DP2E-AI 2025

1st International Workshop on Distributed and Parallel Programming for Extreme-scale AI

June 16-17th, 2025 | Paris, France

Exascale machines and clouds are now available, based on several different architectures and arithmetic. Supercomputing and their **programming paradigms are no longer only lead by historical computational science applications**.

AI, including LLM and machine learning applications in general, is currently **redesigning the architectures and the distributed and parallel programming approaches.**

Nevertheless, **linear algebra and supercomputing expertise are still and will be required**. Decade long researches on distributed and parallel computational science, linear algebra and middleware may be adapted and optimize these new application deployments.

Convergence between those domains is not guaranteed and we need to build bridges and be able to **propose interoperability allowing to avoid separate roadmaps and ecosystems.**

Potential solution mixing distributed and parallel computing, based on a hierachical programming paradigm.

Big Data

Exsacale:Post-Petascale
- Accelarators or not
- Arm-sve
- Network

Mixed and ”new” arithmetic, accuracy analyse

New architectures
- TPU
- Data parallelism oriented
- Parametric parallelism

Arithmetic: 64bits

Linear algebra

From 4 to 64 bits

AI

Data format:

HPC

Sparse : CSR
Matrices
Mesches

Distributed and parallel new programming paradigm, scheduling

TENSORS
Sparse : COO
Graphs

Langage:

C++, …
Librairies

Resilience and energy optimisation

Pyhon,…
API

Dara migration optimisation

HPC, BigData,AI and Quantum computing share important challenges in linear algebra, arithmetic, resilence, energy. They also all contribute together to compute trustable solution to large AI problems.

The main challenges are often the same, but the evaluations and the solutions may be differents.

Quantum

Post-Petascale performance

GC, Sparse matrices
Iterative method
HPCG

Diagonal structures allowing stencil optimisations

Rank	TOP500 Rank	System	Cores	Rmax (PFlop/s)	HPCG (TFlop/s)
1	1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	17406.90
2	7	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	16004.50
3	2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	14054.00
4	3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	5612.60
5	9	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	4586.95

17,4 Pflops, 1% Peak — New #1, June 2025

16 Pflops, 4% Peak — #1 2020-2025

Approx 1% Peak

Approx 0.5% Peak

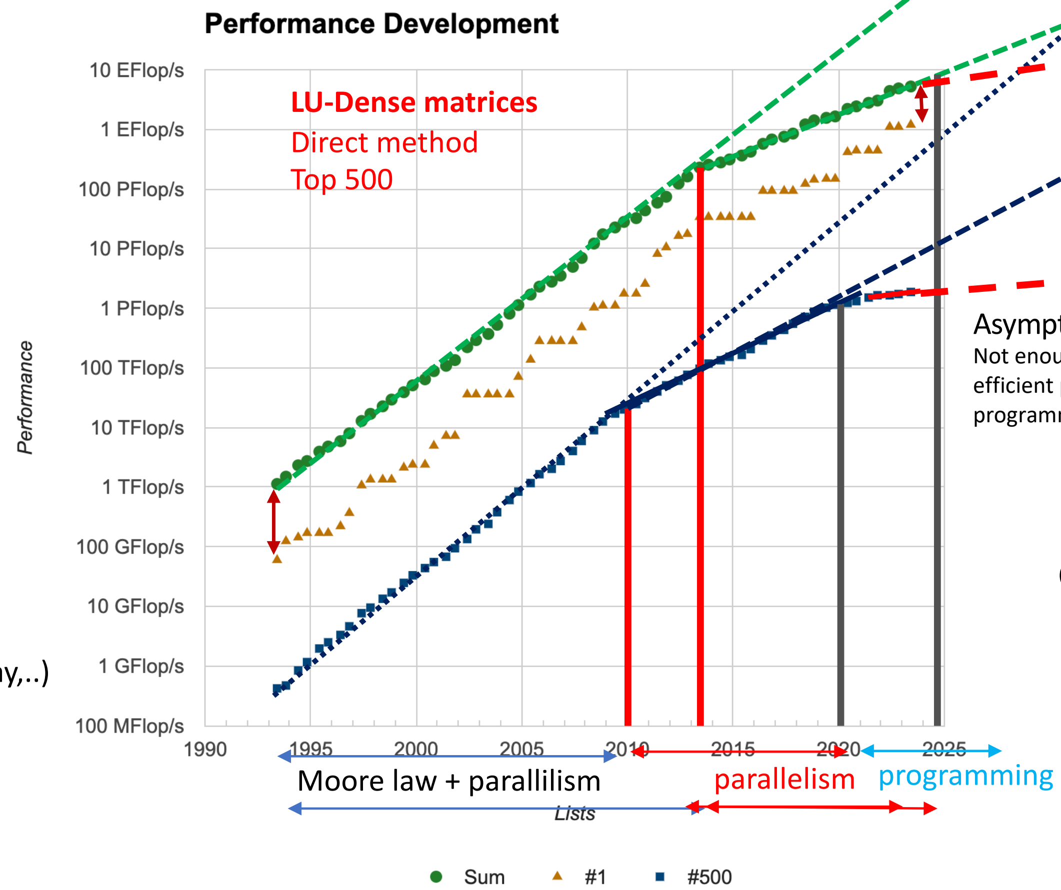
Approx 1%

China owns probably exascale computers (tianhe, sunway,..)

Sparse matrix computation :
Petascale performance (64 bits)

0 3113.94

AI often requires the faster supercomputers, or clouds, or clusters
But not always 64 bit FP



Asymptote?
Not enough efficient parallel programming?

C++ programming

It exists 16bit Exascale computers, such as Cerebras ones, efficient for sparse computation

Toward distributed and parallel smart orchestration for AI

A complex ecosystem : we need decision at runtime and a smart scheduling based on expertise

Arithmetic
- Hybrid (IEEE and others)
- Mixed
- CESTAC, FAYES
- ….

PGAS
API
Librairies
MPI-OpenMP

Distributed and parallel programming

Local optimisation
- Arm vs GPU
- Accelrator vs multicore

Graph of
Tasks-Components.
Independent from the platforms

LA
- U&C
- Iterative refinement
- Dimension reduction (SVD)
- Laplacian (GCN)
- ….

Scheduling
Orchestration

Languages ,Expertises, Algorithms : smart fault tolerant tunning for AI application

AI applications
- Hyperparameters
- Validation?
- Softmax, activation algorithms?

Data
- I/O prefetching (ADIOS)
- Compressions
- Data persitence
- Anticipation of dat migration
- (on fly) conversion
- ….

Fault Tolerance
Resilience
Check-Pointing

Energy
optimisation

We have to exploit expertise and librairies, APIs, software from all these domains : to be able to orchestrate the computing and data management.

Exascale, i.e Post PetaScale

New level of programming

- Higher level : very large number of node lead to consider supercomputers as distributed clusters (and a cloud enters the top10 list)
- Lower level : Network on chip (distributed computation at the lower level

GPU versus Arm-sve

64 bits on top500, but what future for those chips (for example LLM may run with a small number of bits)

Network Topologies : FatTree, Hypercube and others

Energy consumption and fault tolerance are two crucial problems to optimise ,

- The energy consumption is mainly associated with data migrations, including I/0,
- **Software and Hardware faults. As the number of nodes and the complexity of networks and software increase, the time between two faults decreases and the time to restart the computation increases (it is really a problem for long LLM training computing)**

Scheduling and graph of parallel task is often require

Linear Algebra

Non-Hermitian Sparse matrix (dynamic patterns)

Unite and Conquer methods

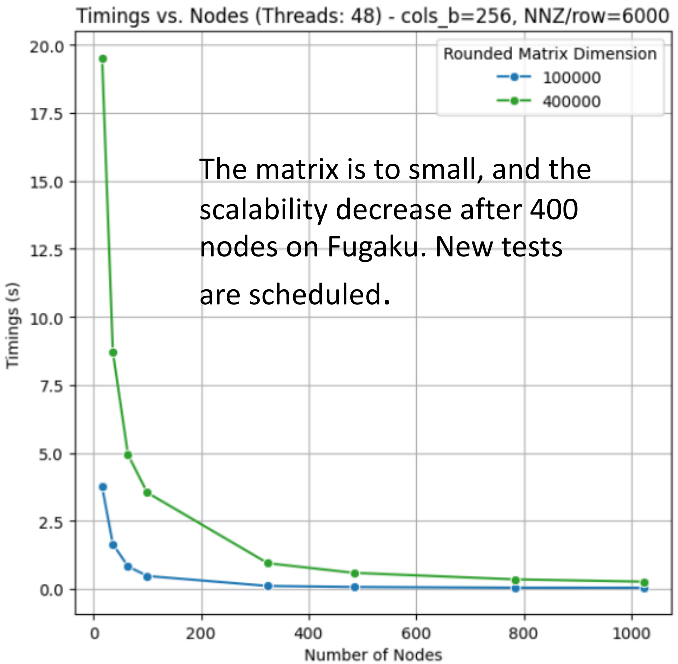
SVD (dimension reduction,...)

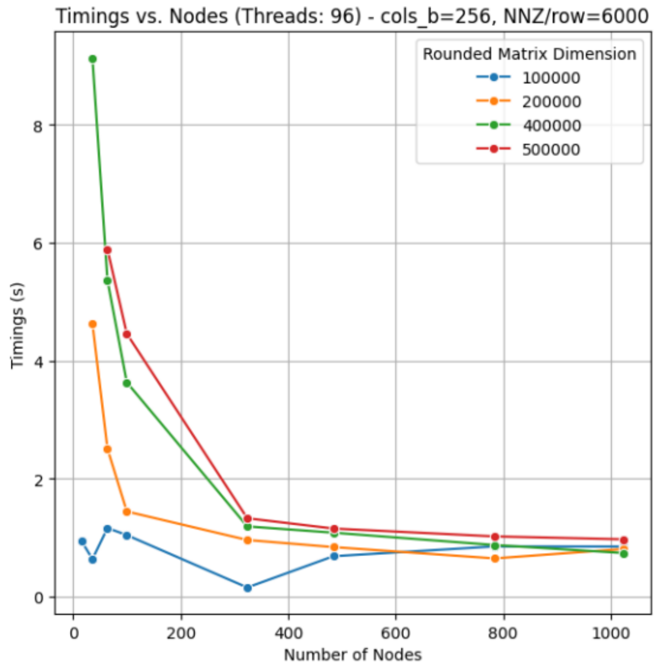
Sequence of sparse (by block or not) matrix by dense matrix-or-vector products

Mixed and new arithmetic, accuracy analysis

Non symetric linear algbra is very important for LLM and AI, as we have often to sparsify the data and/or managing large graphs.

For sequence of the product of a **sparse matrix by a rectangular dense skiny matrix** (similar to some part of the attention computatiion on some LLM methods), on Fugaku, the scalability may be interesting.





Fugaku

Machine Learning, AI

Arithmetic mixed, multi

- What arithmetic for a given part of a software
- Direct method (low precision) – Iterative method (high precision)
- Floating point-Integer : more parallelism, then data migration, and an extra synchronisation

Graphs : very larges and very sparse non-symmetrical
Matrices (Impossible to store dense vectors or right hand dense matrices).

LLM/Attention : Sparse per blocks by dense matrix products (dynamic scheduling of the dense blocks, dense computation at the lower level, Polyhedric)

And exascale computing for the training and memory optimisation for inference (in particular)

Training versus inference : different distributed and parallel optimisations

Machine Learning and AI are new applications for HPC, with some criteria sometime different than the ones from computational science. Nevertheless, we may exploit the decade long knowledge on HPC, Linear algebra and computational science programming and computing (even if new language and the evolution of APIs also change the general programming ecosystem).

Linear Algebra and PR is for example often required
Laplacian iteration and oversmothing is also en example of linear alebra problems for AI (for example : GCN)

Programming Paradigm

Very large distributed data
Large number of nodes
Large networks
Multi-mixed-arithmetic

Graph of parallel Tasks

Composents (for each task) :

- Abstracts
- Implementations
- Executions
- Graphs

Exascale
Big Data
Arithmetic
Linear Algebra

Smart-Tunning based on expertise. Selection at runtime of better parameter and programming straiegies

We may give some expertise inside abstract and/or implemantation components.

Language allowing that programming
Linear Algebra methods analysed and optimised for those applications
Adapted arithmetic (using several numerical methods)

We need a smart middleware and scheduler

Abstract component : describes the method used, the data "in", "out", or "inout", and give some general expertises (numerical stability, number of operations, arithmetic requiered depending of the size of the data,…)

Implementation component : depends of the chosen language. For example : if it is a parallel language, give the number of processors for a task, or other informations.

Execution component: the (cross-)compiled code to be executed

Graph component : the parallel code using a language allowing the describtion of any parallel graph.

On order to discuss on these challenges, we organize this two-day workshop at Ecole de Mines de Paris to bring together scientists in the fields of HPC, AI and Linear Algebra, or involved in associated applications. The program is composed of Keynote talks, dedicated sessions on distributed and parallel programming programming for AI methods, added with 2 panels and one general discussion.

We selected 3 questions to initiate discussions :

- What **missing interoperability layers** (software, standard, or abstraction) would most accelerate convergence between traditional HPC linear algebra workflows and today's extreme-scale AI workloads.
- Looking ahead to 2030, do you expect the principal bottleneck for extreme-scale AI to be **data**, **algorithms**, **resilience** or **energy**, and how does that prediction shape your research priorities today
- Given the different developments in architecture processors for AI and "computational science", do you think we'll see a convergence or divergence of roadmaps?

Have a nice workshop

https://dp2eai-2025.cri.minesparis.psl.eu/

Thanks to ecole des mines for hosting us and for the organization

Merci à l'Ecole des Mines de nous accueillir et pour l'organisation