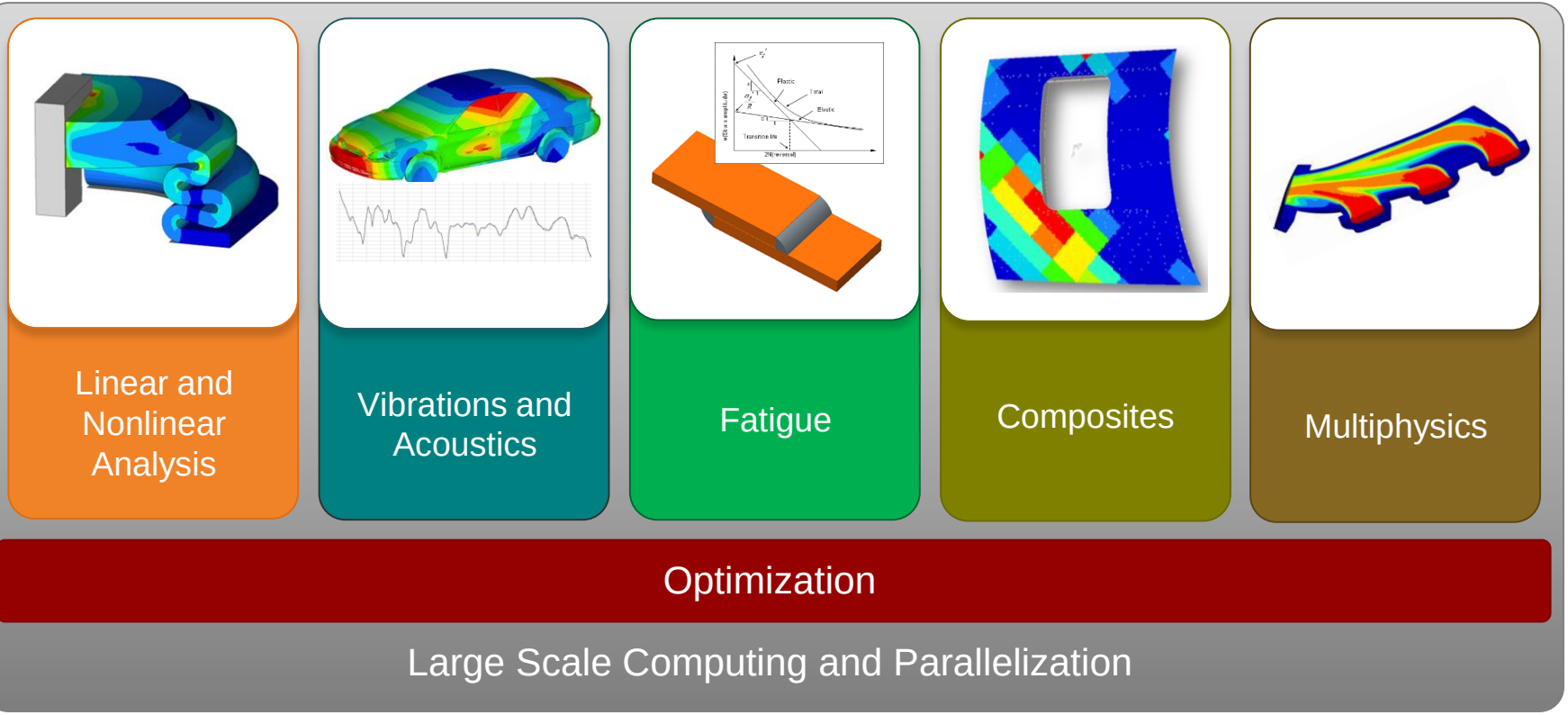
MUMPS v 5.1.1

Kostas Sikelis, Phd
Software Development specialist, Optistruct
MUMPS User Days, June 2017

Outline

- Optistruct - Overview
- Comparison of MUMPS 5.1.1 vs 5.0.1
- 64 bit MUMPS
- BLR/ABLR – Preliminary results
- Wish list

OptiStruct - Overview



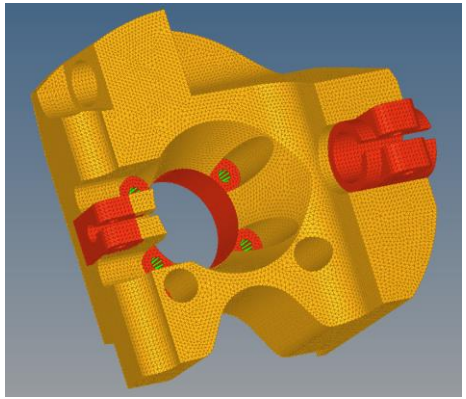
Optistruct – MUMPS usage

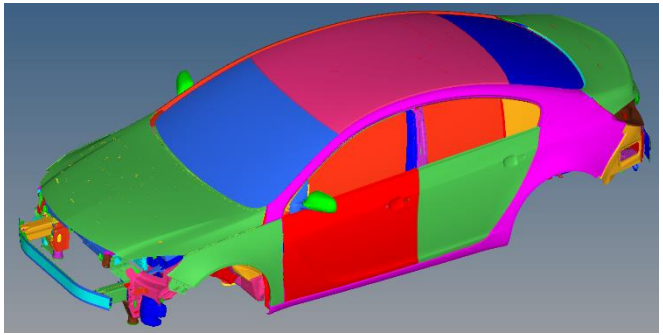
- **Linear static analysis**
  - Symmetric positive definite or indefinite systems
- **Nonlinear static/transient analysis**
  - Unsymmetric systems
- **Lanczos Eigensolver**
  - Symmetric indefinite systems
- **Direct Frequency response Analysis**
  - Symmetric complex systems

Optistruct – MUMPS Benchmark

- **Incore** linear static analysis.

Model	# of Equations	# of Nonzeros	# of Elements	Element Type
Knuckle	2.8M	117M	650K	Solid (CTETRA10)
Car body	12M	320M	1.8M	Shell (CQUAD4/CTRIA3)
			366K	Solid (CTETRA/CPENTA/CHEXA





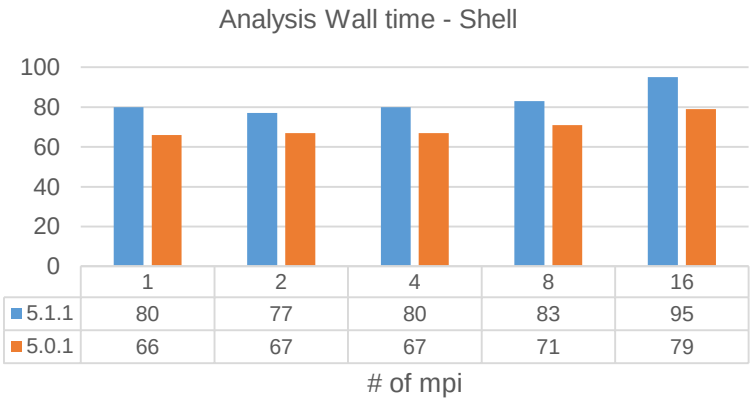
Optistruct – MUMPS Benchmark

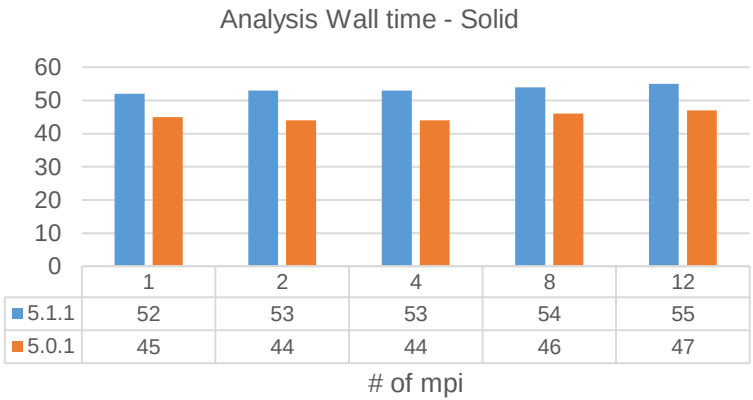
Data extracted from MUMPS report

- **Time**
  - ELAPSED TIME IN ANALYSIS DRIVER
  - ELAPSED TIME IN FACTORIZATION DRIVER
  - ELAPSED TIME IN SOLVE DRIVER
- **Total Memory**
  - TOTAL space in MBYTES for IC factorization
  - ** EFF Min: Avg. Space in MBYTES per working proc (x # of MPI-Processes)
  - ** Avg. Space in MBYTES per working proc during solve (x # of MPI-Processes)

MUMPS in Optistruct – 5.1.1 vs 5.0.1

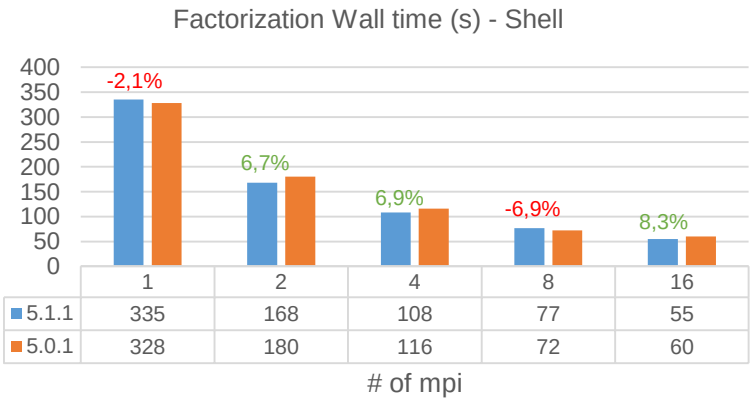
- Slight performance drop in Analysis phase due to 64bit METIS
- Improved MPI scaling both in Factorization and Solve phases
- Signifficantly improved SMP scaling both in Factorization and Solve phases

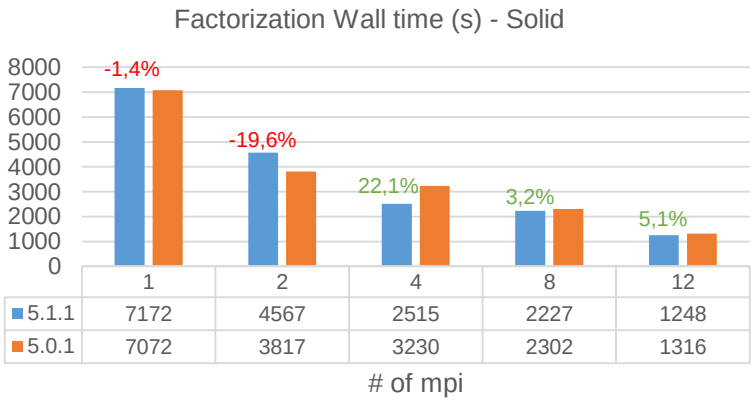




MUMPS in Optistruct – 5.1.1 vs 5.0.1

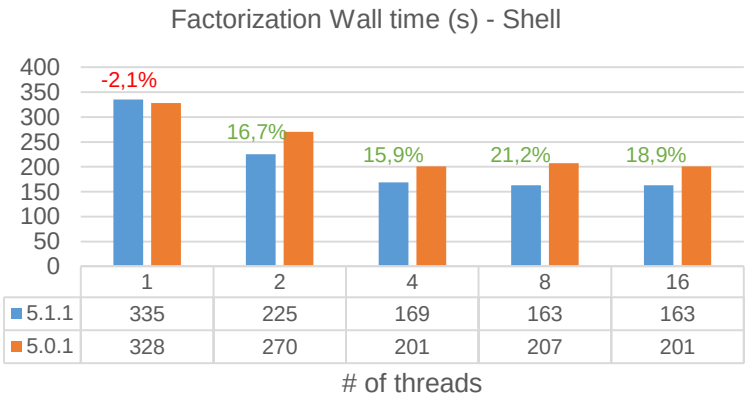
- Slight performance drop in Analysis phase due to 64bit METIS
- Improved MPI scaling both in Factorization and Solve phases
- Signifficantly improved SMP scaling both in Factorization and Solve phases

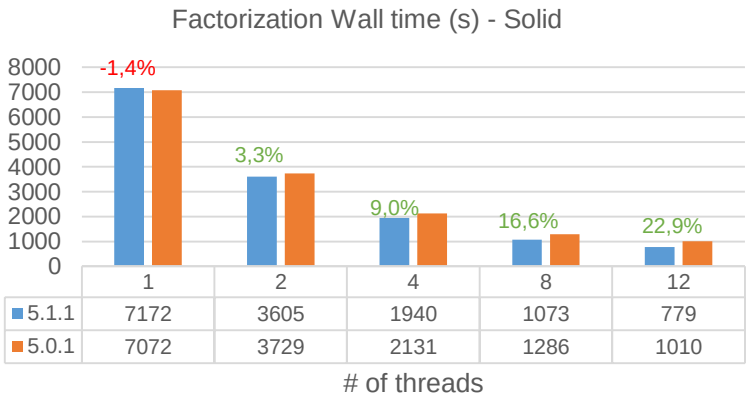




MUMPS in Optistruct – 5.1.1 vs 5.0.1

- Slight performance drop in Analysis phase due to 64bit METIS
- Improved MPI scaling both in Factorization and Solve phases
- Signifficantly improved SMP scaling both in Factorization and Solve phases (even more potential with aggressive setting)





Optistruct – MUMPS 64bit (full)

- Motivation
  - Industrial models challenge 32bit integer capacity
  - Quick solution: Promote all integers to 64

- MPI-Free Implementation
  - Compile Fortran source forcing all integers to be 64bit (-i8 or -fdefault-integer-8 in OPTF)
  - Compile C source forcing all integers to be 64bit (-DINTSIZE64 in OPTC)
  - Link with 64bit BLAS libraries

- MPI : Intel supports 64 bit integer mpi with option –ilp64 in mpirun command, but with the following limitation
  - In REDUCE operations MPI_2INTEGER data type is not promoted, send and receive buffers must be INTEGER(4)
  - For custom reduction functions used in REDUCE operations the integer arguments must be INTEGER(4)

Optistruct – MUMPS 64bit (full)

i. bcast_errors.F@16 : INTEGER IN(2), OUT(2) --> INTEGER(4) IN(2), OUT(2)

ii. tools_common.F@269 : INTEGER TEMP1(2), TEMP2(2) --> INTEGER(4) TEMP1(2), TEMP2(2)

iii. [dzsc]fac_scalings_simScale_util.F@23 : INTEGER IWRK(IWSZ) --> INTEGER(4) IWRK(IWSZ)

iv. [dzsc]fac_scalings_simScale_util.F@433-436: INTEGER LEN, INV(2*LEN), INOUTV(2*LEN), DTYPE --> INTEGER(4) LEN, INV(2*LEN), INOUTV(2*LEN), DTYPE

v. [dzsc]fac_scalings_simScale_util.F@466 : INTEGER IW(IWSZ) --> INTEGER(4) IW(IWSZ)

vi. [dzsc]fac_scalings_simScale_util.F@923 : INTEGER IWRK(IWSZ) --> INTEGER(4) IWRK(IWSZ)

vii. [dzsc]mumps_driver.F@1642 : INTEGER TMP1(2),TMP(2) --> INTEGER(4) TMP1(2),TMP(2)

- MUMPS 5.0.1 : Apply above changes manually
- MUMPS 5.1.1 : -DWORKAROUNDINTELILP64MPI2INTEGER

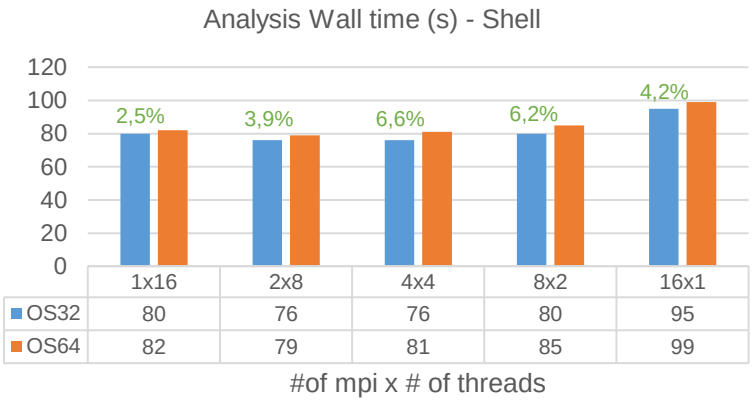
Optistruct – MUMPS 64 bit (selective)

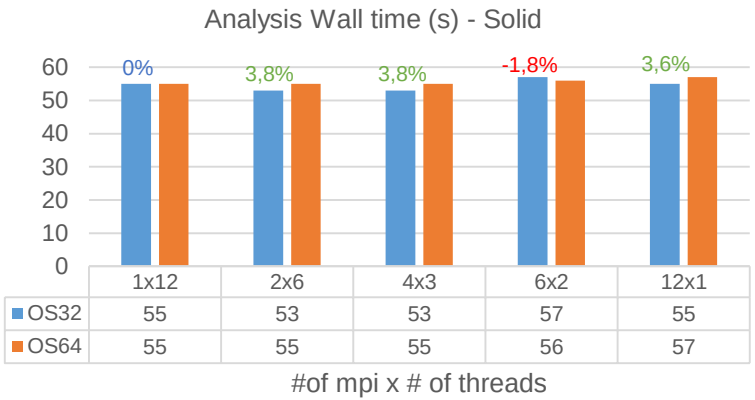
- MUMPS 5.0.1
- NZ -> 32 bit
  - Stiffness definitely less than 2 billion
- 32bit integer for internal symbolic analysis
  - No more than 1 billion terms
  - Even smaller model for unsymmetric matrix from nonlinear with friction
- Interface to METIS 32bit API
  - Need scotch sometimes

- **MUMPS 5.1.1**
- **NZ -> 32 bit, NNZ -> 64bit**
  - Backward compatible
  - Solve stiffness with more than 2 billion terms if NNZ is used
- **64bit integer for internal symbolic analysis**
  - Solve more than 1 billion terms
- **Interface to METIS 64bit API**
  - Avoid METIS overflow
  - Better performance

MUMPS in Optistruct – selective 64 bit vs full

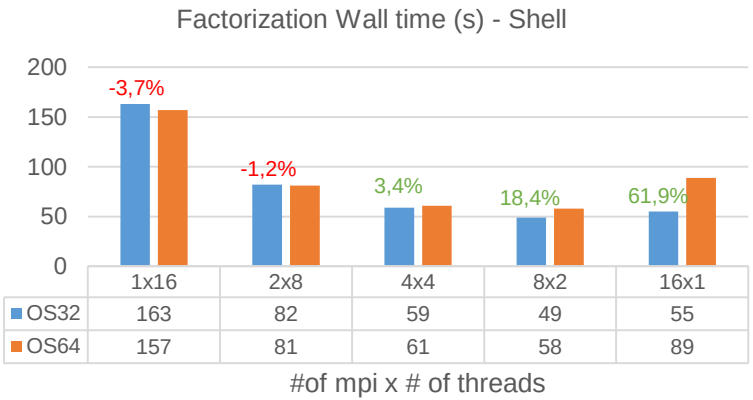
- Analysis phase is a bit faster
- Factorization phase occasionally much faster.
- Solve phase a bit slower.
- Considerable reduction in Total Memory.

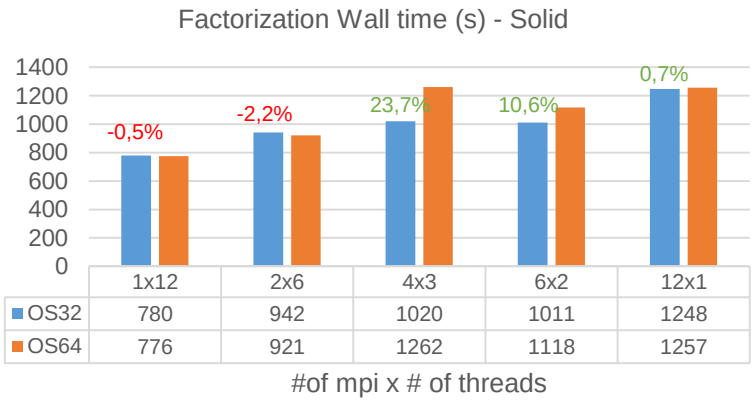




MUMPS in Optistruct – selective 64 bit vs full

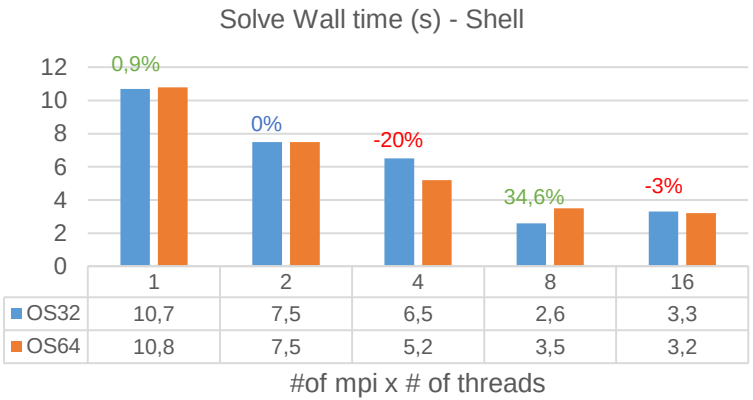
- Analysis phase is a bit faster
- Factorization phase occasionally much faster.
- Solve phase a bit slower.
- Considerable reduction in Total Memory.

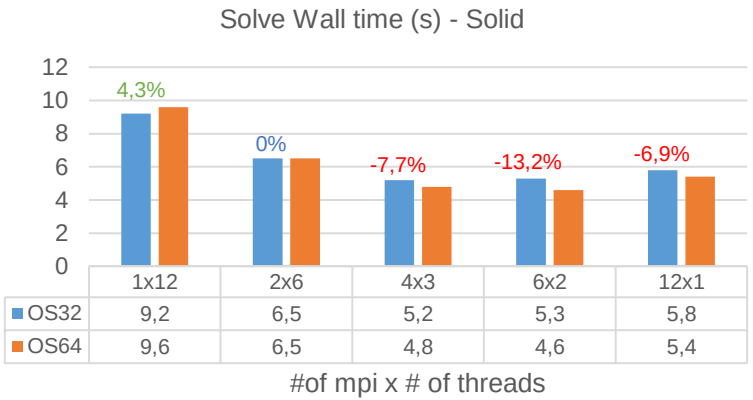




MUMPS in Optistruct – selective 64 bit vs full

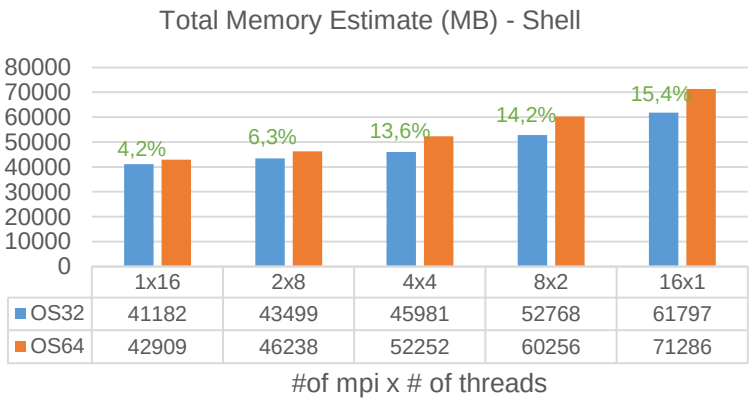
- Analysis phase is a bit faster
- Factorization phase occasionally much faster.
- Solve phase a bit slower.
- Considerable reduction in Total Memory.

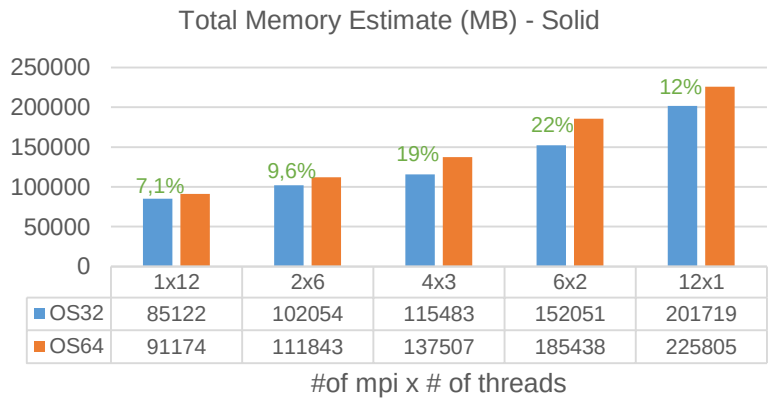




MUMPS in Optistruct – selective 64 bit vs full

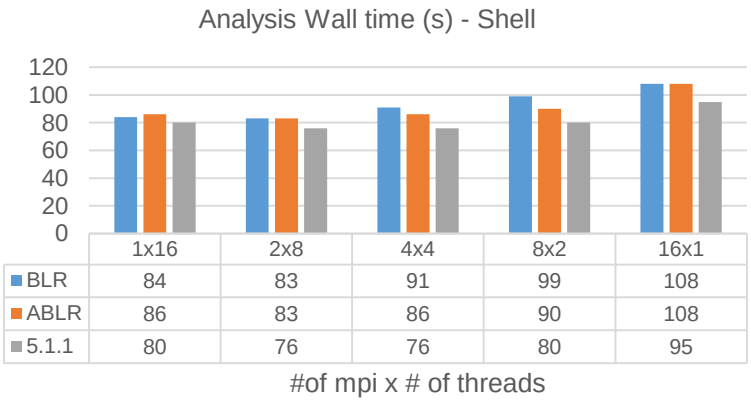
- Analysis phase is a bit faster
- Factorization phase occasionally much faster.
- Solve phase a bit slower.
- Considerable reduction in Total Memory.



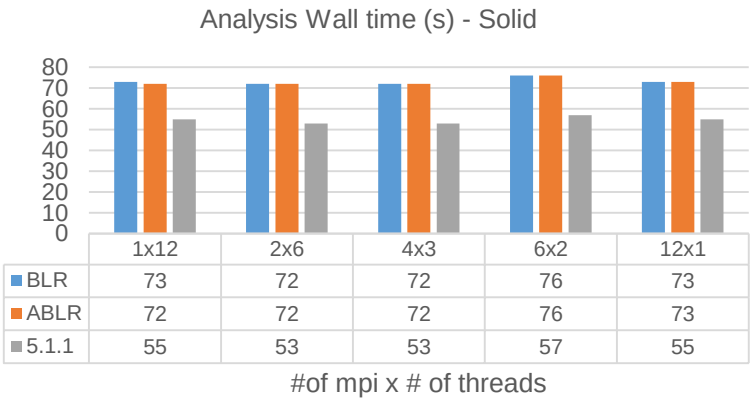


MUMPS in Optistruct – BLR/ABLR

- Analysis phase is a bit slower (for solid).
- Factorization and Solve phases up to 3 times faster (for solid)
- Reduced accuracy of the solution. Solution changes with the number of MPI-processes



	1x16	2x8	4x4	8x2	16x1
BLR	84	83	91	99	108
ABLR	86	83	86	90	108
5.1.1	80	76	76	80	95

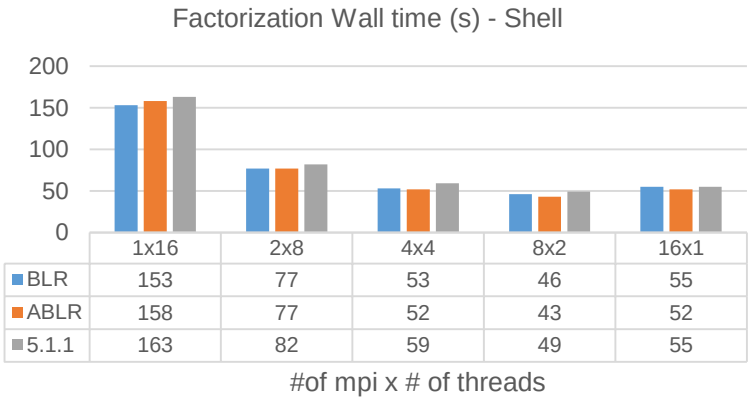


	1x12	2x6	4x3	6x2	12x1
BLR	73	72	72	76	73
ABLR	72	72	72	76	73
5.1.1	55	53	53	57	55

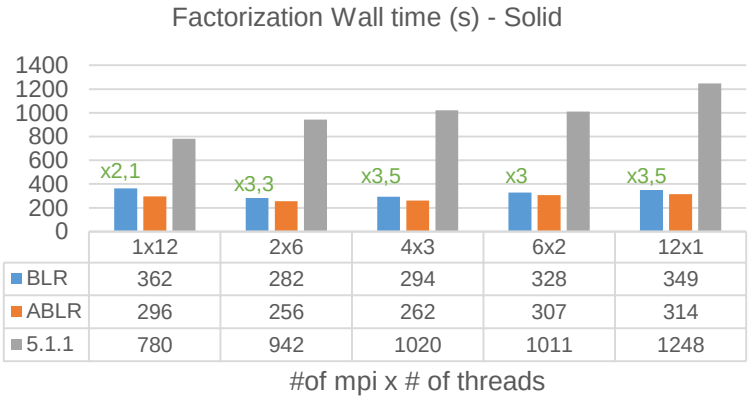
Altair

MUMPS in Optistruct – BLR/ABLR

- Analysis phase is a bit slower (for solid).
- Factorization and Solve phases up to 3 times faster (for solid)
- Reduced accuracy of the solution. Solution changes with the number of MPI-processes



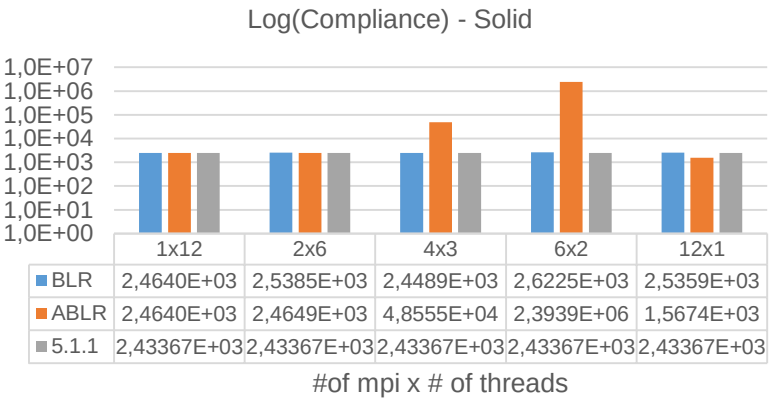
	1x16	2x8	4x4	8x2	16x1
BLR	153	77	53	46	55
ABLR	158	77	52	43	52
5.1.1	163	82	59	49	55



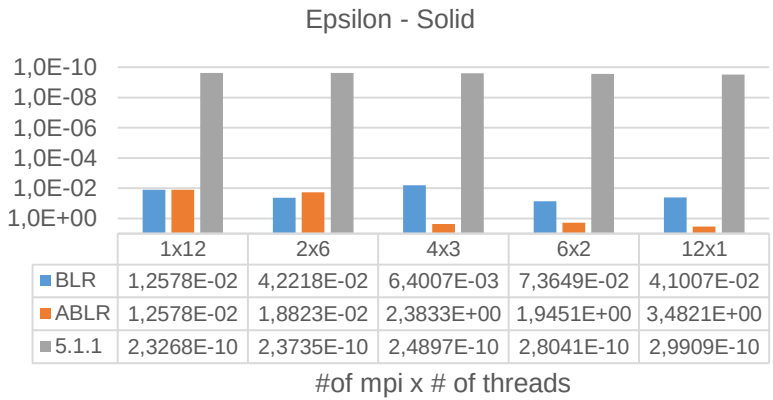
	1x12	2x6	4x3	6x2	12x1
BLR	362	282	294	328	349
ABLR	296	256	262	307	314
5.1.1	780	942	1020	1011	1248

MUMPS in Optistruct – BLR/ABLR

- Analysis phase is a bit slower (for solid).
- Factorization and Solve phases up to 3 times faster (for solid)
- Reduced accuracy of the solution. Solution changes with the number of MPI-processes



	1x12	2x6	4x3	6x2	12x1
BLR	2,4640E+03	2,5385E+03	2,4489E+03	2,6225E+03	2,5359E+03
ABLR	2,4640E+03	2,4649E+03	4,8555E+04	2,3939E+06	1,5674E+03
5.1.1	2,43367E+03	2,43367E+03	2,43367E+03	2,43367E+03	2,43367E+03



	1x12	2x6	4x3	6x2	12x1
BLR	1,2578E-02	4,2218E-02	6,4007E-03	7,3649E-02	4,1007E-02
ABLR	1,2578E-02	1,8823E-02	2,3833E+00	1,9451E+00	3,4821E+00
5.1.1	2,3268E-10	2,3735E-10	2,4897E-10	2,8041E-10	2,9909E-10

Optistruct – Wish list

- **Robust detection of ill-conditioned (near-singular) matrix**
  - Often MUMPS provides physically meaningless solutions to ill-conditioned systems.
  - Preferable just error out.
  - Possibly a parameter similar to **MAXRATIO** used for LDL^T i.e $\boldsymbol{r = max(\frac{Diag(A)_{ii}}{D_i})}$

```
-------------------- array : disp-vec1
      1)  -7.90004E+09 -2.38407E+09  0.00000E+00  0.00000E+00  0.00000E+00
         -4.55055E+07 -7.44499E+09 -2.38407E+09  0.00000E+00  0.00000E+00
     11)   0.00000E+00 -4.55055E+07 -6.98993E+09 -2.38407E+09  0.00000E+00
           0.00000E+00  0.00000E+00 -4.55055E+07 -6.53488E+09 -2.38407E+09
     21)   0.00000E+00  0.00000E+00  0.00000E+00 -4.55055E+07 -6.07982E+09
         -2.38407E+09  0.00000E+00  0.00000E+00  0.00000E+00 -4.55055E+07
     31)  -5.62477E+09 -2.38407E+09  0.00000E+00  0.00000E+00  0.00000E+00
         -4.55055E+07 -5.16971E+09 -2.38407E+09  0.00000E+00  0.00000E+00
     41)   0.00000E+00 -4.55055E+07 -4.71466E+09 -2.38407E+09  0.00000E+00
           0.00000E+00  0.00000E+00 -4.55055E+07 -4.25961E+09 -2.38407E+09
    611)   0.00000E+00 -4.55055E+07 -6.07982E+09 -1.47396E+09  0.00000E+00
           0.00000E+00  0.00000E+00 -4.55055E+07 -6.53488E+09 -1.47396E+09
    621)   0.00000E+00  0.00000E+00  0.00000E+00 -4.55055E+07 -6.07982E+09
         -1.92901E+09  0.00000E+00  0.00000E+00  0.00000E+00 -4.55055E+07
    631)  -6.53488E+09 -1.92901E+09  0.00000E+00  0.00000E+00  0.00000E+00
         -4.55055E+07 -6.07982E+09 -1.01890E+09  0.00000E+00  0.00000E+00
    641)   0.00000E+00 -4.55055E+07 -6.53488E+09 -1.01890E+09  0.00000E+00
           0.00000E+00  0.00000E+00 -4.55055E+07 -6.07982E+09 -5.63848E+08
    651)   0.00000E+00  0.00000E+00  0.00000E+00 -4.55055E+07 -6.53488E+09
         -5.63848E+08  0.00000E+00  0.00000E+00  0.00000E+00 -4.55055E+07
```

Altair

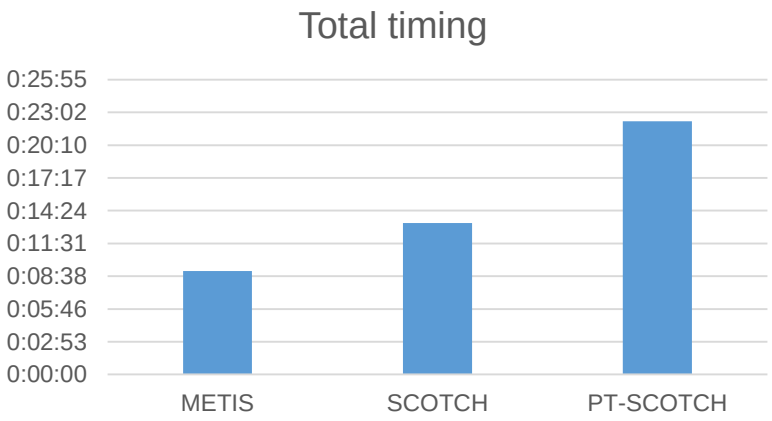
Optistruct – Wish list

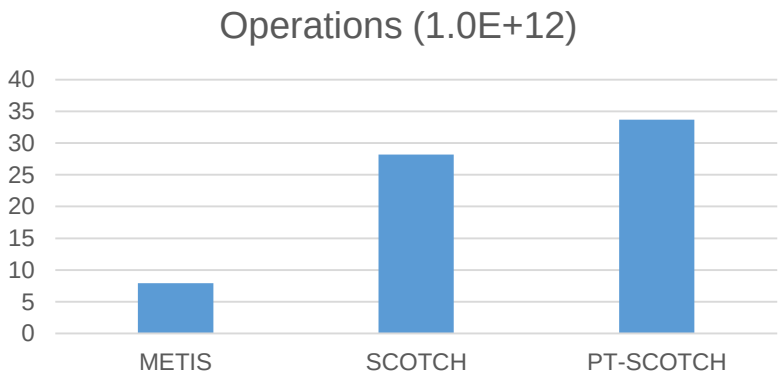
- **Detection of NaN in input matrix**
  - A failsafe in case input is erroneous.
  - Preferable to error out instead of delivering an erroneous solution

```
STATISTICS PRIOR SOLVE PHASE     ...........
NUMBER OF RIGHT-HAND-SIDES                    =           1
BLOCKING FACTOR FOR MULTIPLE RHS              =           1
ICNTL (9)                                     =           1
 --- (10)                                     =           0
 --- (11)                                     =           0
 --- (20)                                     =           0
 --- (21)                                     =           0
 --- (30)                                     =           0


LEAVING SOLVE (MPI41C) WITH
RHS    (first column)
  0.000000D+00  0.000000D+00  0.000000D+00  0.000000D+00  0.000000D+00
  0.000000D+00  0.000000D+00  0.000000D+00  0.000000D+00  0.000000D+00


VECTOR SOLUTION FOR COLUMN            1
RHS
           NaN           NaN           NaN           NaN           NaN
           NaN           NaN           NaN           NaN           NaN
** Rank of processor needing largest memory in solve     :          0
** Space in MBYTES used by this processor for solve      :        174
** Avg. Space in MBYTES per working proc during solve    :        174
```

Optistruct – Wish list

- **Suggestion of optimal reordering**





THANK YOU

Questions?

MUMPS Team, thank you for a great linear equation solver!