

Parallel triangularisation: Distribution of rows

Processor p operates using the following **row-wise** data

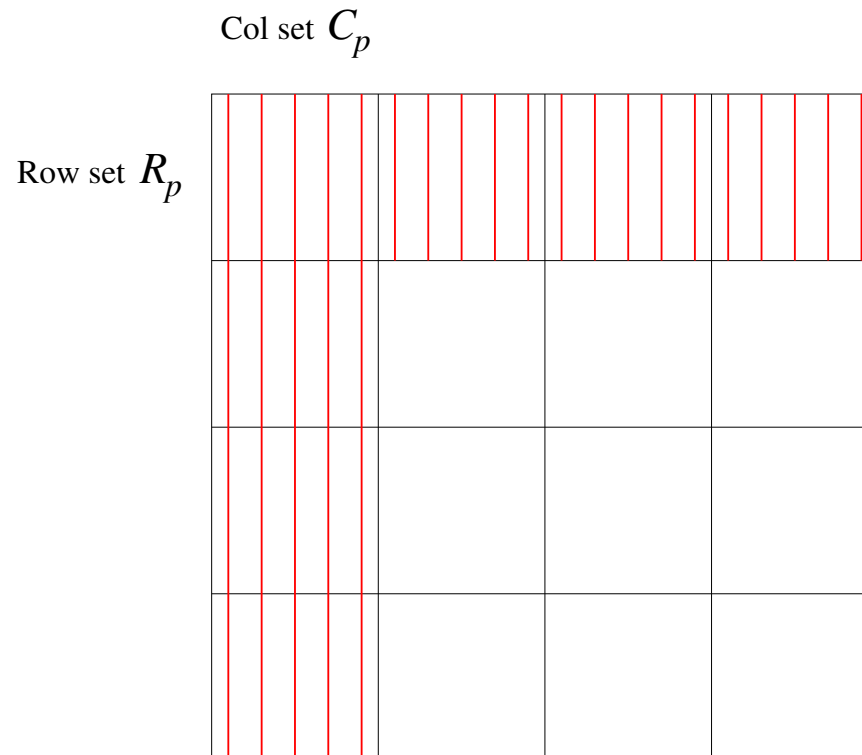
- All entries for rows in set R_p
- Entries in columns C_p for rows in set R'_p

Col set C_p Row set R_p [illegible]

Parallel triangularisation: Distribution of columns

Processor p operates using the following **column-wise** data

- All entries for columns in set C_p
- Entries in rows R_p for columns in set C'_p



Parallel triangularisation: Overview

For each processor $p = 1, \dots, N$:

- **Initialisation:** Determine row and column count data for row set R_p and column set C_p
- **Major iteration:** Repeat:
 - **Minor iteration:** Identify row (column) singletons in R_p (C_p) until it is “wise” to stop
 - Broadcast pivot indices to all other processors
 - Update row (column) count data using pivot indices from all other processors
- Until there are no row or column singletons

Communication cost of $O(m \log N)$ for computation cost $O(\tau)$



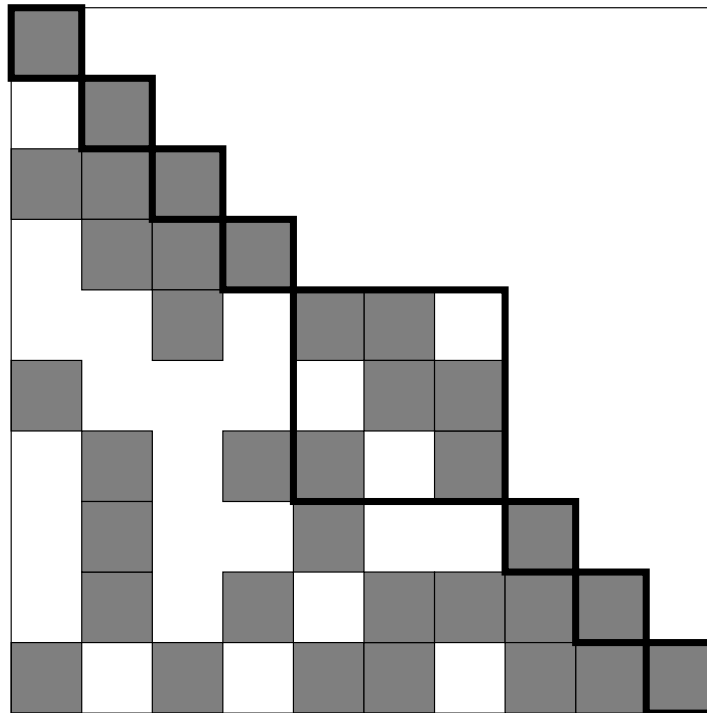
Parallel triangularisation: Minor iteration

Within each minor iteration

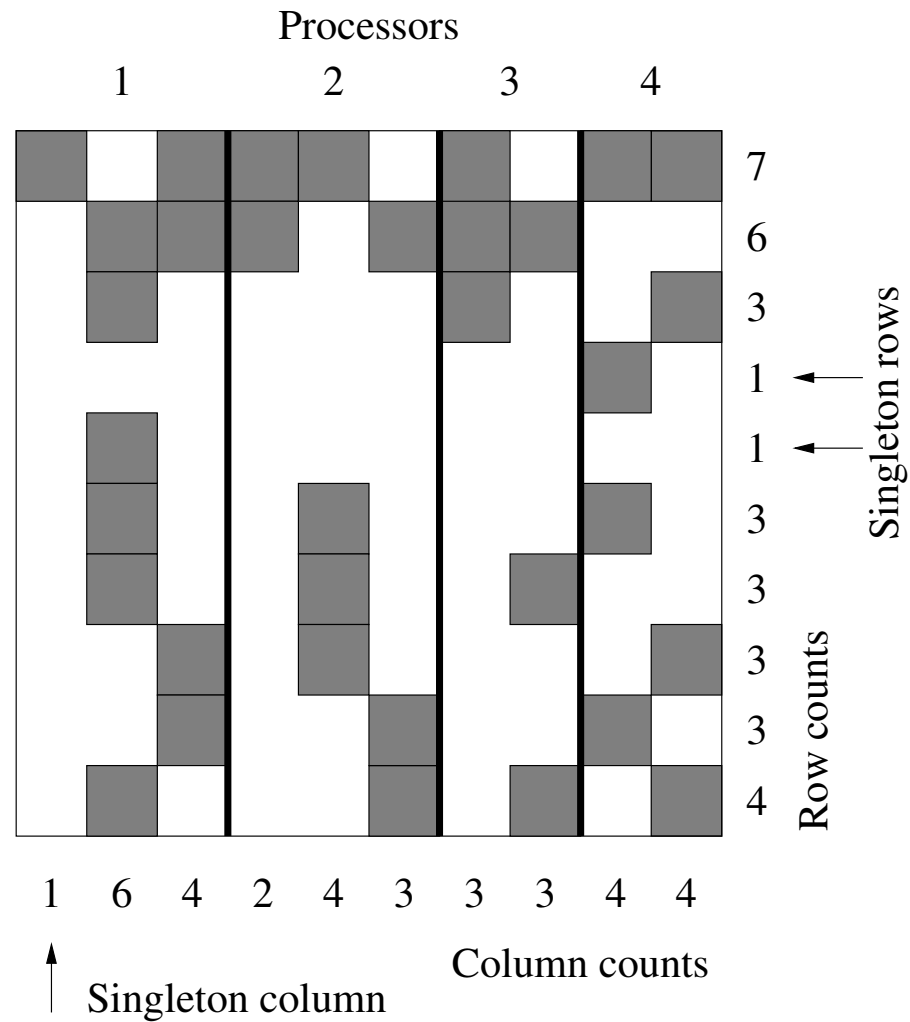
- Row and column counts are
 - initially global;
 - updated according to pivots determined locally;
 - become upper bounds on global counts
- When is it wise to stop performing minor iterations?
 - Too soon: communication overheads dominate
 - Too late: load imbalance
- Aim to find a particular proportion of the pivots in each major iteration



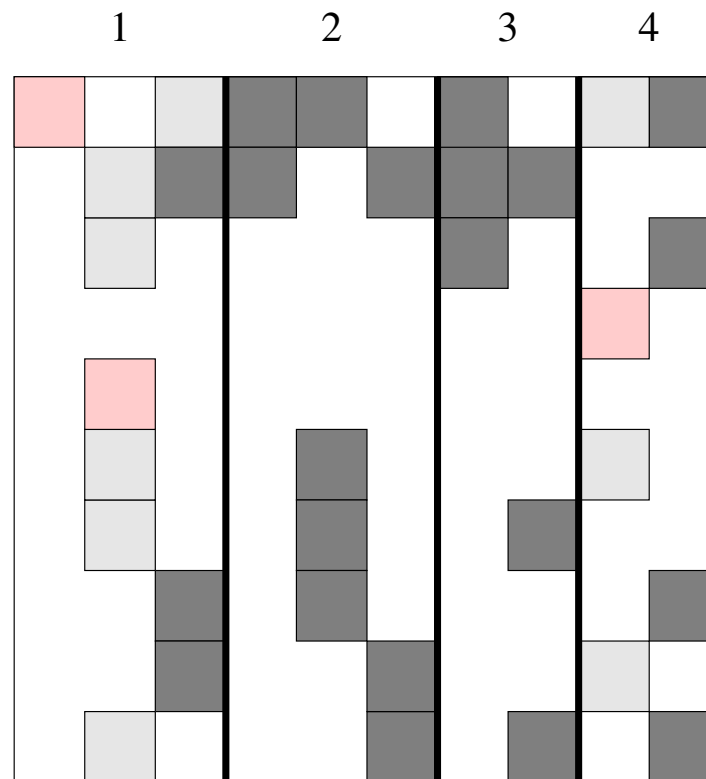
Parallel triangularisation: Example



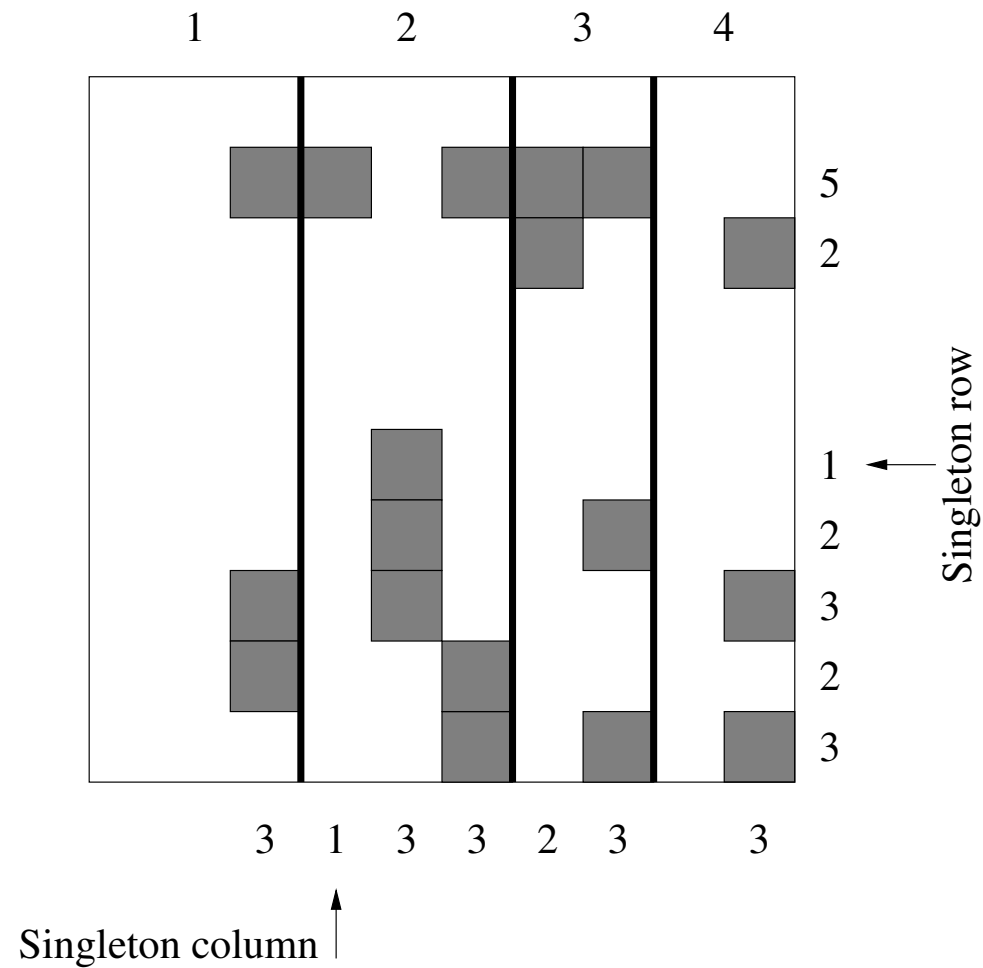
Parallel triangularisation: Iteration 1



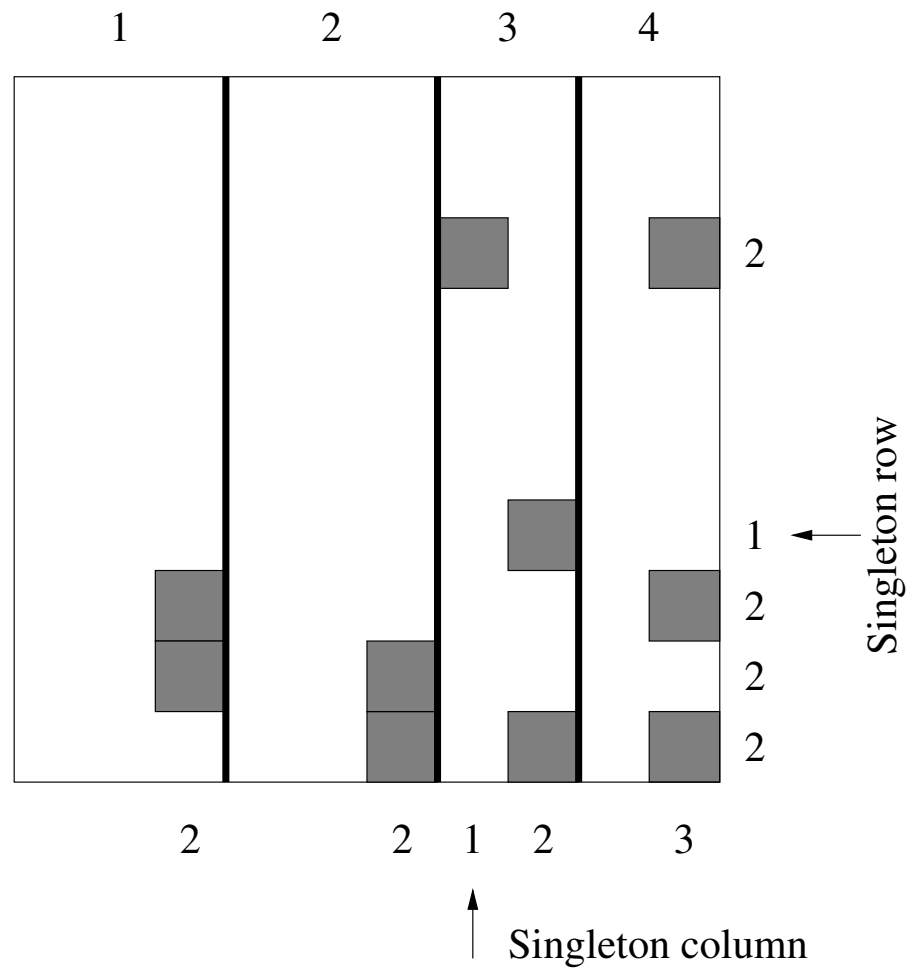
Parallel triangularisation: Iteration 1 result



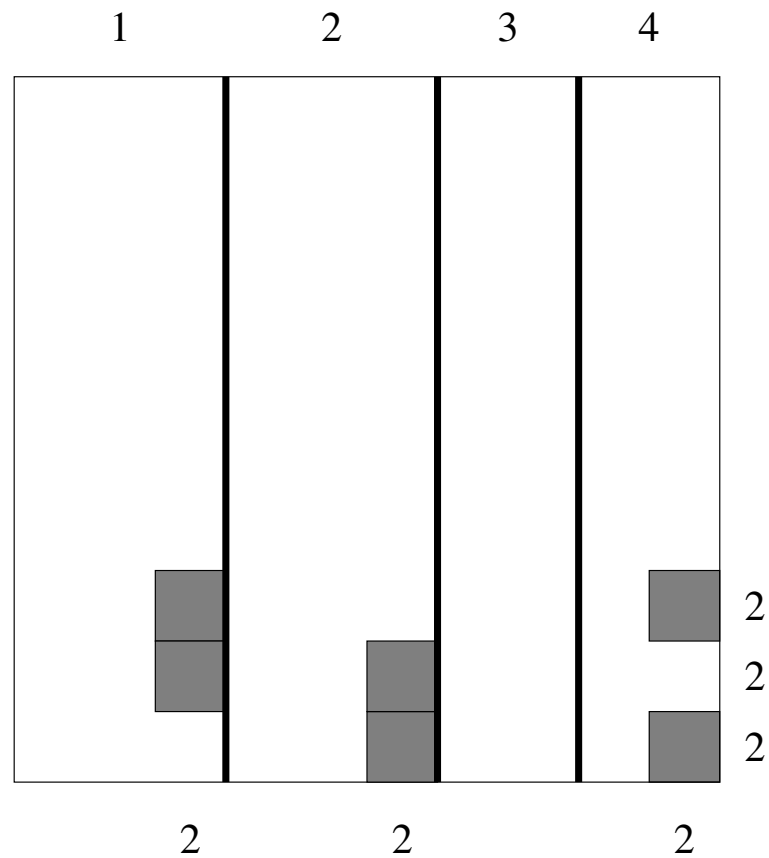
Parallel triangularisation: Iteration 2



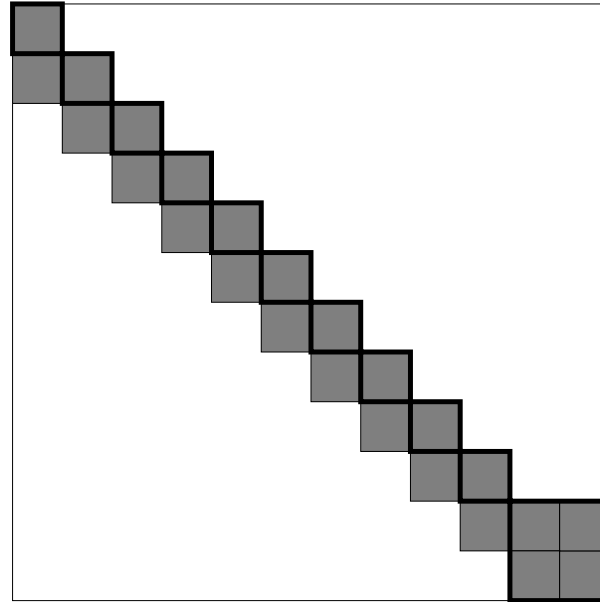
Parallel triangularisation: Iteration 3



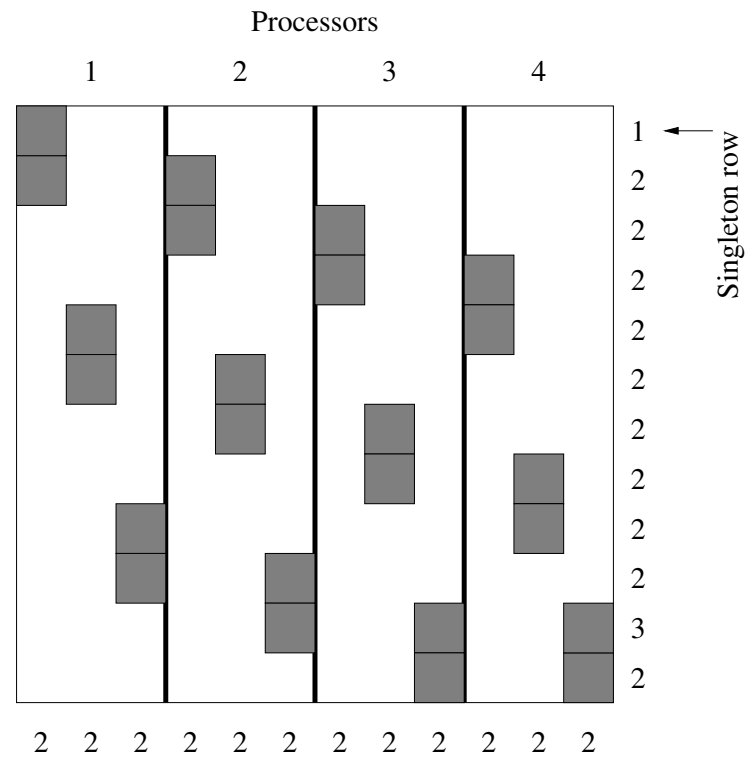
Parallel triangularisation: Iteration 4



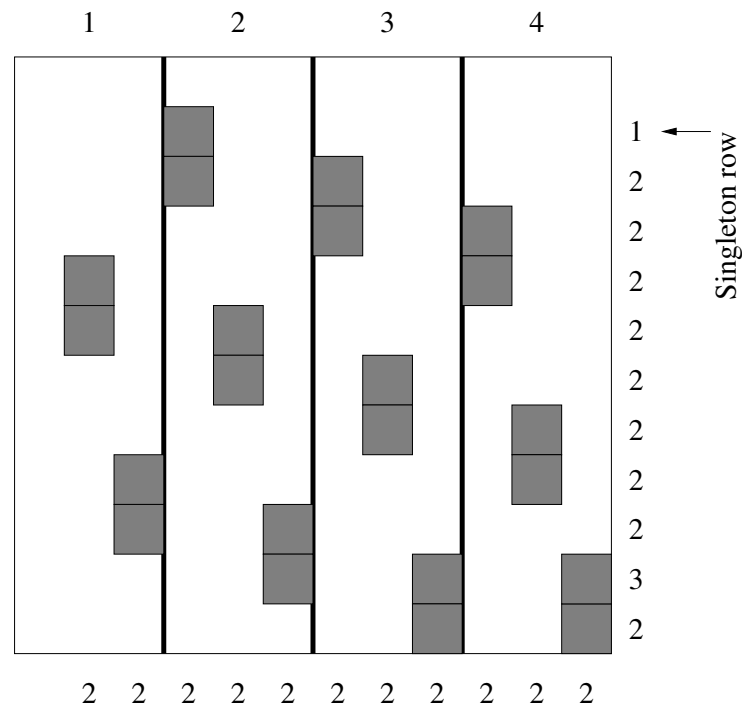
Parallel triangularisation: Worst case behaviour



Worst case behaviour: Iteration 1



Worst case behaviour: Iteration 2



- Only one pivot is identified on one processor until pivot is broadcast
- Reduces to serial case with considerable overhead

Measures of performance

Assess viability of parallel scheme using serial simulator

- **Load balance:** Radically different numbers of pivots on processors
Processor idleness
- **Communication overhead:** Excess numbers of major iterations
Communication costs dominate
- Relate performance to **ideal number of major iterations**

$$\frac{100}{\text{Target \% of triangular pivots per major iteration}}$$

Parallel triangularisation: Good performance

- For model nsct2
- 23003 rows
- 16329 logicals
- Bump dimension is 183
- 4 processors
- 10% of pivots per major iteration
- Ideal number of major iterations is 10

It	Pivots found on processor			
	1	2	3	4
1	163	163	163	163
2	163	163	163	163
3	163	163	163	163
4	163	163	163	163
5	163	163	163	163
6	163	163	163	163
7	163	163	163	163
8	163	163	163	163
9	163	163	163	163
10	162	148	156	155
11	0	1	1	0
12	0	0	0	0



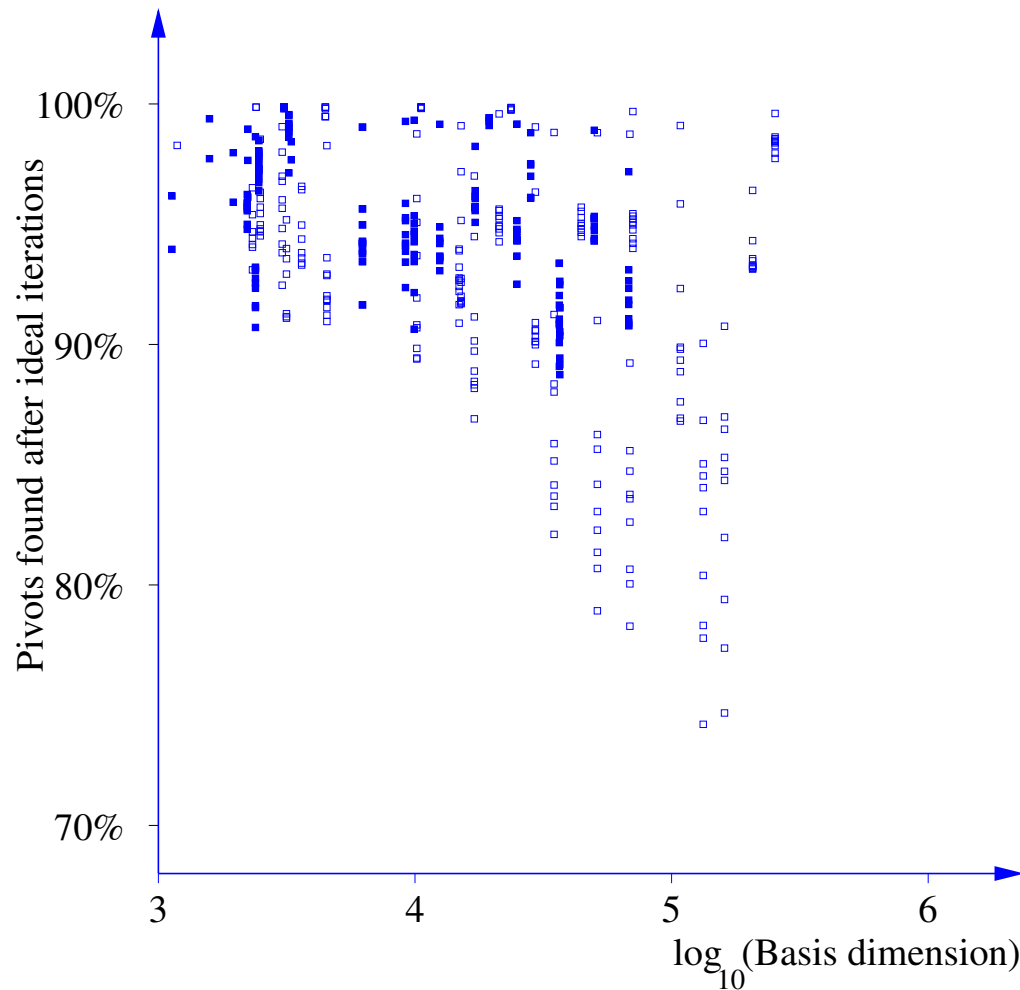
Serial simulation of parallel triangularisation: “Poor” performance

- For model pds-06
- 9881 rows
- 952 logicals
- Bump dimension is 55
- 4 processors
- 10% of pivots per major iteration

It	Pivots found on processor				Pivots
	1	2	3	4	
1	222	222	222	222	10%
⋮	⋮	⋮	⋮	⋮	
8	222	222	222	222	80%
9	162	186	169	195	88%
10	105	78	87	84	92%
11	41	52	47	38	94%
15	11	10	10	4	97%
⋮	⋮	⋮	⋮	⋮	
20	3	3	5	4	98%
⋮	⋮	⋮	⋮	⋮	
40	2	1	0	0	99%
⋮	⋮	⋮	⋮	⋮	
57	0	0	0	0	100%

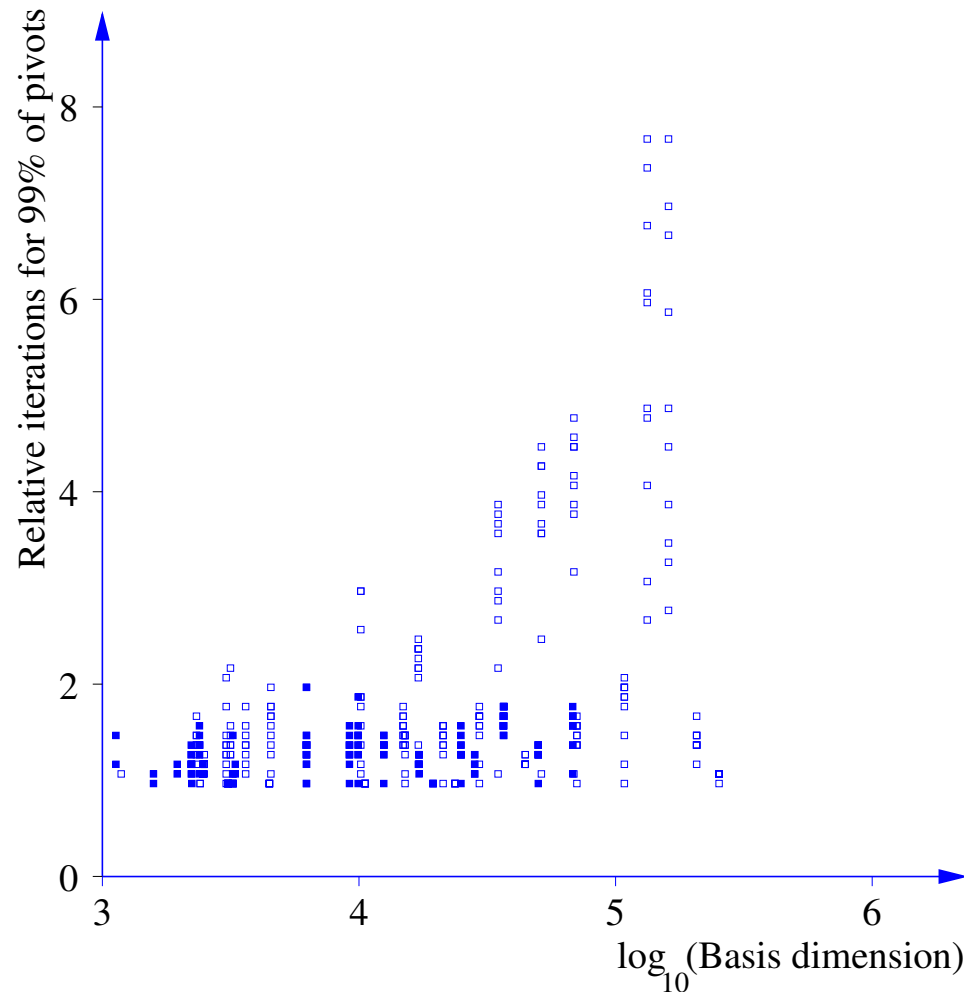


Parallel triangularisation: Percentage of pivots after ideal iterations



- Typically get 90%
- Generally get at least 80%
- Indicates good load balance

Parallel triangularisation: Relative number of iterations for 99% of pivots



- Typically get 99% of pivots within a small multiple of the ideal number of iterations
- Occasionally requires large multiple of the ideal number of iterations
- **Recall:**
 - Additional iterations may be very much faster than “ideal” iterations
 - Communication overhead will dominate if too few pivots are found in an iteration

Prototype parallel implementation: speedup

- For model pds-100
- 156243 rows
- 7485 logicals
- Bump dimension is 1655
- 10% of pivots per major iteration

	Processors				
Pivots	1	2	4	8	16
50%	0.5	1.5	2.8	3.7	4.6
90%	0.8	1.7	3.4	4.3	5.0
99%	0.8	1.6	3.0	3.6	3.4
100%	0.8	1.5	2.6	3.2	2.5

Speed-up relative to serial triangularisation

	Processors				
Pivots	1	2	4	8	16
50%	0.5	0.7	0.5	0.3	0.2
90%	0.9	0.8	0.5	0.3	0.2
99%	1.0	0.7	0.5	0.3	0.2
100%	0.9	0.7	0.5	0.3	0.2

Speed of parallel simulator relative to serial triangularisation

	Processors				
Pivots	1	2	4	8	16
50%	0.9	2.2	5.6	11.4	23.9
90%	0.9	2.1	5.4	13.0	25.8
99%	0.9	2.2	5.7	10.9	19.4
100%	0.9	2.1	5.2	10.1	14.5

Speed-up of parallel triangularisation relative to parallel simulator



Conclusions

- Matrix triangularisation identified as dominant serial cost for hyper-sparse LPs
- Significant scope for parallelisation with scheme presented
 - Communication cost of $O(m)$ for computation cost $O(\tau)$
 - Limited cost of switching to serial in case of poor ultimate parallel performance
- Prototype parallel implementation gives fair speed-up over serial triangularisation
- Scope for greater serial and parallel efficiency of implementation

See

- **Slides:** <http://www.maths.ed.ac.uk/hall/Talks>

Thank you

