A Tale of Two Codes

Patrick Amestoy, Cleve Ashcraft, Roger Grimes,
Jean-Yves L'Excellent, Bob Lucas, Theo Mary,
Francois-Henry Rouet, Clement Weisbecker

Sept. 7, 2017

LSTC
Livermore Software
Technology Corp.

MUMPS Solver
mumps-consortium.org

Charles Dickens Quote

" It was the best of times, it was the worst of times,
it was the age of wisdom, it was the age of foolishness,
it was the epoch of belief, it was the epoch of incredulity,
it was the season of Light, it was the season of Darkness,
it was the spring of hope, it was the winter of despair,
we had everything before us, we had nothing before us,
we were all going direct to Heaven, we were all going direct
the other way—in short, the period was so far like the present
period, that some of its noisiest authorities insisted on
its being received, for good or for evil, in the superlative
degree of comparison only.”

Charles Dickens, A Tale of Two Cities, 1859

We were all once young and impressionable

- Roger and Cleve met Iain (and John) in the early 1980s
  - Boeing started its BCSLIB-EXT multifrontal code mid-1980s
- Patrick and Chiara joined Iain at CERFACS in 1987 and 1988
  - LU and QR multifrontal codes

- Gene Golub introduced me to Iain (~Christmas, 1985)
  - Multifrontal LU on an Intel hypercube
- Jean-Yves was a “trainee” at CERFACS in 1992
  - Started on MUMPS in 1996

Outline

- Introduction
- Triangles vs. squares
- L vs. U
- Static vs. dynamic scheduling
- Summary

Two multifrontal codes

- MUMPS

  Free software (CeCILL-C license) from CERFACS, CNRS, ENS Lyon, INP Toulouse, Inria, and Université de Bordeaux

- mf2

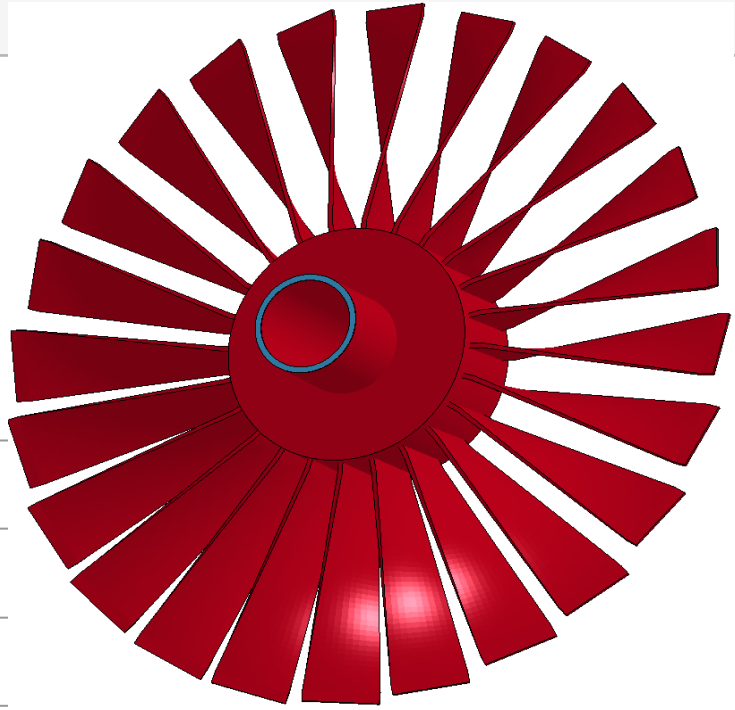
  LSTC's second distributed memory multifrontal solver
  We also have BCSLIB-EXT (SMP) and MUMPS (research)

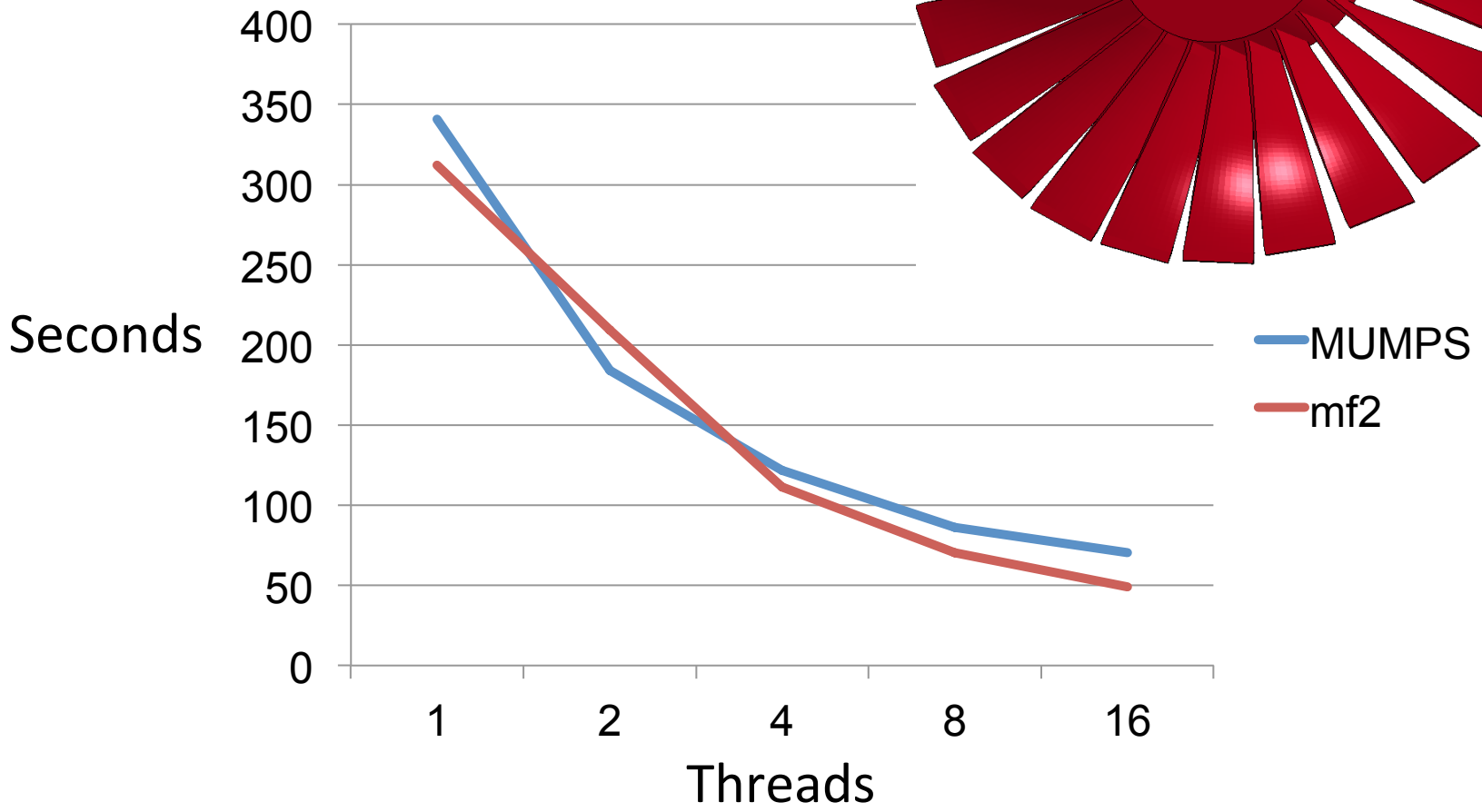
- Many similarities:

  Three decades of history
  O(100K) SLOCs
  MPI and OpenMP
  Out-of-core option
  Real and complex arithmetic
  Symmetric indefinite
  LU with symmetric structure
  Block Low Rank approximations

Similar behavior (OpenMP)

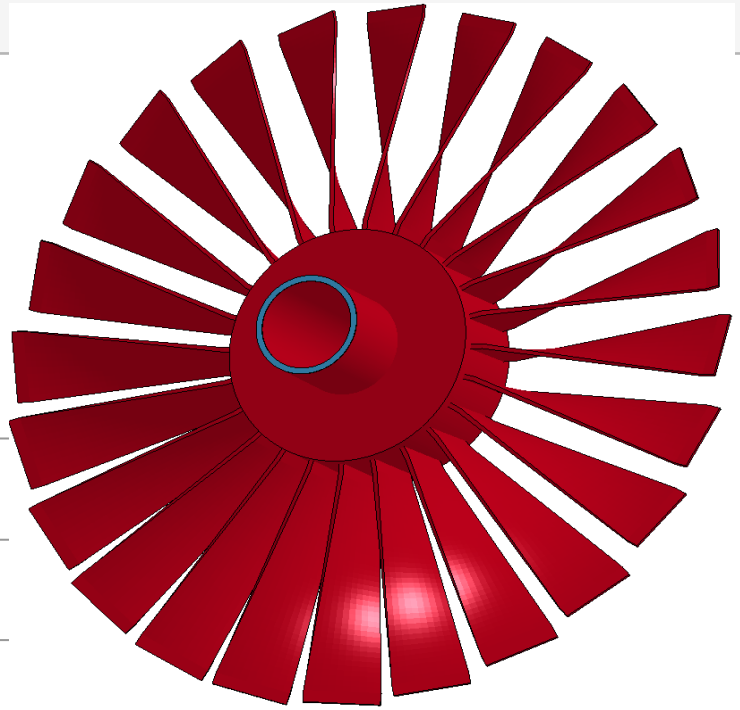
2.8M equations
106M non-zeros in A
2.89G non-zeroes in L
11.2 Tops

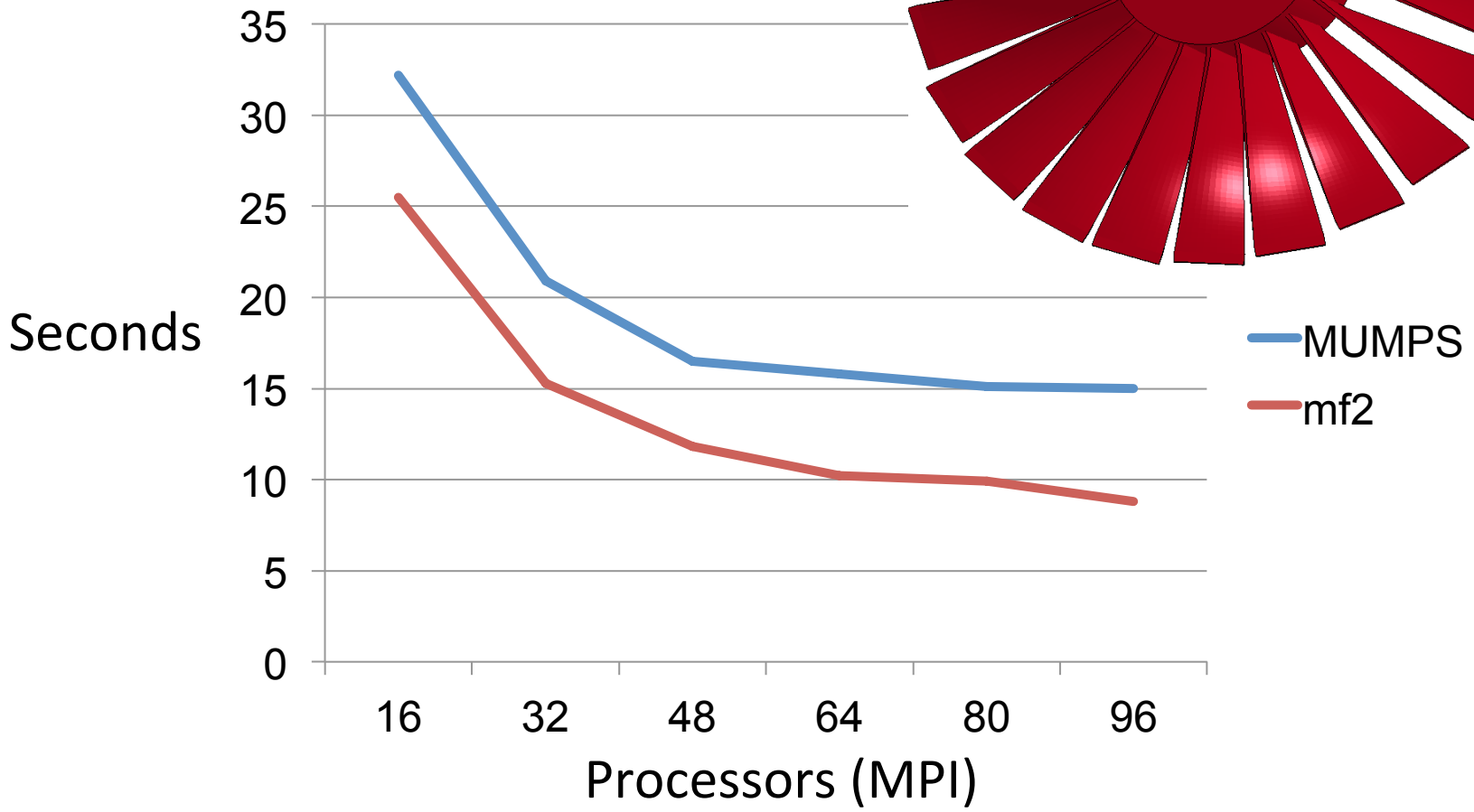




Similar behavior (MPI)

2.8M equations
106M non-zeros in A
2.89G non-zeroes in L
11.2 Tops





There are also many differences

- Heuristics are required

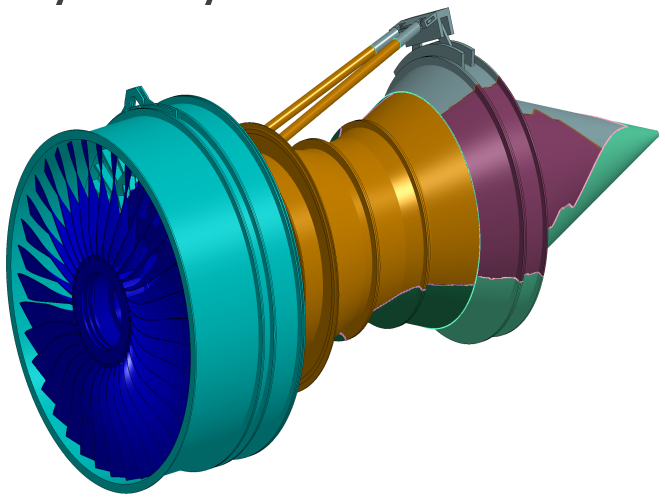
  Reordering is NP-complete (Yannakakis, 1981)
  Scheduling is NP-hard (Garey, 1976)

- Execution model evolves

  Vector mainframes -> Cache-based microprocessors
  Shared memory -> Distributed memory -> Hybrid
  -> Accelerators

- The problems keep changing

  They keep getting bigger



Rolls Royce small dummy engine model
200M degrees of freedom

Choices have to be made

- Are frontal matrices triangles or squares?
- If its symmetric, is it LDL^T or U^TDU?
- Is parallel scheduling static or dynamic?
- Do you scale the input matrix?
- What pivot choices should you make?
- How do you amalgamate supernodes?
- Multi-threaded BLAS, yes or no?

Outline

- Introduction
- Triangles vs. squares
- L vs. U
- Static vs. dynamic scheduling
- Summary

Squares -> Triangles

- 20 years ago, “mf1” solver had square frontal matrices

  It’s easier to think about

  A(i,j) = x

  than

  jm1 = j – 1

  A(ld * jm1 – (jm1 * jm1 – jm1)/2 + (i – jm1)) = x

- Seemed wasteful of memory

  Converted to using triangles of L for symmetric matrices

Unrolled loops -> DGEMM

- Unrolling *k* and *j* loops by 2 and vectorizing *i* yielded asymptotic, right-looking performance on Cray Y-M/P
  - Right-looking is attractive when pivoting
- Limited memory bandwidth of earlier cache-based systems required more unrolling
  - 6x4 in mf2, intended for Itanium and Power
- SIMD extensions (e.g., AVX) require BLAS
  - Compiled Fortran performance has effectively plateaued

Triangles -> Squares (even though symmetric)

- But DGEMM operates on rectangles!
  - mf2 converts a panel of a triangle to a rectangle
  - Performs updates with DGEMM
  - Accumulates rectangular output back into the triangle
- MUMPS keeps active fronts in rectangular panels
  - DGEMMs applied in-place
  - No overhead for copying to and from triangles
- mf2 has adopted this strategy
  - If storage allows, otherwise, revert to a triangle

Performance impact (gprof)

Intel Xeon E5-2690 v4, ifort & MKL 2016.3.210,

	Factor time	Gflop/s	6x4 unrolled	DGEMM kernel	DGEMM copy	Convert panels
unrolled	1138	9.9	1099			
triangles	381	29.5	0	207	53	63
squares	308	36.5	0	210	32	0
MUMPS	341	32.8		217	28	

- Unrolled Fortran doesn't effectively use AVX
- Data movement overhead is significant
  - Even within MKL

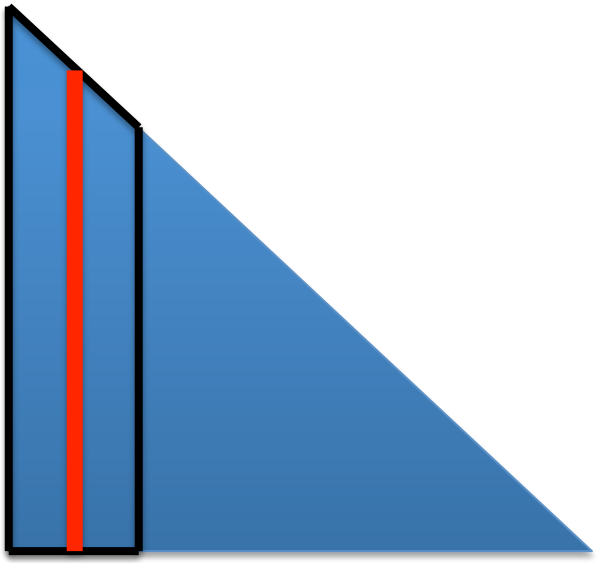
Outline

- Introduction
- Triangles vs. squares
- L vs. U
- Static vs. dynamic scheduling
- Summary

Row major vs. column major?

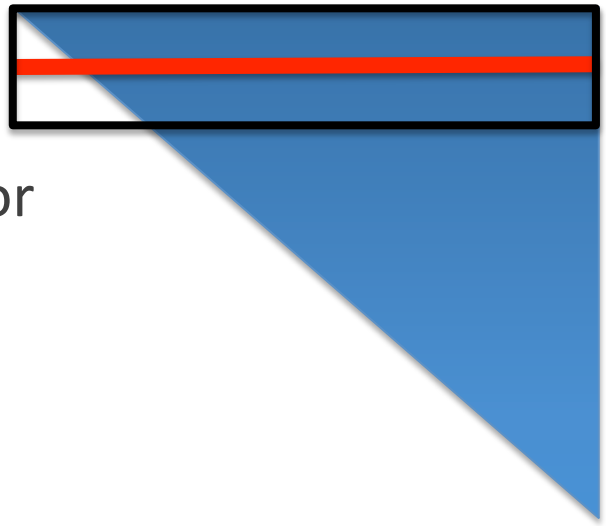
- mf2 thinks its LDL^T

  Stores panels of L, column major
  Stores permuted columns of A
  Sends columns to parents (usually)



- MUMPS thinks its U^TDU

  Stores rectilinear panels of U, row major
  Stores permuted rows of A
  Sends rows to parents



DTRSM

- DTRSM with 128-bit SSE wasn't worthwhile
  - Slightly faster than unrolled Fortran
  - Unrolled Fortran tested for threshold violations as it went
  - Little unnecessary work performed if pivoting
- DTRSM with wider AVX is worthwhile
  - MUMPS could call it in place if it was column major
- mf2 has to extract and transpose columns of L

Impeller profile:	DTRSM	10.1 sec.
	Transpose	6.7 sec.

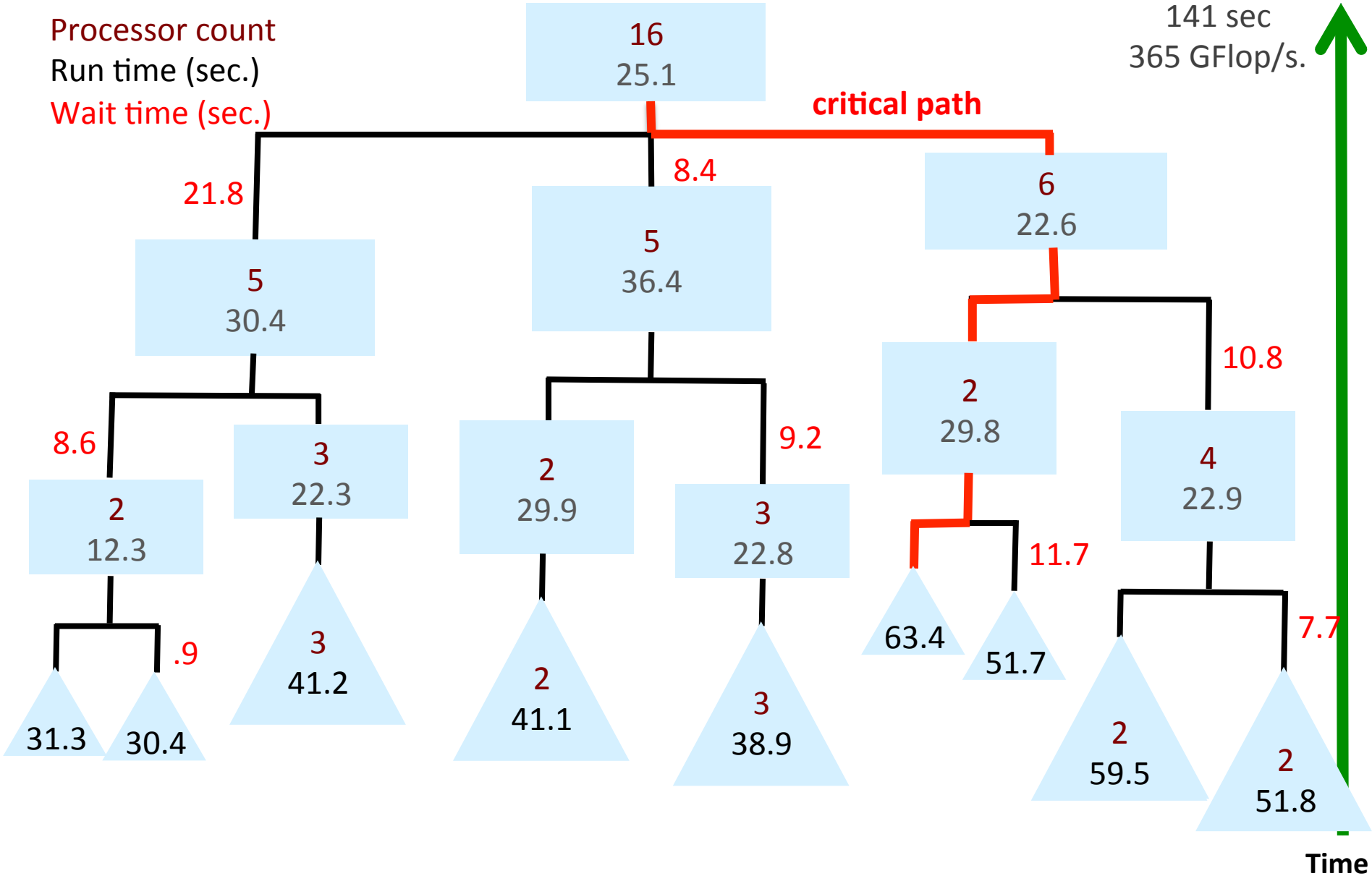
Outline

- Introduction
- Triangles vs. squares
- L vs. U
- Static vs. dynamic scheduling
- Summary

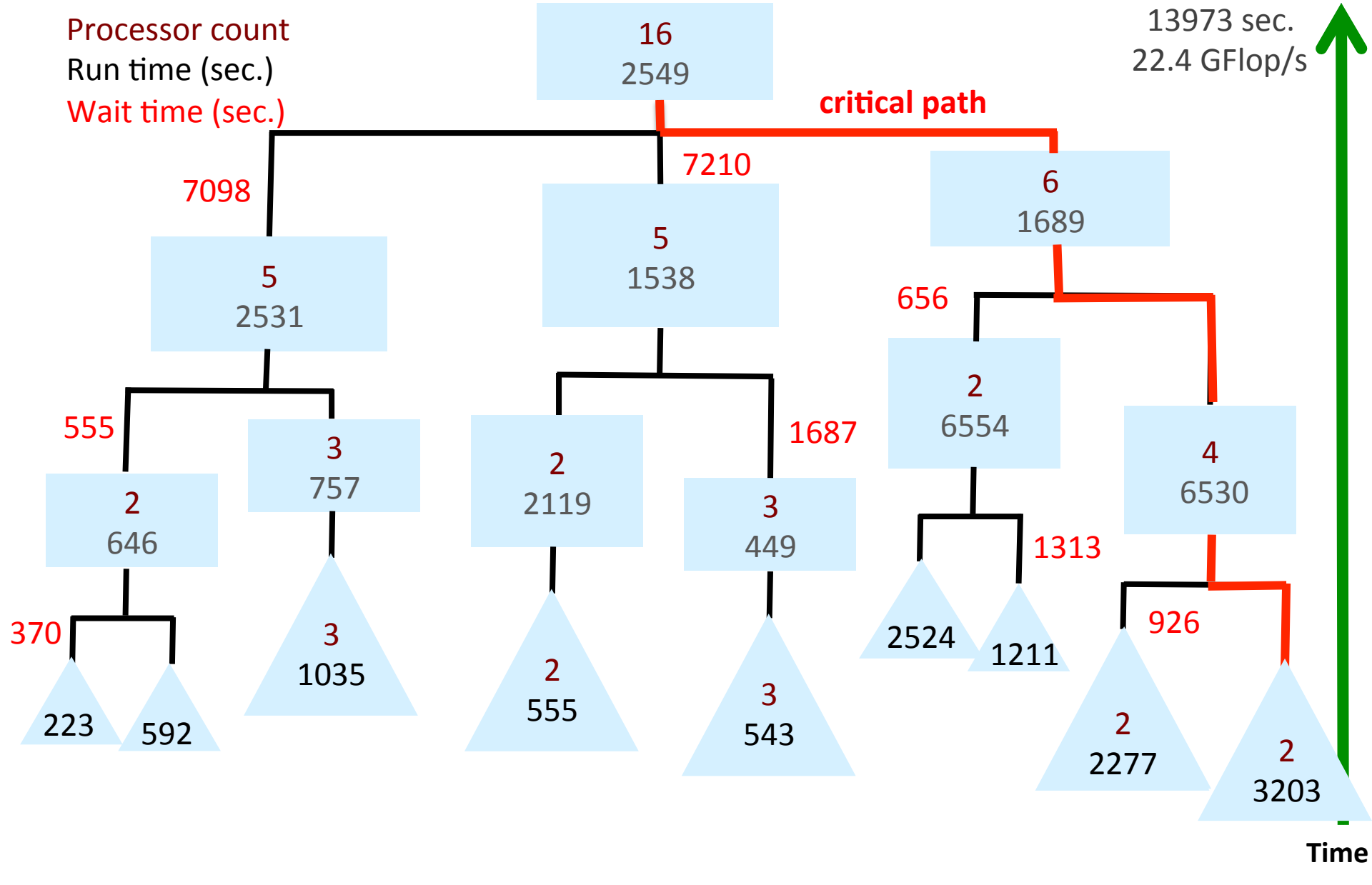
Message passing distribution of work

- mf2 uses a static, subtree-subcube like mapping
  - Processor allocation based on a performance model
  - Each parallel frontal matrix is assigned an MPI communicator
  - MPI collectives used for communication
- MUMPS dynamically reassigns update tasks as needed
  - Static allocation of master processes to fully summed rows
  - Static initial assignment for processing contribution blocks
  - Dynamic reassignment by master, at run time, based on load
  - All communication is point-to-point with MPI_I{SEND,IRECV}

Well behaved: Long_Coup_dt2 (E-6 threshold)



Badly behaved: Long_Coup_dt2 (E-2 threshold)



MUMPS successfully adapts

Code	Pivot threshold	Factor seconds	row/col exchange	2x2 pivots	Deferred equations
mf2	E-2	13973	973136	0	1272687
mf2	E-3	2063	443426	0	416336
mf2	E-6	141	0	0	0
MUMPS	E-2	248	?	2	46

Factoring Long_Coup_dt2 on 16 MPI ranks
1.47M equations, 51.45 Tops w/o deferred eqns.

- MUMPS scales the input matrix, while mf2 doesn't
  - LS-DYNA uses null-space method to eliminate constraints
- MUMPS load balance adapts to deferred equations
  - mf2 does not

Outline

- Introduction
- Triangles vs. squares
- L vs. U
- Static vs. dynamic scheduling
- Summary

Summary

- There is no one optimal multifrontal method
  - Depends on the goals and constraints of the user
  - Depends on properties of the input matrices
  - Depends on the machine its running on
- Codes are optimized for the problems that motivate us
  - LS-DYNA is primarily a multi-physics, finite element code
  - MUMPS has a much wider range of users
- Learning from each other
  - Ongoing effort
  - Good for everybody