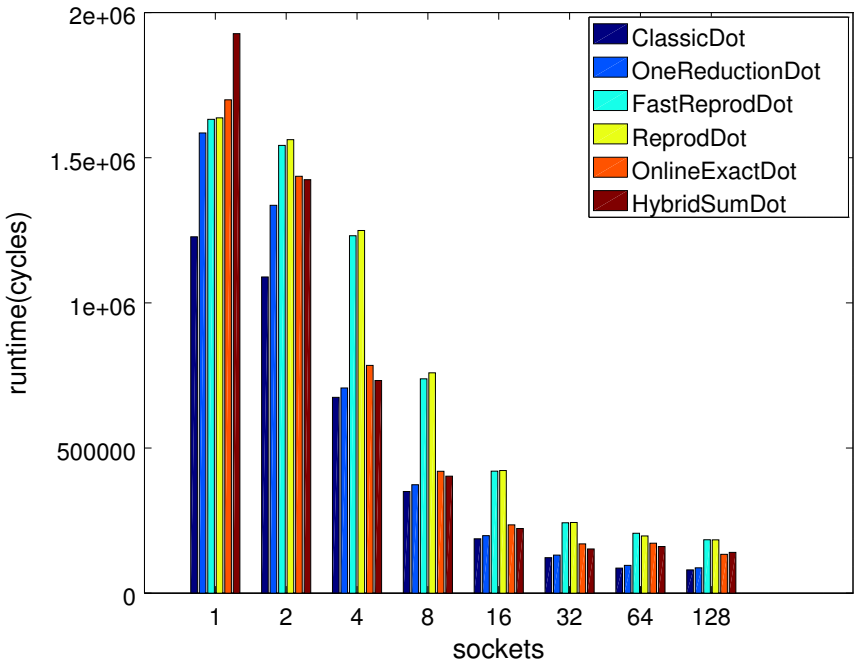
Multi-socket results

configurations

- #sockets = 1 .. 128.
- #threads = 8 per socket.

dataset

- Entry vectors size = 10^6.
- Condition number = 10^{32}.



Note

- Small dataset.
- We do not need too much sockets.

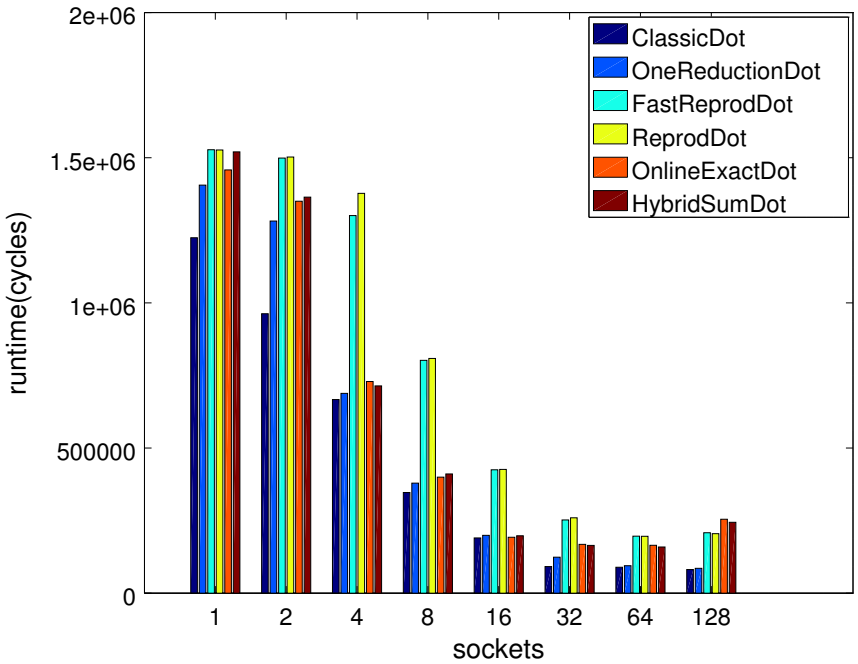
Multi-socket results

configurations

- #sockets = 1 .. 128.
- #threads = 12 per socket.

dataset

- Entry vectors size = 10^6.
- Condition number = 10^{32}.



Note

- Small dataset.
- We do not need too much sockets.

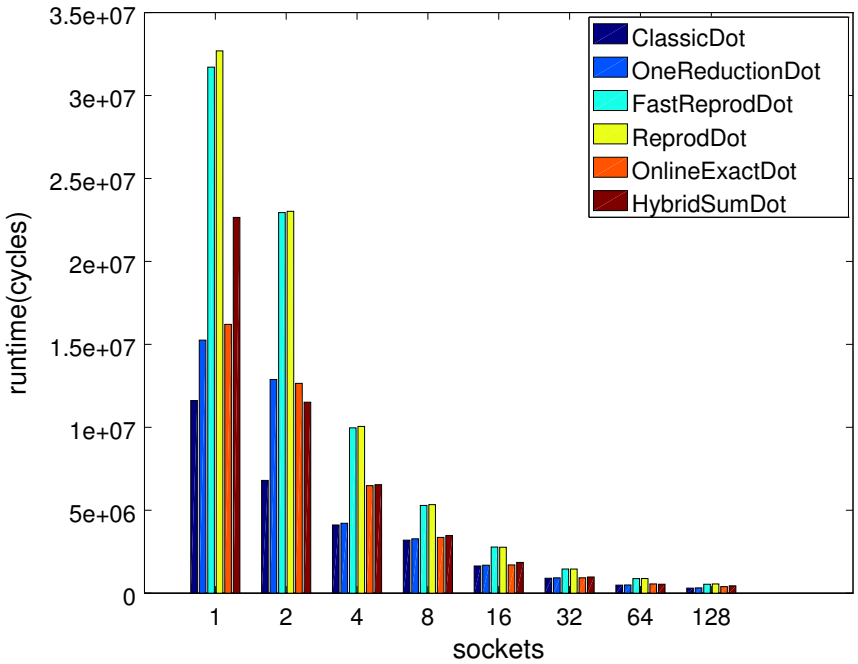
Multi-socket results

configurations

- #sockets = 1 .. 128.
- #threads = 8 per socket.

dataset

- Entry vectors size = 10^7.
- Condition number = 10^{32}.



Note

- Good scaling for large datasets.
- Two communications cost limits ReprodDot and FastReprodDot.
- We need only one communication for OneReduction, HybridSum and OnlineExact.

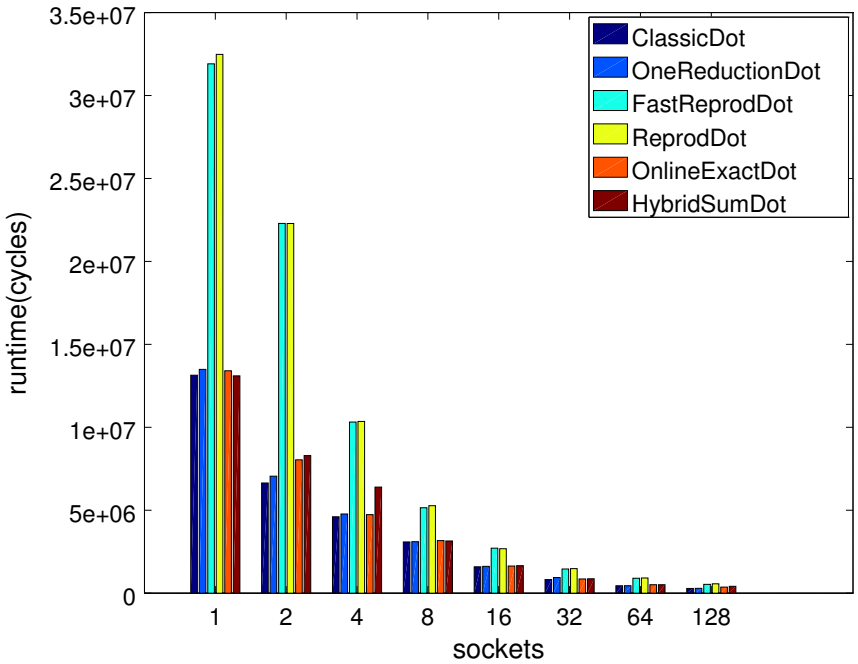
Multi-socket results

configurations

- #sockets = 1 .. 128.
- #threads = 12 per socket.

dataset

- Entry vectors size = 10^7.
- Condition number = 10^{32}.



Note

- Good scaling for large datasets.
- Two communications cost limits ReprodDot and FastReprodDot.
- We need only one communication for OneReduction, HybridSum and OnlineExact.
- Using all cores : advantageous for RTN implementations.

Table of contents

Conclusion

Reproducible level 1 BLAS

- RTN cost for level 1 BLAS is acceptable on massively multi-thread systems.
- Our solution do not depend on condition.
- Intra-socket performance for classic dot is memory bounded.
- Only one communication is required.
- Scale correctly up to 128 sockets.

Future work

Future work

- Level 2 BLAS.
  - GEMV.
  - TRSV.
- Level 3 BLAS will not be easy.
  - DGEMM utilise 95% of peak performance.
  - Used algorithms are not adapted to work on blocks.
  - Some operations can not be vectorised.
  - No balance between FP additions and multiplications.

THANK YOU
FOR
YOUR ATTENTION