Technology watch and achievements in programming on long vector processors

Patrick Bousquet-Mélou[(1)], Jean-Baptiste Harry[(2)], Benoist Gaston[(1)], Patrick Amestoy[(3)], Jean-Yves L'Excellent[(3)], Ahmed-Sherif Omran[(4)], Valentin Valtchev[(4)]

(1) CRIANN (support@criann.fr)

(2) NEC

(3) Mumps Technologies

(4) LCS-ENSICAEN UMR 6506

JCAD 2022, 10-12 Octobre 2022 - Dijon

CRIANN

Normandy Regional Computing Center

- Created in 1991 as a Non Profit Organisation by Higher Educations Institutions & Universities
  - HPC center and regional network
- Located in Rouen Engineering Campus
- 15 employees
  - 9 engineers or PhD, 2 work-study workers
- Funding is mainly public
  - Running costs supported by Normandy Region
  - Projects / investments funded by projects with EU, French State & Normandy Region
  - A small self-financing part
- HPC for public research, also open to private

MesoNET project

Vectorial computing at CRIANN

- Supply awarded to NEC in April 2022
  - 1st stage (end of 2022)
    - 9 compute nodes) x (8 Vector Engines)
    - InfiniBand interconnect (HDR, 2x200 Git/s per node)
    - Storage: Spectrum Scale, 500 TB, 3.5 GB/s
    - Vector Engine: SX-Aurora TSUBASA 20B
      - 8 cores 1.6 GHz
      - 64 registers of 256 double precision elements (16384 bits) per core
      - 48 GB HBM2 (High Bandwidth Memory)
      - 1.53 TB/s memory bandwidth
  - 2nd stage (2024) will double the capacity

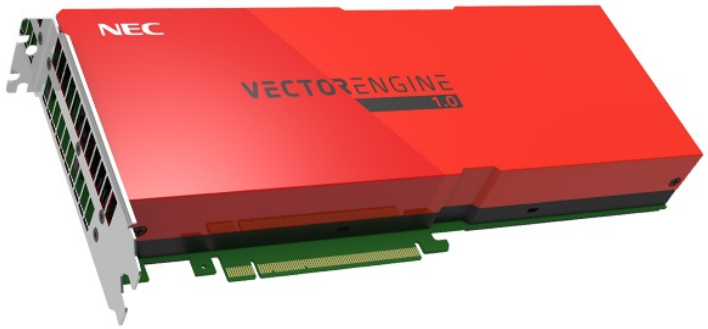
MesoNET project

Vectorial computing at CRIANN

- First stage roadmap
  - Verification of Regular Service: 11/28/2022 - 12/16/2022
  - 12/19/2022: opening to other candidate labs and industries
    - Mechanical engineering, materials and energy sciences, atomistic simulation, climate, ocean circulation, etc.
  - Training program at CRIANN starting in Q1 2023
  - Contact: support@criann.fr

Vectorial computing

Technology watch at CRIANN (prior to MesoNET)

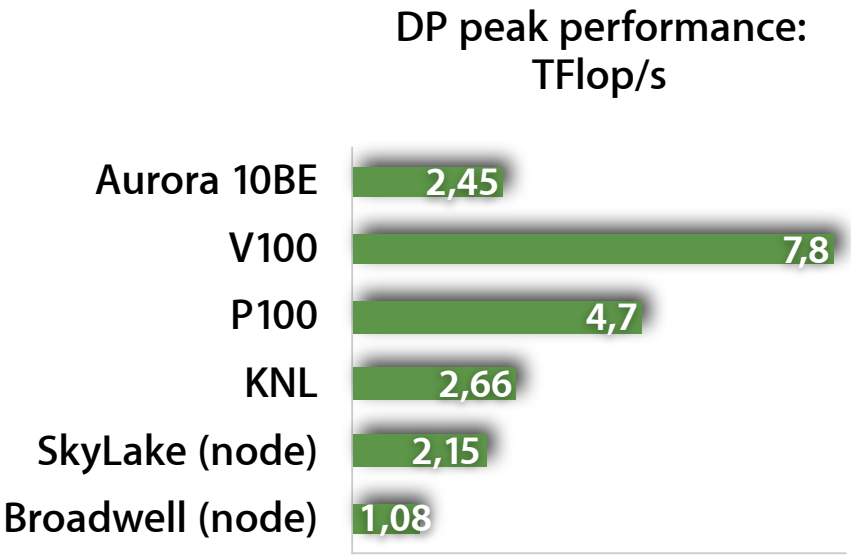


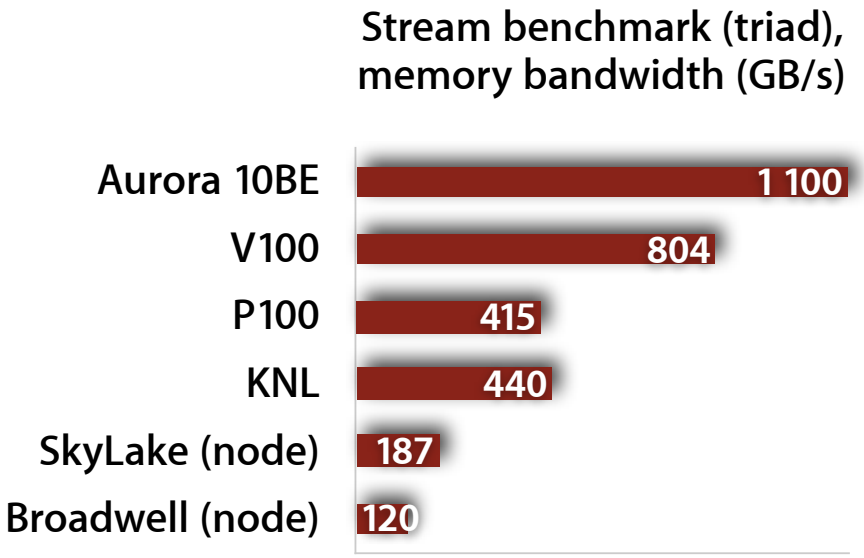
- Purchase of a server in 2020
  - 2 NEC Aurora 10BE vector engines (8 cores, 1.35 TB/s memory bandwidth per VE) + 2 Cascade Lake CPUs
- Tests related to some of the fields of Normandy academic labs
  - Simulation: image processing, CFD/ LBM, FFT, molecular dynamics
  - AI/DL (frameworks tests with SOL library)

Processor architectures

at CRIANN (September 2022)

Aurora 10BE: 8 cores 1.4 GHz, 1.35 TB/s memory bandwidth





- Aurora best target: memory bound applications

2D image processing codes porting to Aurora

Compiler vectorization diagnosis driving code changes

Vectorization obstructive instructions detected in initial C programs => new arrays (stored in new loops) enabled efficient vectorization for native mode porting (automatic full offloading on vector engine) of two applications

```
NEC C/C++ Compiler (3.0.4) for Vector Engine     Tue Jun  2 17:18:26 2020
FILE NAME: interpol2D.c
FUNCTION NAME: interpol2D
DIAGNOSTIC LIST
 LINE            DIAGNOSTIC MESSAGE
   19: par(1803): Parallelized by "for".
   21: vec( 103): Unvectorized loop.
   21: vec( 110): Vectorization obstructive function reference.: diff_b
   …


 LINE   LOOP      STATEMENT

   12:          void interpol2D(int * tab_bounds, int M_Re, int N_Re, double *
ld2U2_local, double * ld1U1_local, double ** C, double * interpolant_local,
   13:                          double * dx_interpolant_local, double *
dy_interpolant_local, int nlig, int ncol){
   14:
   15:                int p,q,i,j;
   16:                double xi_1,xi_2;
   17:
   18:          #pragma omp for
   19: P------>   for(i=0;i<M_Re;i++){
   20: I
   21: I+----->       for(j=0;j<N_Re;j++){
            …
                                dx_interpolant_local(i, j) = C(q-2,
p-2)*diff_b(xi_1+1.0)...
```

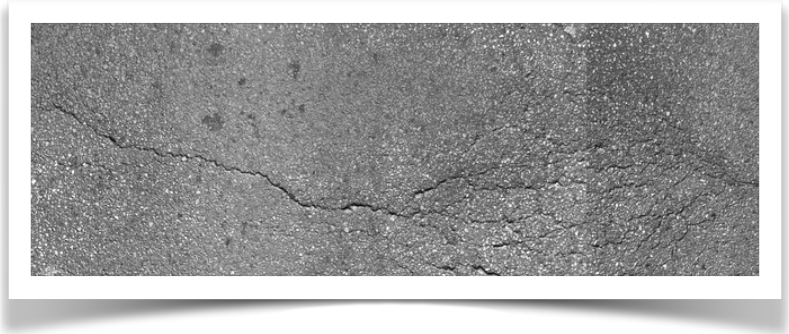
```
NEC C/C++ Compiler (3.0.6) for Vector Engine     Wed Jul  1 17:36:38 2020
FILE NAME: interpol2D.c
FUNCTION NAME: interpol2D
DIAGNOSTIC LIST
 LINE            DIAGNOSTIC MESSAGE
   …
   93: inl(1222): Inlined: func_b
   …
  107: vec( 101): Vectorized loop.
   …
   33:          void interpol2D(int * tab_bounds, int M_Re, int N_Re, double *
ld2U2_local, double * ld1U1_local, double ** C, double * interpolant_local,
   34:                          double * dx_interpolant_local, double *
dy_interpolant_local, int nlig, int ncol){

   63:          #pragma omp for
   64: P------>       for (i = 0; i < M_Re; i++){
   65: I
   66: IV----->              for (j = 0; j < N_Re; j++){
   …
   93: II     I                    d3(i, j) = diff_b(xi_1+1.0);
   …
  104:          #pragma omp for
  105: P------>       for (i = 0; i < M_Re; i++){
  106: I
  107: IV----->F             for (j = 0; j < N_Re; j++){
   …
  121: II     G                   dx_interpolant_local(i, j ) = C(q-2, p-2)*d3(i, j)
```

2D image processing

Bituminous surfacing crack recovery

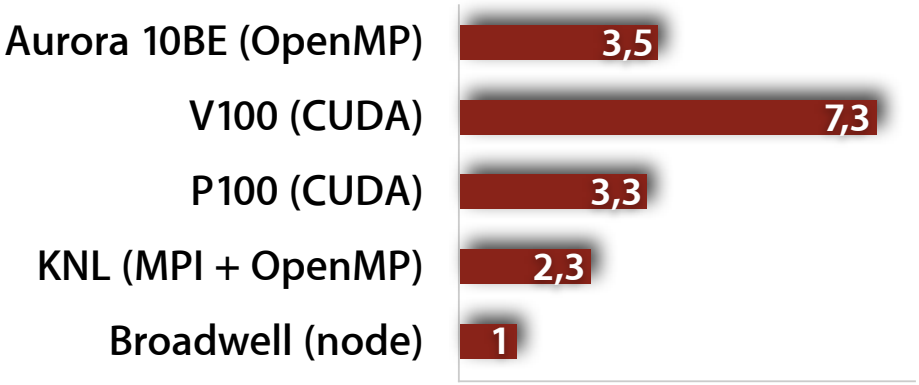
Courtesy of CEREMA (Centre d'Études et d'Expertise sur les Risques, l'Environnement et l'Aménagement)



Aurora 10BE: 8 cores 1.4 GHz, 1.35 TB/s memory bandwidth

9 M pixels

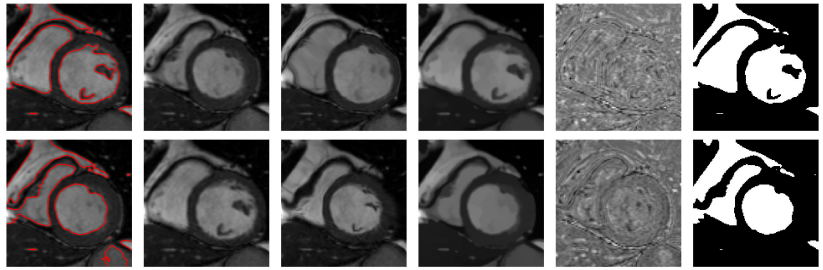
Device or KNL acceleration / (CPU code on 28 Broadwell cores)



- LMI lab (INSA Rouen Normandie)
  - Non-linear elasticity theory, partial differential equations solved by finite difference method
    - high-order spatial scheme
      - high arithmetic intensity adapted to Volta GPU architecture

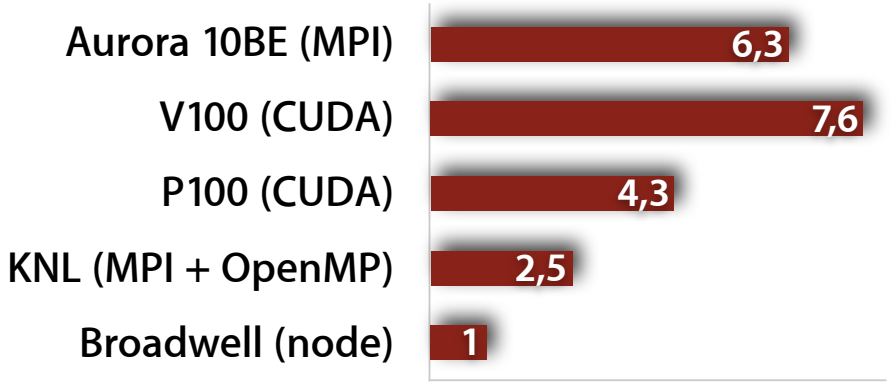
2D image processing

Joint segmentation/ registration model



4,5 M pixels

Device or KNL acceleration / (CPU code on 28 Broadwell cores)



Aurora 10BE: 8 cores 1.4 GHz, 1.35 TB/s memory bandwidth

- LMI lab (INSA Rouen Normandie)
  - Partial differential equations solved by finite difference method
    - 1 day porting to Aurora
    - First-order spatial scheme
      - More memory bound application
  - V100 20% better than Aurora
    - Sufficient qualitative result (Stencil Code Accelerator may improve …)

HPC kernel porting to vector engine (VE)

Prior to applications (Université Rouen Normandie)

Action	Scientific target	Results
Test of a fluid dynamics, 2D LBM (Lattice Bolzmann Method) FORTRAN kernel (lid-driven cavity flow)	Solid-liquid phase change modeling with LBM	Acceleration relatively to an Intel SkyLake 6130 compute node, 4096^2 mesh points, single precision: -1 ***Aurora 10BE*** VE (OpenMP): ***4.1*** -1 ***V100*** GPU (OpenACC): ***3.6*** -1 ***KNL*** (OpenMP): ***2.1*** -1 ***SkyLake*** (node, OpenMP): ***1***
Test of a 3D MPI FFT FORTRAN kernel with FFTW library (vector engine version) **fftw_mpi_execute_dft_r2c()** **fftw_mpi_execute_dft_c2r()**	Materials sciences application, carbon diffusion in martensites	Acceleration relatively to an AMD EPYC 7642 compute node, 1000^3 mesh points, double precision: -1 ***Aurora 10B*** VE, _r2c: ***6*** -1 ***Aurora 10B*** VE, _c2r: ***5.4*** -1 ***EPYC 7642*** (node): ***1***

Aurora 10B: 8 cores 1.4 GHz, 1.22 TB/s memory bandwidth
Aurora 10BE: 8 cores 1.4 GHz, 1.35 TB/s memory bandwidth

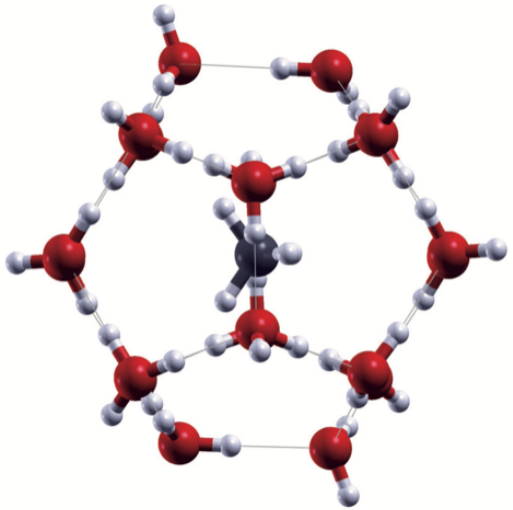
SX-Aurora technology watch in other areas

at CRIANN

Action	Field	Results
Deep learning ResNet50 model benchmarking with SOL library	AI	Aurora performance (*) > x86 ~ P100 < V100 (2.28 factor)
Functional evaluation of optimized Python libraries: NCLPy	Programming tools	- Easy to use (import nclpy as vp) - Interoperability wtih numpy - pros: 10-20 faster than numpy on Cascade Lake host - cons: large arrays do not fit in memory
CPU/Aurora hybrid programming (offloading) evaluation in FORTRAN		FORTRAN version of VEDA release by NEC (Vector Engine Driver API, already existing in C)

(*) For AI kernels called from HPC applications with batch size equal to 1, Aurora is only slightly slower than GPUs

Ab initio study of the stability, diffusion and storage capacity of H2, CH4 and CO2 in clathrate hydrates: view of the methane molecule (center) stored in a small sI hydrate cage.



Few weeks effort by NEC
Vectorization compiler diagnosis and runtime profiling
- Factorial computation optimization
- Changing calls to standard BLAS functions to NEC-optimized versions
- Improvement of vectorization ratio by means of loop structure changes
- etc.

Molecular dynamics

Quantum Espresso (QE)

- LCS-ENSICAEN lab (UMR 6506)
  - Ahmed Sherif-Omran (PhD student), Valentin Valtchev (Research Director)
    - clathrate hydrate modeling for greenhouse gases (CO2, CH4) storage
  - Use of QE 6.5 on CRIANN x86 architecture
  - Porting to vector engine project: 2020-2021, LCS/CRIANN/NEC

Molecular dynamics

Quantum Espresso 6.3: VE porting achievement (2021)

- Large case (408 atoms) with VE (vector engine) 10A / AMD 7642 CPU

Comparison at ~ equal elapsed time

6 AMD 7642 nodes
- 12m41 (4 steps)
- 3750 Watts

vs 3 VE
- 12m52 (4 steps)
- 1100 Watts

=> 3.4 factor reduction of energy consumption with Aurora VEs

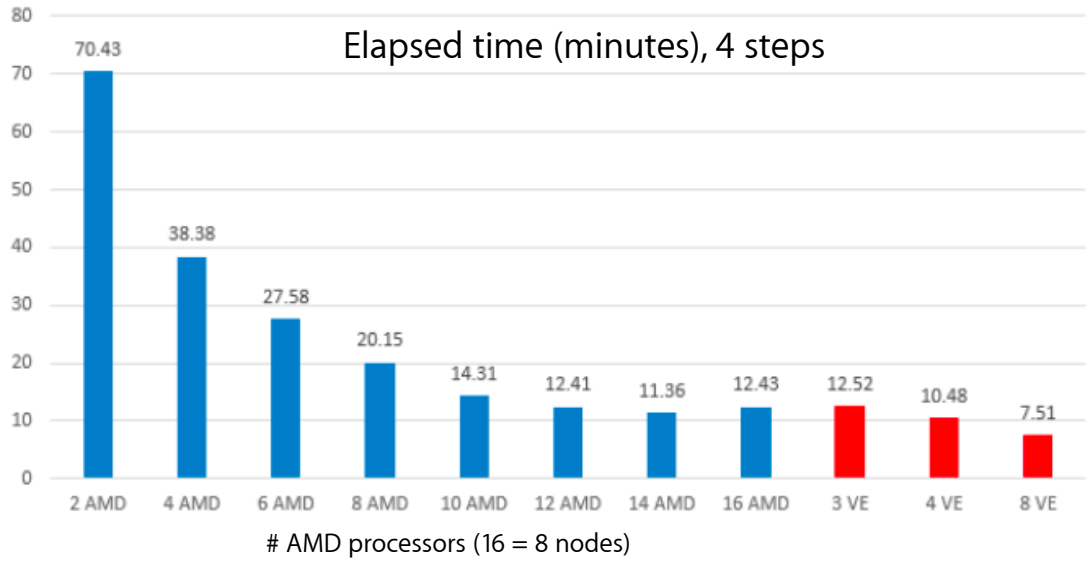
Comparison at ~ equal watts consumption

2 AMD 7642 nodes
- 1250 Watts
- 38m38 (4 steps)

vs 4 VE
- 1300 Watts
- 10m48 (4 steps)

=> 3.5 factor reduction of elapsed time with Aurora VEs

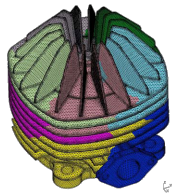
Aurora 10A: 8 cores 1.6 GHz, 1.35 TB/s memory bandwidth



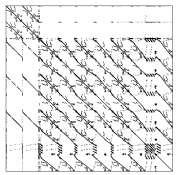
https://github.com/SX-Aurora/QuantumEspresso

Sparse linear algebra – multifrontal solver MUMPS

- `MUMPS` (MUltifrontal Massively Parallel Solver, `http://mumps-solver.org`): free package to solve $\mathbf{AX} = \mathbf{B}$, $\mathbf{A}$ sparse, $\mathbf{B}$ dense or sparse



Code Aster (EDF)



sparse matrix

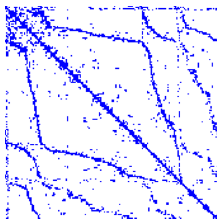
Critical step in HPC simulations (time, memory)!

- codeveloped by CERFACS, CNRS, ENS Lyon, INPT, Inria, Bordeaux University and, since 2019, Mumps Technologies
- Various applications: electromagnetism, structural analysis, circuit simulation, finite element, geophysics, optimization, ...

Mumps Technologies – NEC collaboration (2021-ongoing)

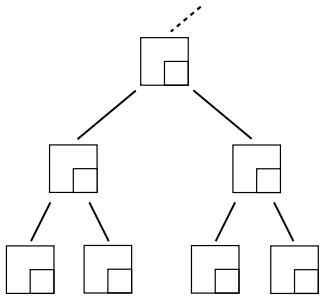
- Experiment and optimize MUMPS solver on NEC hardware
  - NEC SX-Aurora 10B, 8 cores, 1.4 GHz, 2.15 Tflops/s peak, 48 GBytes
  - Initial focus: 1 MPI, 8 threads
- Compile and execute on Vector Engine (VE), except scalar parts
- Make improvements publically available as soon as possible

MUMPS: three-phase scheme to solve $AX = B$



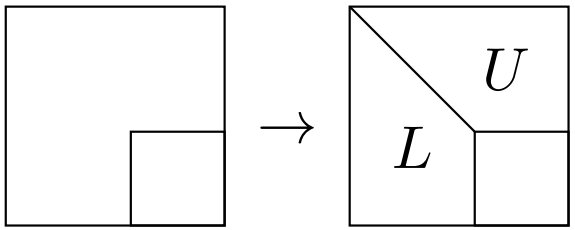
Sparse matrix A

$\Rightarrow$



Elimination tree

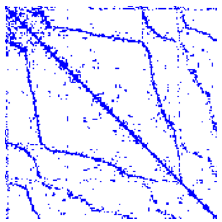
$\Rightarrow$



Frontal matrix at each node: partial factorization

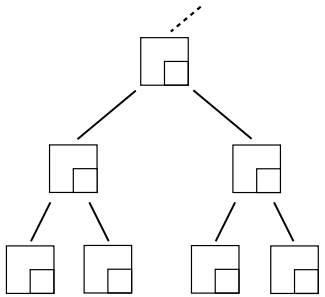
1. Analysis phase:
   - Reorder matrix (Metis, Scotch): $\mathbf{A \rightarrow PAP^T}$
   - Symbolic factorization $\rightarrow$ elimination tree (amalgamation)
2. Factorization phase:
   - $\mathbf{A = LU}$
   - Dense kernels (BLAS) at each tree node (frontal matrix)
3. Solution phase:
   - Triangular solves: $\mathbf{LY = B}$, then $\mathbf{UX = Y}$
   - (possibly) Improvement of solution (iterative refinement), error analysis

MUMPS: three-phase scheme to solve $AX = B$



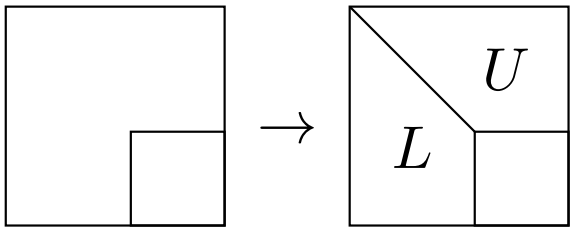
Sparse matrix A

$\Rightarrow$



Elimination tree

$\Rightarrow$



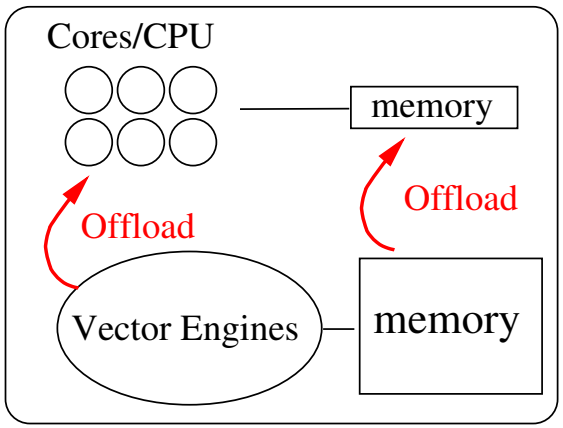
Frontal matrix at each node: partial factorization

1. Analysis phase: graph/scalar algorithms
   - Reorder matrix (Metis, Scotch): $\mathbf{A \rightarrow PAP^T}$
   - Symbolic factorization $\rightarrow$ elimination tree (amalgamation)
2. Factorization phase: computationally intensive, long vectors needed
   - $\mathbf{A = LU}$
   - Dense kernels (BLAS) at each tree node (frontal matrix)
3. Solution phase:
   - Triangular solves: $\mathbf{LY = B}$, then $\mathbf{UX = Y}$
   - (possibly) Improvement of solution (iterative refinement), error analysis

Tuning the analysis phase

graph/scalar algorithms not so efficient on VE

- Offload from VE to CPU host (skylake gold 6126) serial graph-related algorithms: Metis ordering and symbolic factorization
- Use a new symbolic factorization algorithm (based on [Gilbert, Ng, Peyton, SIAM J. Matrix Anal. Appl., 1994])



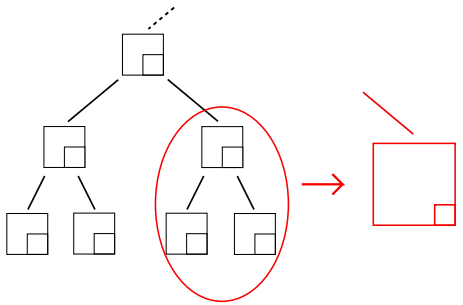
Time (seconds)

	Metis ordering		symbolic factorization			
	on VE	on host	on VE		on host	
Matrix			old	new	old	new
ecology1	37.2	4.4	3.8	0.3	0.4	0.1
G3_circuit	72.0	9.0	10.3	0.8	0.9	0.2
thermal2	52.6	7.3	6.3	0.8	0.9	0.2
atmosdl	**83.9**	**11.7**	**30.0**	**1.7**	**2.2**	**0.4**
CurlCurl_4	148.0	23.6	28.3	2.1	2.6	2.1
Serena	58.4	9.3	17.7	2.0	1.9	0.5
dielFilterV3real	30.2	5.2	4.6	2.5	0.7	0.7

Tuning the numerical phases (factorization, solve)

computationally intensive, long vectors needed

- New amalgamation algorithm: group nodes in the elimination tree $\rightarrow$ larger dense matrices, at the cost of extra computations on zeros



- Add compiler directives, e.g. to indicate vectorizable loops

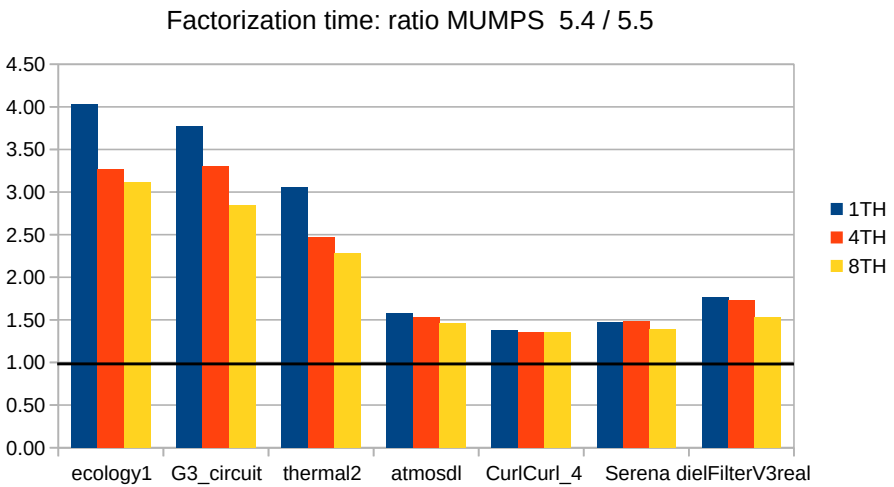
```
#if defined(__ve__)
!NEC$ IVDEP
#endif
```

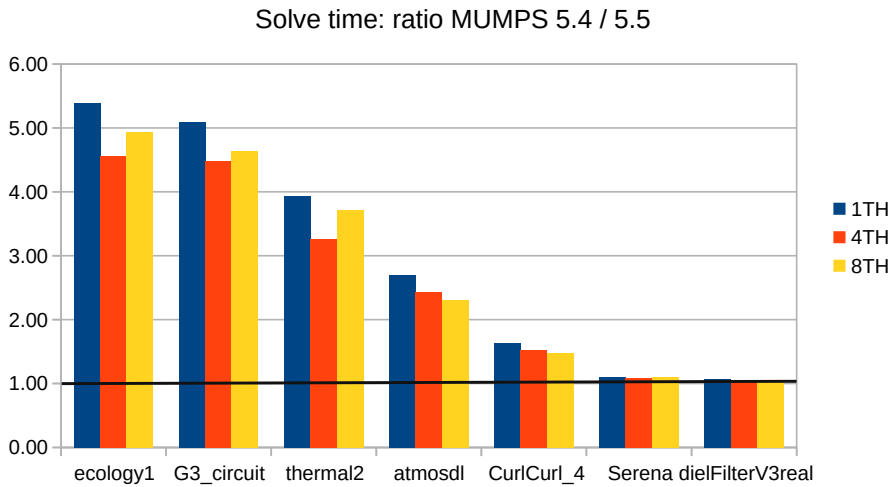
- Tune vectorization, e.g., revert loop order during LDL^T inner panel factorization, targeting large vectors in inner loop, even though stride > 1

- Play with (some) block sizes

extensive use of ftrace statistics (NEC profiler) and compiler reports

Tuning of the numerical phases on NEC SX-Aurora





(5.4=MUMPS before tuning, 5.5=MUMPS after tuning)

→ significant gains on 1, 4, 8 threads (TH), especially on the smaller matrices

Time for factor., VE 10B, **8 threads** (seconds)

	MUMPS 5.4	MUMPS 5.5
ecology1	**5.4**	**1.7**
G3_circuit	8.2	2.9
thermal2	5.1	2.2
atmosdl	16.0	10.9
CurlCurl_4	105.3	77.7
Serena	**45.2**	**32.5 (*)**
dielFilterV3real	5.7	3.7

Time for solve, VE 10B, **8 threads** (seconds)

	MUMPS 5.4	MUMPS 5.5
ecology1	**1.83**	**0.37**
G3_circuit	2.79	0.60
thermal2	1.70	0.46
atmosdl	2.10	0.91
CurlCurl_4	1.52	1.03
Serena	**0.86**	**0.79**
dielFilterV3real	0.44	0.43

(*) With GEMMT BLAS3 kernel → **27 s.** (1.1 Tflops/s out of 2.15 Tflops/s peak); VE 20B instead of 10B → **23.7 s.**; time on host (12-core Intel skylake) → 62 s.

Study in geophysics – Block Low Rank compression

Matrix	Type		N	NZ/N	complex	factors	1MPIx8threads	
	arith	facto	(1E6)		flops	GBytes	Gflops/s	Tfacto
Geo115^3	c	LU	1.52	27	3.1E13	32.2	717(*)	44 s

(*)corresponds to **2.8 Tflops/s** in single precision, real arithmetic (peak = 4.3 Tflops/s).

- 3D frequency-domain Full Waveform Inversion, 27-point stencil, domain of size 115^3 (matrix generator provided by Geoazur -- S. Operto)

Study in geophysics – Block Low Rank compression

Matrix	arith	Type facto	N (1E6)	NZ/N	complex flops	factors GBytes	1MPIx8threads Gflops/s	Tfacto
Geo115^3	c	LU	1.52	27	3.1E13	32.2	717(*)	44 s

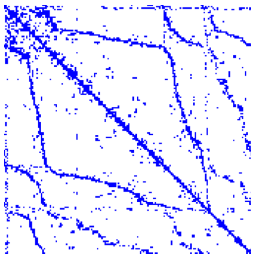
(*)corresponds to **2.8 Tflops/s** in single precision, real arithmetic (peak = 4.3 Tflops/s).

- 3D frequency-domain Full Waveform Inversion, 27-point stencil, domain of size 115^3 (matrix generator provided by Geoazur -- S. Operto)

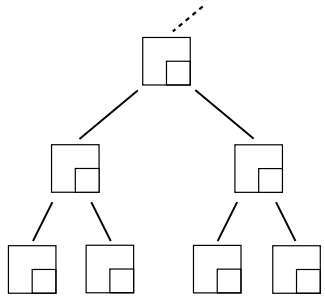
Block Low-Rank compression (BLR): $A \approx L_\varepsilon D_\varepsilon L_\varepsilon^T$

- ϵ chosen by the user, guarantees backward error of same order [Higham, Mary, IMA J.Numer.Anal., 2022], $\epsilon = 10^{-4}$ for this application [Amestoy, Brossier, Buttari, L'Excellent, Mary, Métivier, Miniussi, Operto, Geopysics, 2016]
- Complexity reduction (3D Helmholtz, $n \times n \times n$ mesh, BLR rank in $O(n)$):
  $O(n^6) \rightarrow O(n^5)$ flops
  $O(n^4) \rightarrow O(n^{3.5})$ MEMORY
  [Amestoy, Buttari, L'Excellent, Mary. On the complexity of the Block Low-Rank multifrontal factorization, SIAM J.Sci.Comput., 2017]

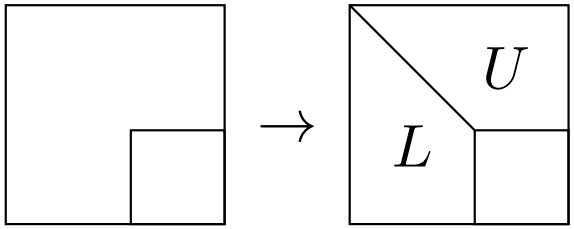
Block Low-Rank (BLR) multifrontal factorization



Sparse matrix A

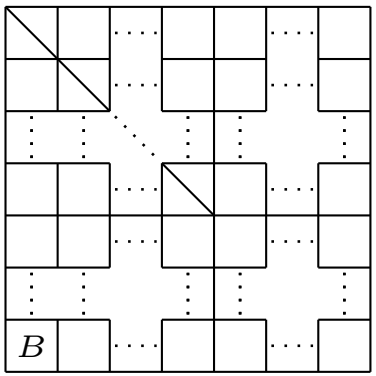


Elimination tree

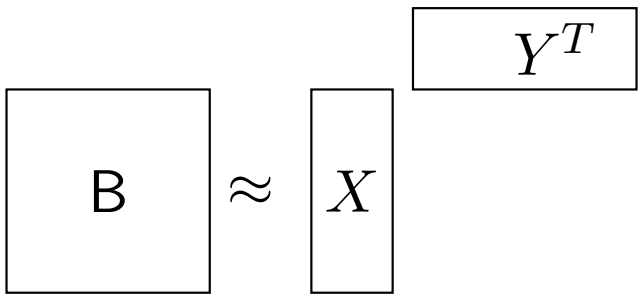


Frontal matrix at each node

(partial factorization)



Frontal matrix



Apply SVD or RRQR on block B

$\Rightarrow$ X and Y s. t.

$\|B - XY^T\| \leq \epsilon\|A\|$

Reduced storage and computations (but smaller blocks)

Block Low Rank on Geo115^3 matrix ($\epsilon = 10^{-4}$)

Size of factors (INFOG(9) in Gbytes)		
Full-rank	BLR	BLR tuned
32.2	18.3	21.0
Number of operations during facto. (E+13)		
Full-rank	BLR	BLR tuned
3.13	0.77	1.00

`BLR tuned`: exploits BLR feature only on large enough fronts

	Facto. time MUMPS 5.5.1 (sec)		
Nb TH	.	BLR	BLR tuned
1	290	342	232
4	78	108	67
8	44	70	39

	Memory eff. used (Gbytes)		
Nb TH	Full-rank	BLR	BLR tuned
1	36	25	28
4	45	31	35
8	42	33	38

	Time for solution, 1RHS (sec)		
Nb TH	Full-rank	BLR	BLR tuned
1	3.1	3.5	2.7
4	1.1	1.0	0.9
8	0.5	0.6	0.5

Summary and ongoing work

- Improvements publically available in latest MUMPS releases
- More than half the peak performance obtained on geophysics application
- Potential for Block Low-Rank (BLR) to reduce memory while preserving performance (to be further tuned)
- Address larger problems (only 48 GB per Vector Engine):
  - mixed precision BLR [PhD M. Gerest, EDF and LIP6]
  - experimentation/tuning with MPI on many VE 20B (MesoNET!)

Note: we also work on porting MUMPS to GPU's

1. larger memory on host: offload large BLAS calls to GPU's (support from Altair, use `xkblas` runtime library, T. Gautier, Inria)
2. larger memory on GPU (CINES Adastra "contrat de progrès"): use OpenMP 5.0 (target...), collaboration with CINES, GENCI, HPE, EOLEN.

   ongoing work requiring significant efforts, no BLR envisaged for the moment

Appendix: test environment

- Test matrices (suitesparse collection):

Matrix	Type facto	N (1E6)	NNZ/N	flops facto	Factors GBytes
ecology1	LDLT	1.00	5.0	1.4E+10	0.3
G3_circuit	LDLT	1.59	4.8	4.9E+10	0.8
thermal2	LDLT	1.23	7.0	1.4E+10	0.4
atmosdl	LU	1.49	6.9	1.1E+13	16.2
CurlCurl_4	LDLT	2.38	11.1	1.9E+13	19.3
Serena	LDLT	1.39	46.1	3.0E+13	22.3
dielFilterV3real	LDLT	1.10	81.0	1.1E+12	4.7

- mpinfort, compiler versions 3.2.1 $\rightarrow$ 3.5.1

- compiler options for MUMPS 5.5.1:

  -finline-functions -finline-max-depth=5 -finline-max-function-size=250
  -O2 -fpp -fopenmp -DBLR_MT [-DGEMMT_AVAILABLE]