SYNPLEX
A task-parallel scheme for the revised simplex method
(Part 2)

Julian Hall

School of Mathematics

University of Edinburgh

June 23rd 2005

SYNPLEX

- Synchronous variant of PARSMI

SYNPLEX

- Synchronous variant of PARSMI
- INVERT not overlapped with basis changes $\Rightarrow$ numerical stability

SYNPLEX

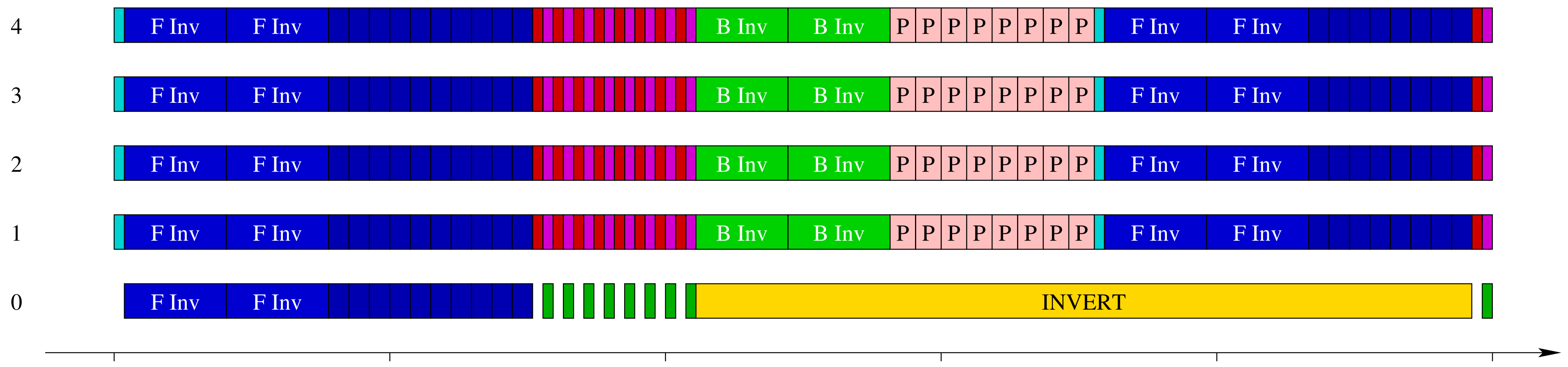
- Synchronous variant of PARSMI
- INVERT not overlapped with basis changes $\Rightarrow$ numerical stability
- CHUZC uses up-to-date reduced costs $\Rightarrow$ better candidate persistence

SYNPLEX

- Synchronous variant of PARSMI
- INVERT not overlapped with basis changes $\Rightarrow$ numerical stability
- CHUZC uses up-to-date reduced costs $\Rightarrow$ better candidate persistence
- Target platform: shared memory Sun Fire E15k (OpenMP)

SYNPLEX

- Synchronous variant of PARSMI
- INVERT not overlapped with basis changes $\Rightarrow$ numerical stability
- CHUZC uses up-to-date reduced costs $\Rightarrow$ better candidate persistence
- Target platform: shared memory Sun Fire E15k (OpenMP)



Data parallelism

When using $1 + p$ processors

- Rows distributed over p processors

Data parallelism

When using $1 + p$ processors

- Rows distributed over p processors for data parallel
  - FTRAN for UPDATE etas
  - CHUZR
  - UPDATE tableau in minor iterations
  - UPDATE RHS

Data parallelism

When using $1 + p$ processors

- Rows distributed over p processors for data parallel
  - FTRAN for UPDATE etas
  - CHUZR
  - UPDATE tableau in minor iterations
  - UPDATE RHS
- Columns distributed over p processors

Data parallelism

When using $1 + p$ processors

- Rows distributed over p processors for data parallel
  - FTRAN for UPDATE etas
  - CHUZR
  - UPDATE tableau in minor iterations
  - UPDATE RHS
- Columns distributed over p processors for data parallel
  - PRICE
  - CHUZC

Data location challenge

- B_0 factored serially on one processor

Data location challenge

- B_0 factored serially on one processor
- Factors used serially on all processors to solve linear systems

Data location challenge

- B_0 factored serially on one processor
- Factors used serially on all processors to solve linear systems
- Each solution used for data parallel operations over all processors

Practical implementation

- Use a task manager processor

Practical implementation

- Use a task manager processor
  - Task parallel operations allocated according to processor activity
  - Enables different computational components to overlap

Practical implementation

- Use a task manager processor
  - Task parallel operations allocated according to processor activity
  - Enables different computational components to overlap
- Prevent different processors from writing to consecutive components

Practical implementation

- Use a task manager processor
  - Task parallel operations allocated according to processor activity
  - Enables different computational components to overlap
- Prevent different processors from writing to consecutive components
  - Insert “padding” between row partitions: implemented

Practical implementation

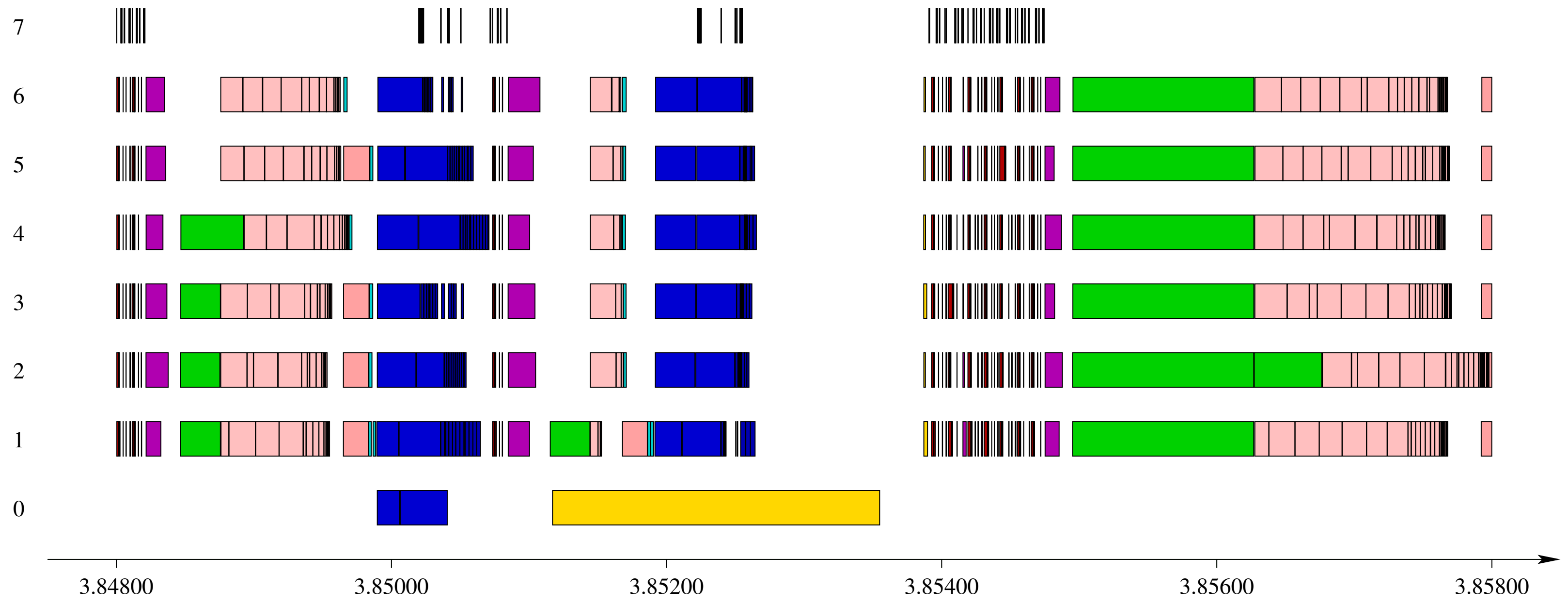
- Use a task manager processor
  - Task parallel operations allocated according to processor activity
  - Enables different computational components to overlap
- Prevent different processors from writing to consecutive components
  - Insert “padding” between row partitions: implemented
  - Insert “padding” between column partitions: not yet implemented

Results

			1 Processor	Speed-up	
Model	Rows	Columns	CPU (s)	4 processors	8 processors
cre-a	3517	4067	5.76	1.16	1.83
25fv47	822	1571	8.78	1.54	1.99
greenbea	2393	5405	29.22	-	2.30
ken-11	14695	21349	41.26	1.40	2.52
stocfor3	16676	15695	98.44	1.50	2.76
pds-06	9882	28655	138.84	1.58	3.05

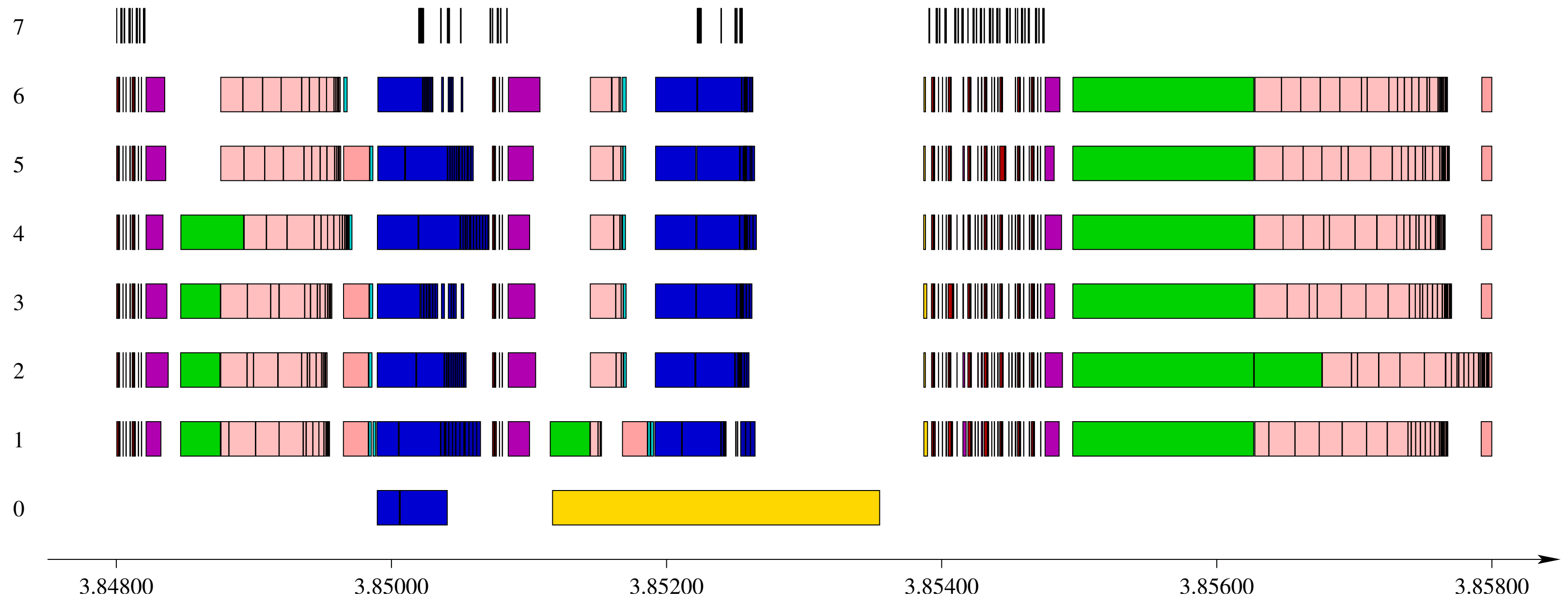
Poor performance: cre-a

Least speed-up (1.83) on 8 processors



Poor performance: cre-a

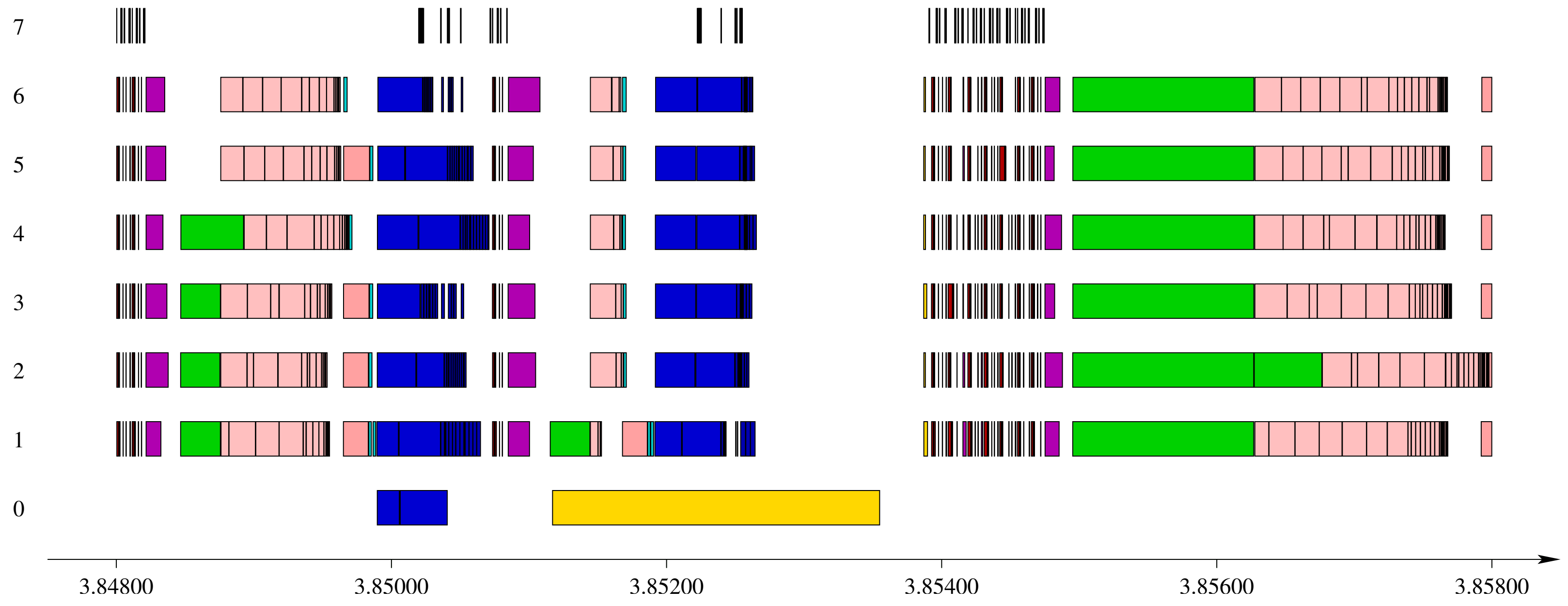
Least speed-up (1.83) on 8 processors



Operation	Slow-down in total time	Overall speed-up
Inv-FTRAN	3.29	-

Poor performance: cre-a

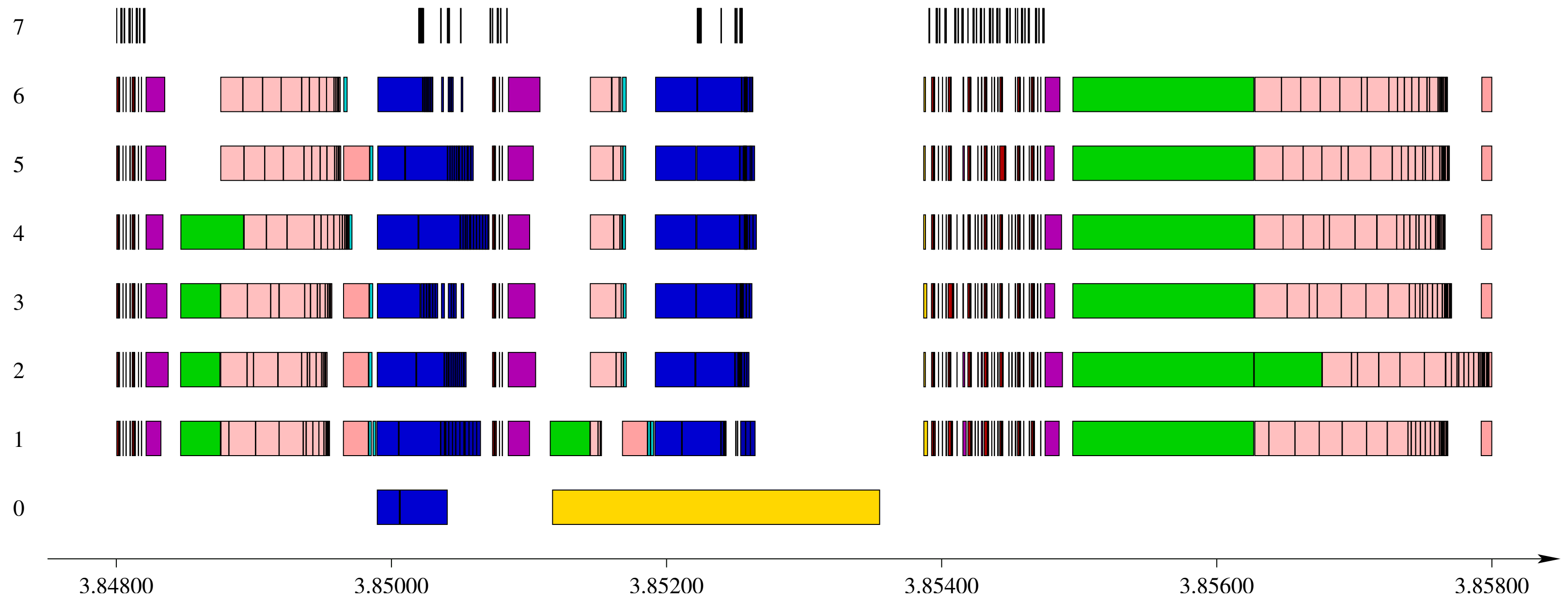
Least speed-up (1.83) on 8 processors



Operation	Slow-down in total time	Overall speed-up
Inv-FTRAN	3.29	-
Inv-BTRAN	2.11	-

Poor performance: cre-a

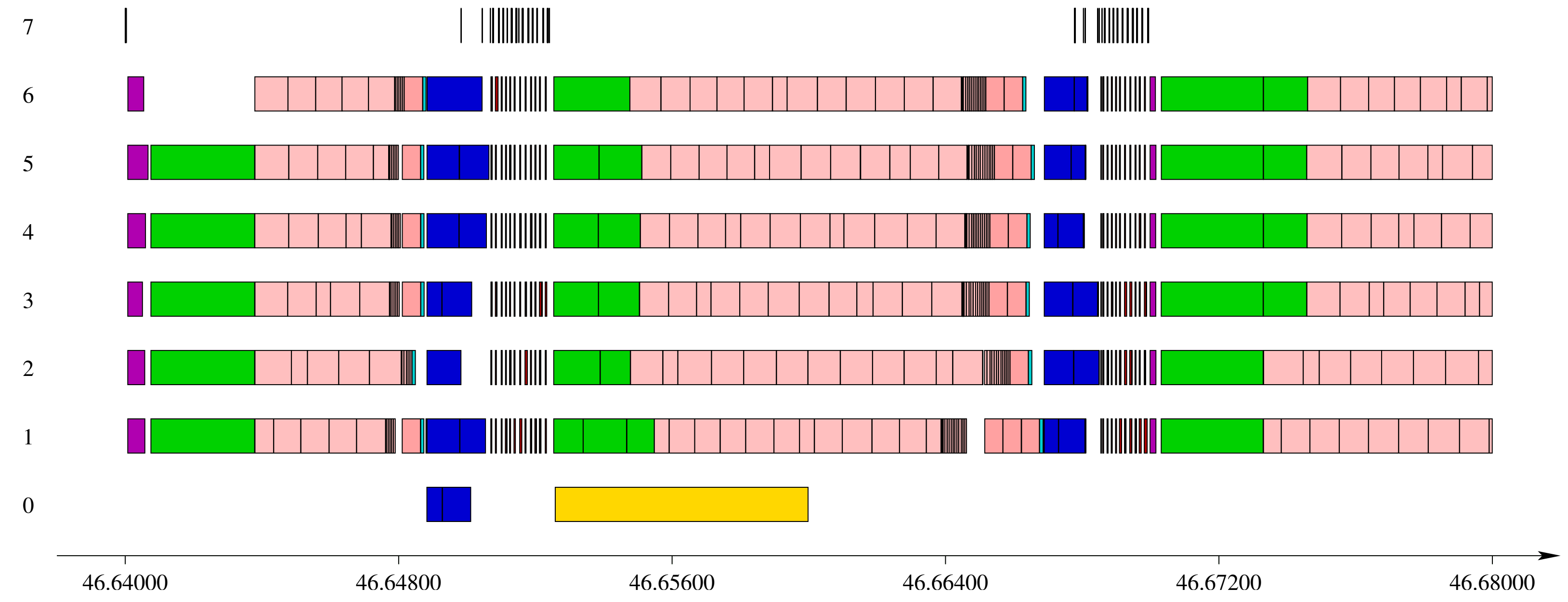
Least speed-up (1.83) on 8 processors



Operation	Slow-down in total time	Overall speed-up
Inv-FTRAN	3.29	-
Inv-BTRAN	2.11	-
PRICE	2.94	2.04

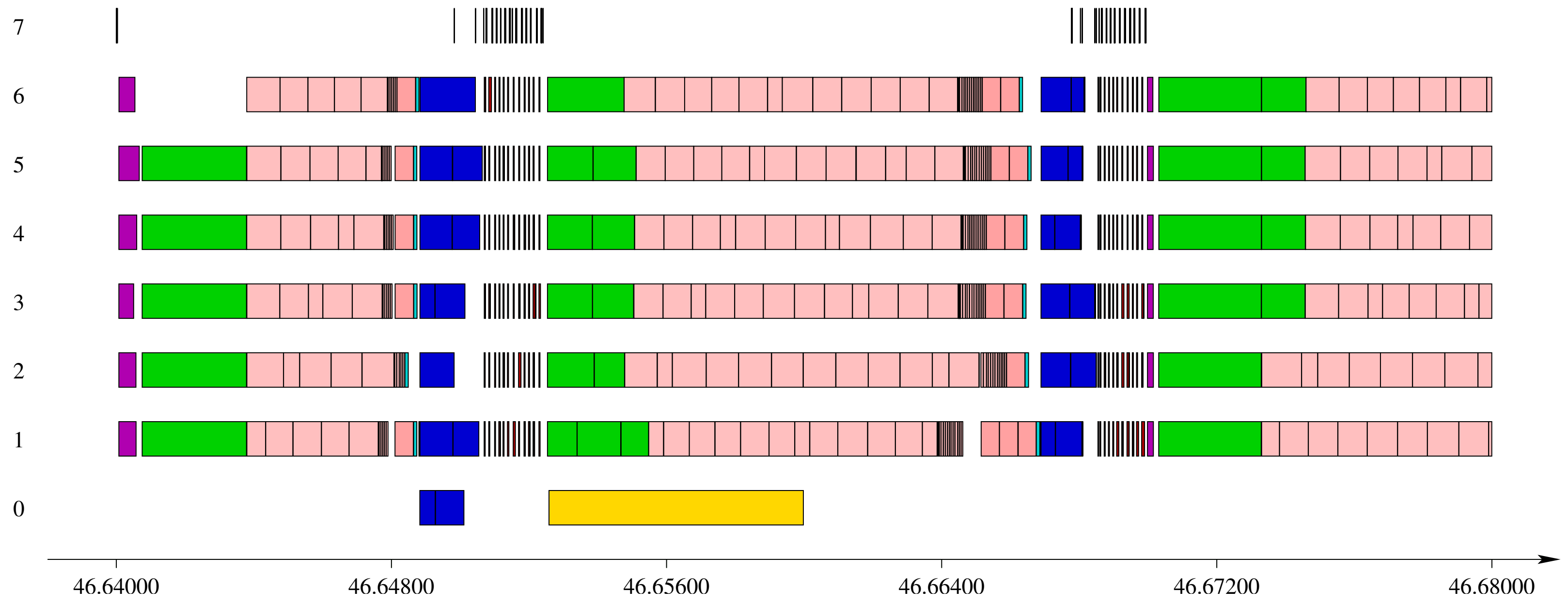
Fair performance: pds-06

Best speed-up (3.05) on 8 processors



Fair performance: pds-06

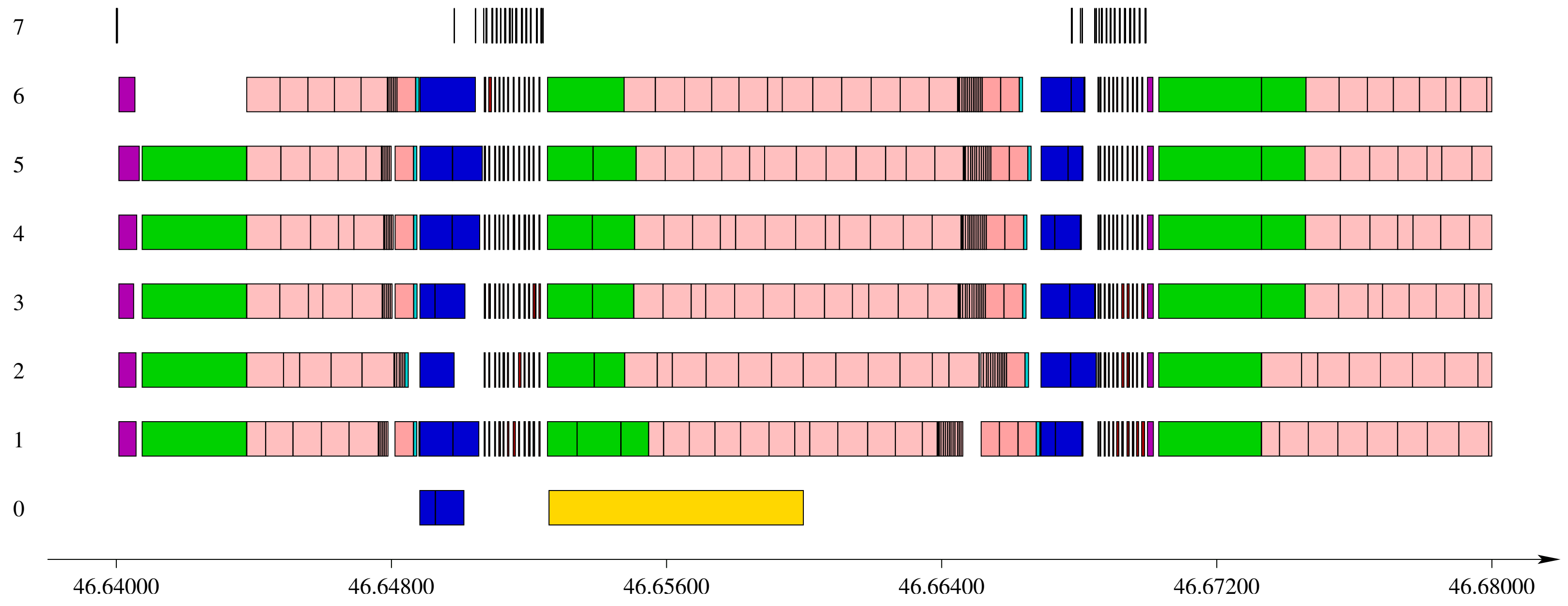
Best speed-up (3.05) on 8 processors



Operation	Slow-down in total time	Overall speed-up
Inv-FTRAN	2.02	-

Fair performance: pds-06

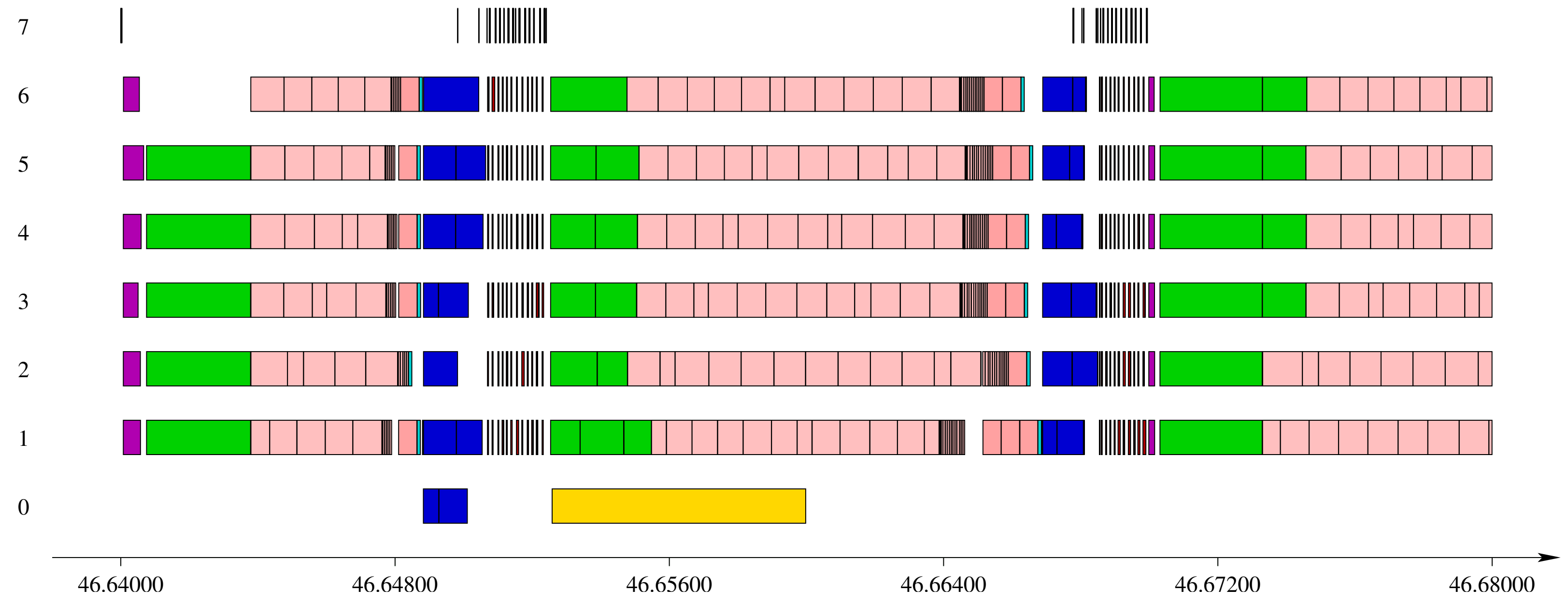
Best speed-up (3.05) on 8 processors



Operation	Slow-down in total time	Overall speed-up
Inv-FTRAN	2.02	-
Inv-BTRAN	1.79	-

Fair performance: pds-06

Best speed-up (3.05) on 8 processors



Operation	Slow-down in total time	Overall speed-up
Inv-FTRAN	2.02	-
Inv-BTRAN	1.79	-
PRICE	1.69	3.55

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
  - Wasted FTRANs
  - Increased number of iterations

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
  - Wasted FTRANs
  - Increased number of iterations
- Inevitable(?) data management limitations
  - Some operations are significantly slower in parallel than in serial

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
  - Wasted FTRANs
  - Increased number of iterations
- Inevitable(?) data management limitations
  - Some operations are significantly slower in parallel than in serial
- Serial INVERT limits scalability

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
  - Wasted FTRANs
  - Increased number of iterations
- Inevitable(?) data management limitations
  - Some operations are significantly slower in parallel than in serial
- Serial INVERT limits scalability

SYNPLEX refinements:

- Parallel INVERT
  - Should allow larger problems to be solved than serial revised simplex solvers
  - Impressive results from parallel direct methods for linear systems give hope

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
  - Wasted FTRANs
  - Increased number of iterations
- Inevitable(?) data management limitations
  - Some operations are significantly slower in parallel than in serial
- Serial INVERT limits scalability

SYNPLEX refinements:

- Parallel INVERT
  - Should allow larger problems to be solved than serial revised simplex solvers
  - Impressive results from parallel direct methods for linear systems give hope

Future prospects:

- Pure data parallel revised simplex *without* multiple pricing

Bibliography

Paper:	`http://www.maths.ed.ac.uk/hall/ParSimplex`
This talk:	`http://www.maths.ed.ac.uk/hall/CSC05`

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
    - Wasted FTRANs
    - Increased number of iterations

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
  - Wasted FTRANs
  - Increased number of iterations
- Inevitable(?) data management limitations
  - Some operations are significantly slower in parallel than in serial

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
  - Wasted FTRANs
  - Increased number of iterations
- Inevitable(?) data management limitations
  - Some operations are significantly slower in parallel than in serial
- Serial INVERT limits scalability

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
  - Wasted FTRANs
  - Increased number of iterations
- Inevitable(?) data management limitations
  - Some operations are significantly slower in parallel than in serial
- Serial INVERT limits scalability

SYNPLEX refinements:

- Parallel INVERT
  - Should allow larger problems to be solved than serial revised simplex solvers
  - Impressive results from parallel direct methods for linear systems give hope

Conclusions

SYNPLEX limitations:

- Algorithmic limitations of revised simplex with multiple pricing
  - Wasted FTRANs
  - Increased number of iterations
- Inevitable(?) data management limitations
  - Some operations are significantly slower in parallel than in serial
- Serial INVERT limits scalability

SYNPLEX refinements:

- Parallel INVERT
  - Should allow larger problems to be solved than serial revised simplex solvers
  - Impressive results from parallel direct methods for linear systems give hope

Future prospects:

- Pure data parallel revised simplex *without* multiple pricing

Bibliography

Paper:	`http://www.maths.ed.ac.uk/hall/ParSimplex`
This talk:	`http://www.maths.ed.ac.uk/hall/CSC05`