Porting XNS code to SX-Aurora TSUBASA

Tim Cramer, Fabian Orland

High Performance Computing | IT Center | RWTH AACHEN UNIVERSITY

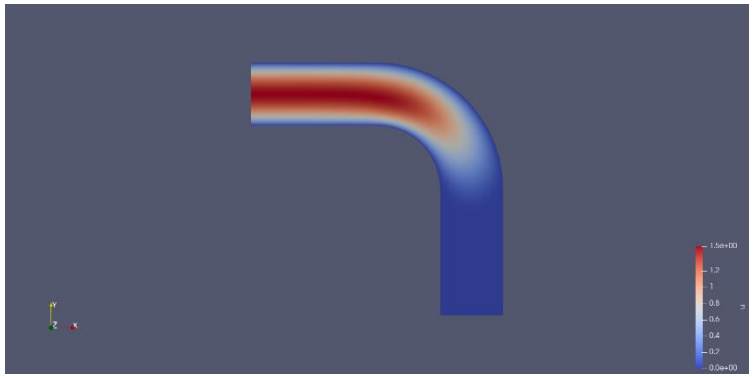
General information

XNS

- CFD code developed by CATS institute of RWTH Aachen
- Fortran code, parallelized by using MPI
- 24.529 GFLOP/s + 181 GB/s memory bandwidth (on CLAIX-2018, 48 MPI ranks)

Porting to Aurora

- New Makefile for new hosttype Aurora
- Replace old compilers by NEC compilers
- Find equivalent fortran compiler flags for nfort
- Link NEC BLAS, LAPACK libraries



XNS – Hotspots – Aurora

- 1 VE of SX-Aurora TSUBASA with 8 MPI processes

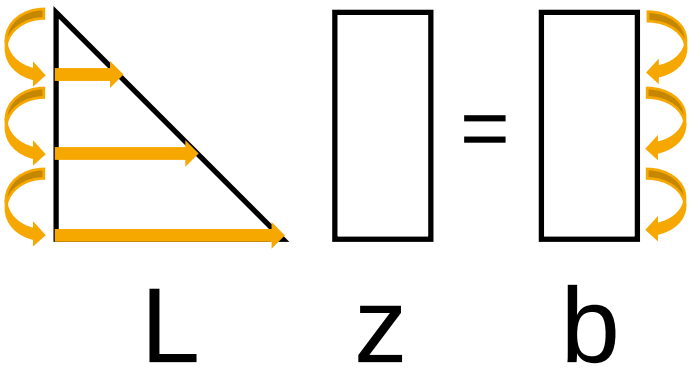
- Hotspot functions
  - lusol (43.1% of CPU time)
  - ilut (42.6% of CPU time)

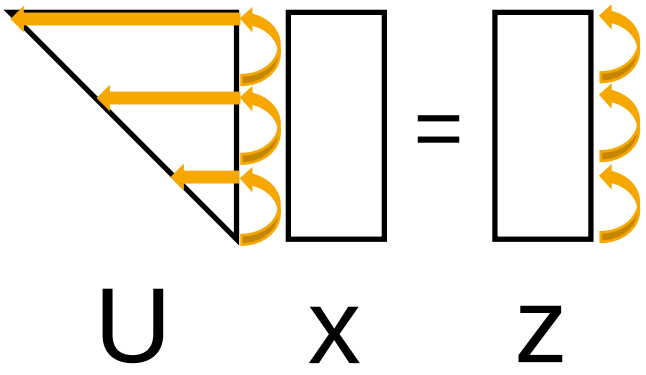
FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	L1CACHE MISS	CPU PORT CONF	VLD LLC HIT E.%	PROC.NAME
72000	3439.200 (43.1)	47.767	1476.0	299.9	0.92	254.5	0.619	1391.716	0.000	30.40	LUSOL
240	3397.203 (42.6)	14155.014	2491.7	119.1	42.81	143.7	423.584	508.158	0.000	100.00	ILUT
240	786.068 (9.9)	3275.284	2679.1	987.8	87.00	33.2	511.192	27.354	1.528	68.74	NEWD
240	157.279 (2.0)	655.330	1342.9	333.1	10.82	4.6	36.159	48.568	0.000	88.05	BLKINSRF2DST
72000	74.974 (0.9)	1.041	551.4	0.0	22.98	3.0	74.798	0.042	0.000	83.18	MAPRHS2RES
72000	73.846 (0.9)	1.026	551.2	0.0	23.34	3.0	73.818	0.023	0.000	94.69	MAPRES2RHS
14334810	21.181 (0.3)	0.001	890.0	224.7	0.00	0.0	0.000	12.992	0.000	0.00	CURRENTVALUESINS2DST
240	14.802 (0.2)	61.674	993.3	35.4	26.19	3.6	13.836	0.339	0.000	91.96	REDMATRIX
248	5.846 (0.1)	23.575	697.4	7.9	21.82	3.2	4.654	0.293	0.000	98.79	GENBC
240	1.216 (0.0)	5.065	1288.8	180.3	3.14	2.0	0.351	0.124	0.000	96.78	STABHE
15103923	7975.104(100.0)	0.528	2004.2	284.7	34.72	62.8	1140.218	1990.109	1.528	77.75	total

XNS – Hotspots – LUSOL – Background

- Solving (sparse) linear system $Ax = b$
  - $A = LU$
  - L is lower triangular
  - U is upper triangular

- Solving $LUx = b$ in two steps:
  - Substitute $Ux = z$ and solve $Lz = b$ by forward substitution
  - Solve $Ux = z$ by backward substitution





XNS – Hotspot Tuning – LUSOL V0

- Compiled with nfort, no code changes

n around 26,000
Non zeroes per row: max 149

- Forward substitution – unvectorized!
- Backward substitution – unvectorized!

- Reason: unvectorizable dependency (`x`) is assumed

```
428:                ! Forward solve.
429: S------>       do i=1,n
430: |
431: |                 x(i) = y(i)
432: |
433: |+----->          do k=jlu(i),ju(i)-1
434: ||                    x(i) = x(i) - alu(k) * x(jlu(k))
435: |+-----           end do
436: |
437: S------        end do
438:
439:                ! Backward substitution.
440: S------>       do i=n,1,-1
441: |
442: |+----->          do k=ju(i),jlu(i+1)-1
443: ||                    x(i) = x(i) - alu(k) * x(jlu(k))
444: |+-----           end do
445: |
446: |                 x(i) = alu(i) * x(i)
447: |
448: S------        end do
```

```
429: err( 504): The number of VLOAD, VSTORE.:  3,  2.
429: err( 505): The number of VGT,   VSC.   :  0,  0.
429: vec( 102): Partially vectorized loop.
431: opt(1036): Potential feedback - use directive or compiler option if OK.
433: vec( 103): Unvectorized loop.
434: opt(1036): Potential feedback - use directive or compiler option if OK.
434: opt(1019): Feedback of scalar value from one loop pass to another.
434: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: X
435: opt(1512): Loop unrolled.: K
440: vec( 102): Partially vectorized loop.
440: err( 504): The number of VLOAD, VSTORE.:  4,  5.
440: err( 505): The number of VGT,   VSC.   :  0,  0.
442: vec( 103): Unvectorized loop.
443: opt(1036): Potential feedback - use directive or compiler option if OK.
443: opt(1019): Feedback of scalar value from one loop pass to another.
443: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: X
444: opt(1512): Loop unrolled.: K
```

XNS – Hotspot – LUSOL – Tuning effects

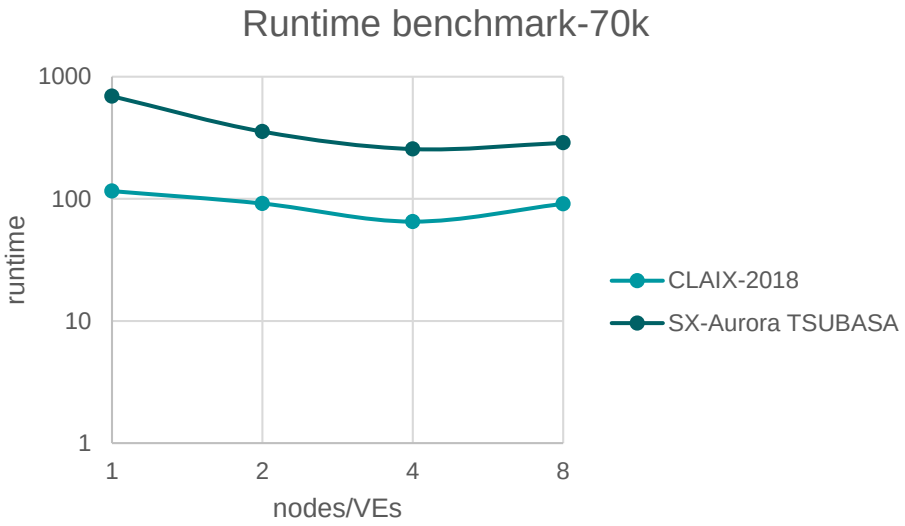
Ftrace results for each code modification step:

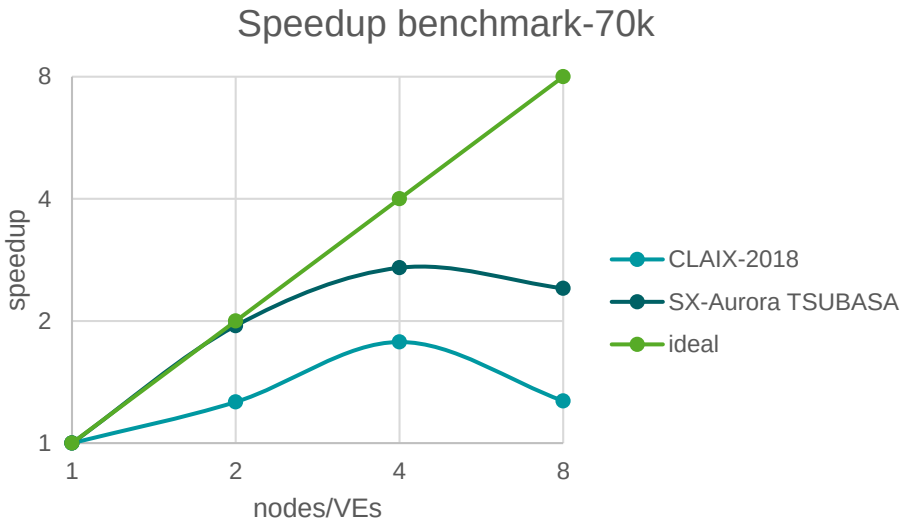
- Speedup of 2.14 by adding `!$NEC ivdep` (LUSOL-V1)
- Speedup of 3.65 by adding `!$NEC ivdep` and `!$NEC unroll(2)` (LUSOL-V2)
- Vector time is 98% (LUSOL-V1) respectively 99% (LUSOL-V2) of exclusive time
- However, shorter average vector length!

version	excl. time [sec] (%)	vec. time [sec]	MFLOPS	v. op. ratio	avg. v. len
0	3439.200 (43.1)	0.619	299.9	0.92	254.5
1	1605.522 (26.7)	1572.193	965.4	95.39	122.5
2	942.668 (17.5)	934.965	1642.1	90.77	57.5

XNS - Scalability

- On Aurora LUSOL-V2 is used
- XNS runs between 3.16 and 5.98 times faster on CLAIX-2018 than on Aurora
- Significantly better speedups on Aurora





XNS – Hotspot – ILUT – Ftrace

- Ftrace of loop structures inside ILUT
- most time spent in LOOP3 (33.6%) and LOOP2 (12.3%)
  - LOOP3 not vectorized by compiler
  - LOOP2 vectorized by compiler
- LOOP3 performs Gaussian elimination of current row with previous rows

FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	L1CACHE MISS	CPU PORT CONF	VLD LLC HIT E.%	PROC.NAME
1365208862	3322.311(33.6)	0.002	1447.8	121.1	0.00	0.0	0.000	932.931	0.000	0.00	LOOP3(+)
1939741166	1211.027(12.3)	0.001	3524.9	0.0	84.62	143.9	412.958	586.170	0.000	100.00	LOOP2(V)
6200700	17.551(0.2)	0.003	800.1	0.0	0.00	0.0	0.000	3.307	0.000	0.00	LOOP4(+)
6200700	13.473(0.1)	0.002	2140.7	0.5	0.00	0.0	0.000	1.710	0.000	0.00	LOOP6(+)
6200700	4.950(0.1)	0.001	1488.1	0.0	0.00	0.0	0.000	2.119	0.000	0.00	LOOP1(+)
6200700	3.170(0.0)	0.001	2811.1	0.0	74.75	134.5	0.959	1.615	0.000	100.00	LOOP5(V)
6200700	2.184(0.0)	0.000	1464.1	232.5	39.70	40.9	0.864	0.945	0.000	96.33	LOOP0(V)
6200700	0.929(0.0)	0.000	5967.1	0.0	63.75	142.5	0.691	0.371	0.000	100.00	LOOP7(V)

XNS – Hotspot – ILUT – Diagnosis

```
628: |                ! Combine current row and row jrow.
629: |S----->         do k=ju(jrow),jlu(jrow+1)-1
630: ||
631: ||                   s = fact * alu(k)
632: ||                   j = jlu(k)
633: ||                   jpos = jw(n+j)
634: ||
635: ||                   if (j.ge.ii) then
636: ||
637: ||                      ! Dealing with upper part.
638: ||                      if (jpos.eq.0) then
639: ||                         ! This is a fill-in element.
640: ||                         lenu = lenu + 1
641: ||                         !if (lenu.gt.n) goto 994
642: ||                         i = ii + lenu - 1
643: ||                         jw(i) = j
644: ||                         jw(n+i) = i
645: ||                         w(i) = - s
646: ||                      else
647: ||                         ! This is not a fill-in element.
648: ||                         w(jpos) = w(jpos) - s
649: ||                      end if
650: ||
651: ||                   else
652: ||
653: ||                      ! Dealing  with lower part.
654: ||                      if (jpos.eq.0) then
655: ||                         ! This is a fill-in element.
656: ||                         lenl = lenl + 1
657: ||                         !if (lenl.gt.n) goto 995
658: ||                         jw(lenl) = j
659: ||                         jw(n+i) = lenl
660: ||                         w(lenl) = - s
661: ||                      else
662: ||                         ! This is not a fill-in element.
663: ||                         w(jpos) = w(jpos) - s
664: ||                      end if
665: ||
666: ||                   end if
667: |S-----            end do
```

Vectorization issues:

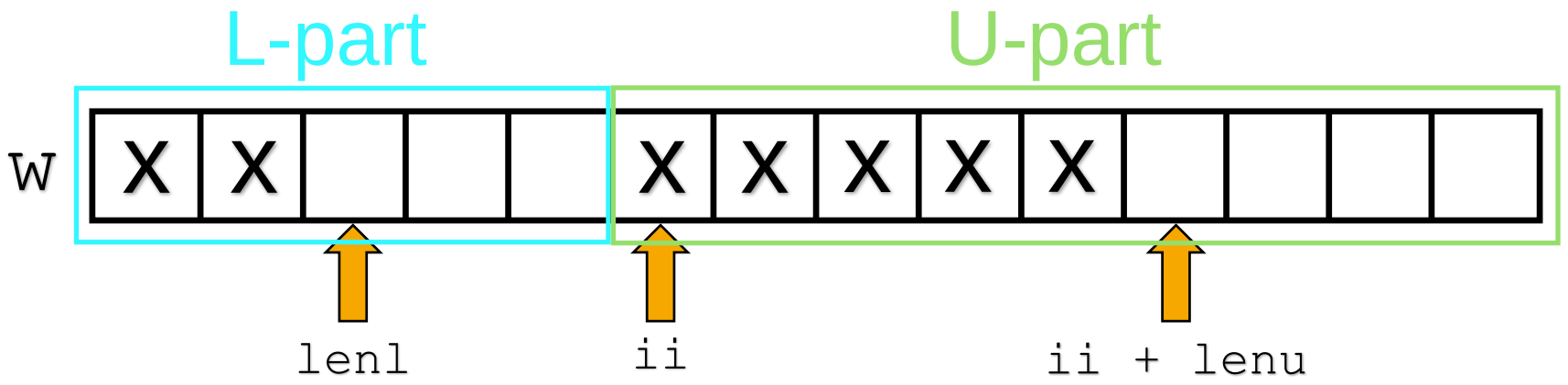
- `goto` prohibits vectorization
  - Error checking for correct matrix dimensions
  - commented out (for now)
- increment of `lenu, lenl`
  - loop iterations not independent
- Unvectorizable dependency `JW, W`
  - Indexing depends on `lenu, lenl`

```
629: vec( 102): Partially vectorized loop.
633: opt(1036): Potential feedback - use directive or compiler option if OK.
640: opt(1019): Feedback of scalar value from one loop pass to another.: LENU
643: opt(1033): Potential multiple store conflict -- use directive if OK.
643: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: JW
644: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: JW
645: opt(1033): Potential multiple store conflict -- use directive if OK.
645: opt(1036): Potential feedback - use directive or compiler option if OK.
645: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: W
648: opt(1036): Potential feedback - use directive or compiler option if OK.
648: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: W
656: opt(1019): Feedback of scalar value from one loop pass to another.: LENL
658: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: JW
659: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: JW
660: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: W
663: vec( 122): Dependency unknown. Unvectorizable dependency is assumed.: W
```

XNS – Hotspot – ILUT – Diagnosis

- ILUT uses two working arrays
  - `w(1:n)` stores non-zero values of current row `(jrow)`
  - `jw(1:n)` stores column index of non-zero values in `w`
  - `jw(n+1:2n)` stores non-zero indicator to identify fill-in elements

- Structure of working arrays
  - LU factorization is stored in place
  - `lenl` = current length of L-part
  - `lenu` = current length of U-part
  - `ii` = index of current row / index of diagonal element



XNS – Hotspot – ILUT – GMRES

- Matrix data:
  - Dimension: 203,514 x 203,514
  - Non zero entries: 8,407,204

- Libray of Iterative Solvers for Linear Systems (LIS)
  - 242 iterations until convergence
  - CSR format:
    - Preconditioner: 21.2% faster on Aurora (~23 sec)
    - Linear solver: 52.7% faster on Aurora (~19 sec)
  - ELLPACK format:
    - Preconditioner: 21.3% faster on Aurora (~23 sec)
    - Linear solver: 32x faster on Aurora (~1 sec)

- NEC‘s Advanced Scientific Library (ASL)
  - only supports ELLPACK format
  - 97634 iterations until convergence
    - Simple scaling as preconditioner (no ILUT)
  - Runtime: 5-9 minutes
  - Multithreading did not work

Conclusion

- XNS is a memory bound code
- Hotspot LUSOL
  - Speedup of 3.65 by adding compiler directives
  - Poor vector length of 57.5
- Scalability CLAIX-2018 vs Aurora
  - Still 3.16x - 5.98x slower on Aurora than on CLAIX after tuning
- Hotspot ILUT
  - Compiler directives not enough to enable vectorization
  - LIS library:
    - Preconditioner: comparable results on CLAIX-2018 and Aurora
    - Linear solver of GMRES up to 32x using ELLPACK format
  - ASL library:
    - Not competetive (5-9 minutes runtime)
    - No speedup from multithreading

Thank you
for your attention!