Dissecting Transactional Executions in Haskell

Cristian Perfumo[+*], Nehir Sonmez[+*], Adrian Cristal[+], Osman S. Unsal[+], Mateo Valero[+*], Tim Harris[#]

[+]Barcelona Supercomputing Center

[*]Computer Architecture Department, UPC, Barcelona, Spain

[#] Microsoft Research Cambridge

Microsoft

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Motivation

- Haskell is a great tool to try out ideas on transactional memory.
- Need more detail than just execution time.
  - Big rollback rate?
  - Time in the commit phase?
  - Overhead of the transactional runtime?
  - Relationship between number of reads and readset? Writes? Transactional read-to-write ratio?
  - Trend with more processors?
- Dearth of transactional benchmarks for Haskell.

Contributions

- A Haskell STM application suite that can be used as a benchmark by the research community.
- Addition of detailed transactional data gathering module in Haskell STM.
- Based on the collected raw data, new metrics are derived.
- These metrics can be used to characterize STM applications.

Background in Haskell STM

- Pure and lazy functional programming language.
- Write-buffer and lazy conflict detection.
- Object-based conflict detection.
- The IO world and the STM world are separated thanks to monads.
  - Tvars can't be accessed non-transactionally

Running STM Operations	Transactional Variable Operations
`atomically::STM a->IO a`	`data TVar a`
`retry::STM a`	`newTVar::a->STM(TVar a)`
`orElse::STM a->STM a->STM a`	`readTVar::TVar a->STM a`
	`writeTVar::TVar a->a->STM()`

Applications in the suite

- Some are developed by us and some by developers that don't know about the internals of the (underlying) STM implementation .
- Different lengths.
- Different number of atomic blocks.

Application	# lines	# atomic
BlockWorld	1150	13
GCD	1056	13
Prime	1071	13
Sudoku	1253	13
UnionFind	1157	13
TCache	315	6
SingleInt	82	1
LL	250	7
LLUnr	250	7

Gathered statistics

- For committed and aborted transactions:
  - Number of transactions.
  - Work time.
  - Commit phase time.
  - Number of transactional reads and writes.
  - Readset and writeset lengths (in objects).
- Histogram of rollbacks

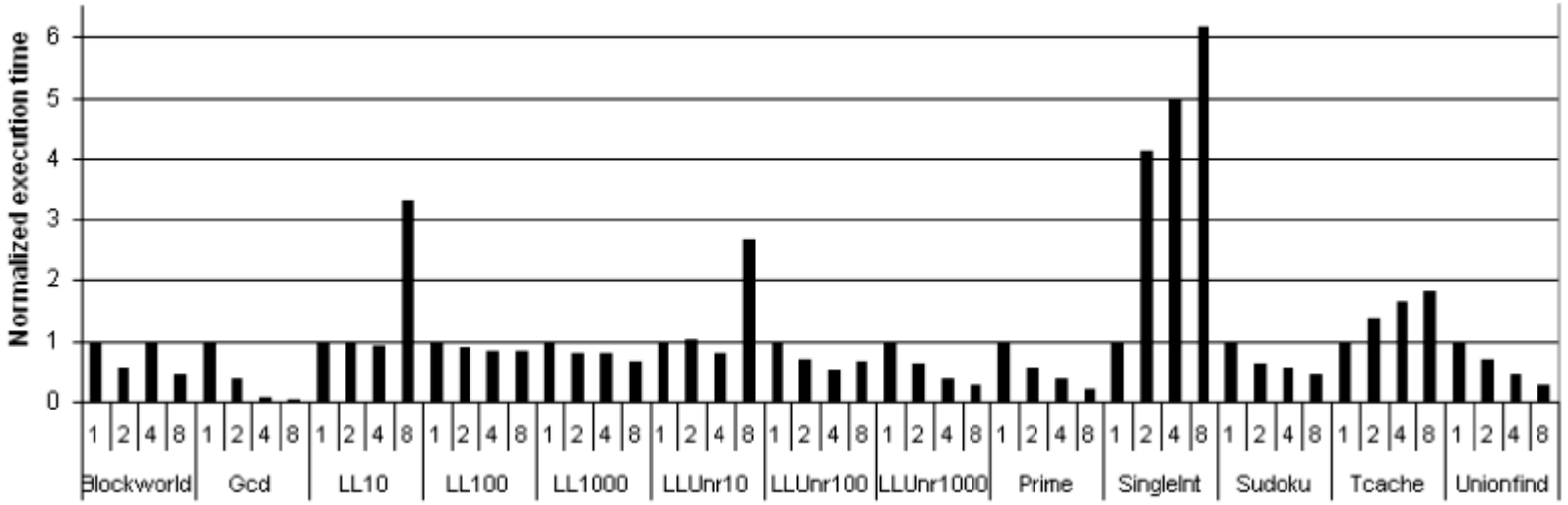
Execution time

- 8 cores (four dual-core SMP) Intel Xeon 5000 3.0 GHz processors.
- 4MB L2 cache/processor.
- 16GB of total memory.
- Exactly as many threads as physical cores.
- All of the reported results are based on the average of five executions.

Application	1 core	2 cores	4 cores	8 cores
Blockworld	13.43	7.30	13.24	5.91
Gcd	76.83	28.78	5.40	2.35
LL10	0.08	0.08	0.07	0.26
LL100	0.31	0.28	0.25	0.25
LL1000	4.84	3.77	3.72	3.19
LLUnr10	0.08	0.08	0.06	0.21
LLUnr100	0.36	0.24	0.18	0.24
LLUnr1000	3.10	1.92	1.16	0.83
Prime	38.41	21.19	14.14	8.44
SingleInt	0.12	0.52	0.62	0.77
Sudoku	0.72	0.44	0.40	0.32
TCache	2.58	3.52	4.20	4.70
UnionFind	2.64	1.78	1.17	0.74

Execution time (cont.)

- Normalized to one-core configuration execution times.
- They allow us to see scalability.

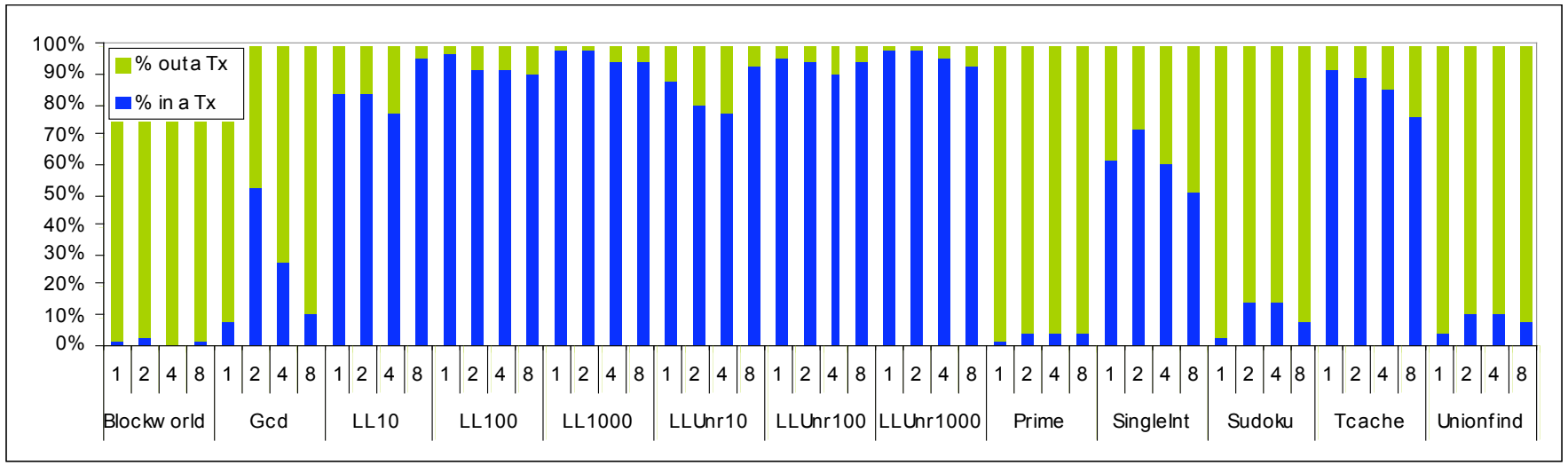


Microsoft

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Inside and outside a transaction

- The more the time inside a transaction, the more the gain in performance by optimizing STM runtime. (Amdahl's Law)

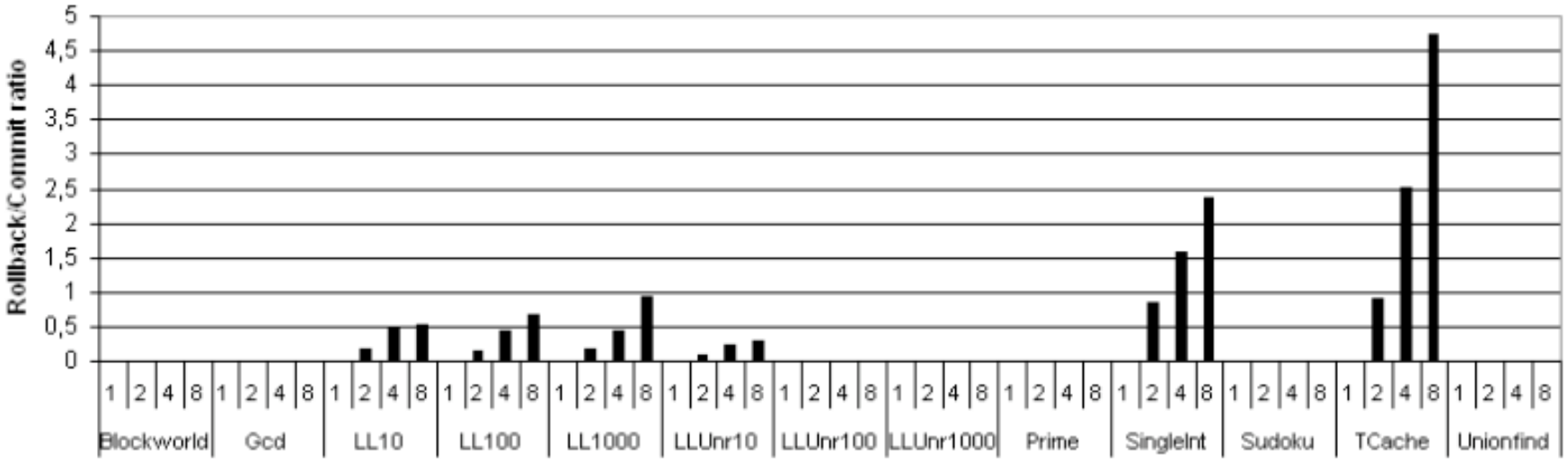


Microsoft

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Stats: Rollback rate

- Allows classifying applications in different groups.
- Accordingly to the group they belong to, the STM runtime can implement different optimizations.

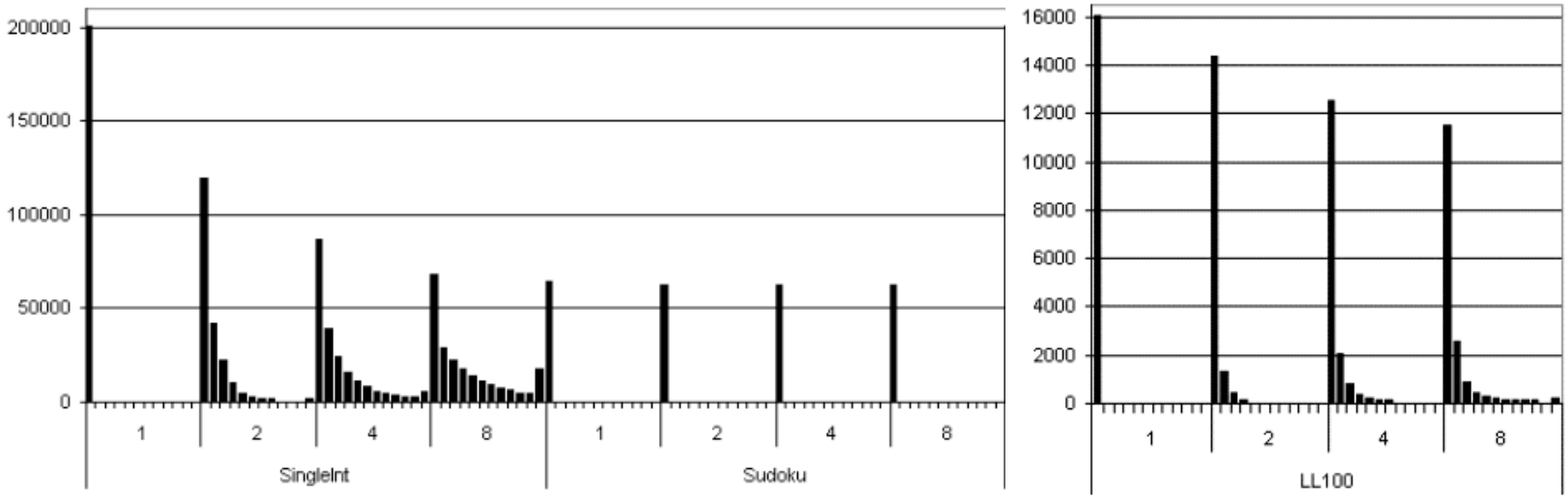


Microsoft

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Stats: Rollback histograms

- Observation: a transaction can be rolled back several (10+) times.
- Therefore: STM can incorporate mechanisms to ensure fairness



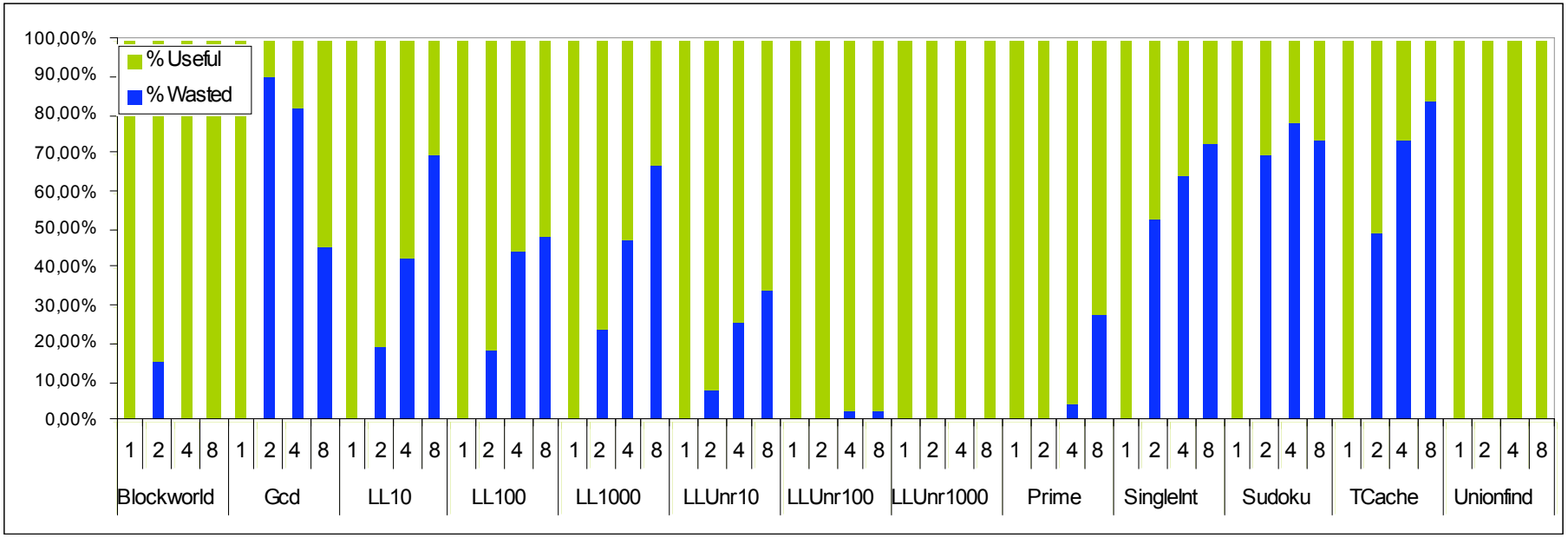
Microsoft

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Stats: Wasted work

- Wasted work:

$$\frac{T(aborted)}{T(aborted)+T(committed)}$$



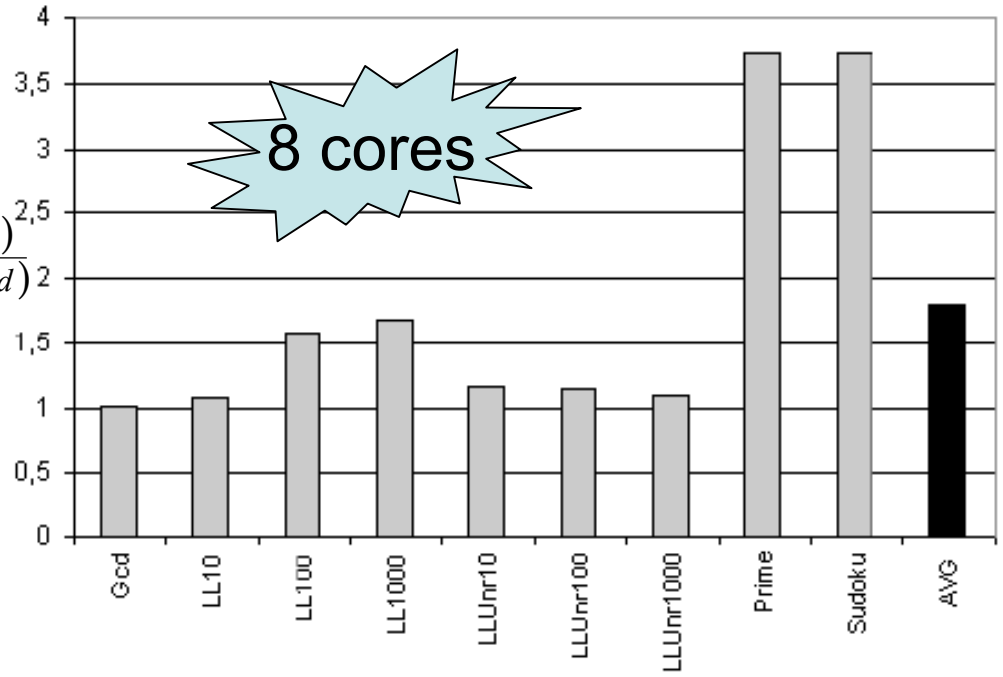
Microsoft

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Stats: Readset size and aborts

- Some apps have transactions with various readset sizes.
- The bigger the readset, the bigger the probability of rollbacks (Intuition confirmed!)

$$\frac{AVG\_readset(aborted)}{AVG\_readset(committed)}$$



Microsoft

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Conclusions

- Applications' internal behavior was analyzed
- When atomic is used for "non-parallelizable" problems, high rollback rates and "late commits" appear.
- Foresight: A smart (dynamic) runtime system could avoid some of the problems that appeared.
- Future work: expand the application set and run it with more cores (128).

Thank you!

Questions?

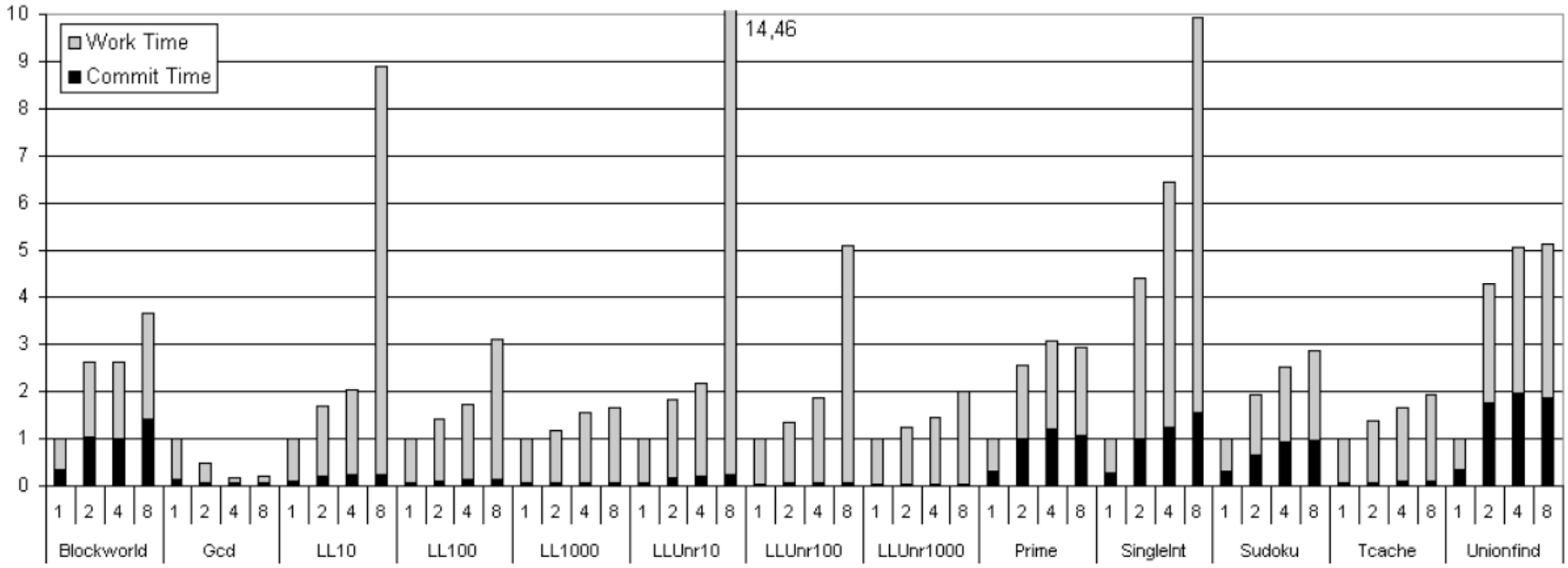
Now or later to *cristian.perfumo@bsc.es*

Microsoft

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Stats: Commit phase overhead

- Commit Overhead



Microsoft

Barcelona Supercomputing Center
Centro Nacional de Supercomputación