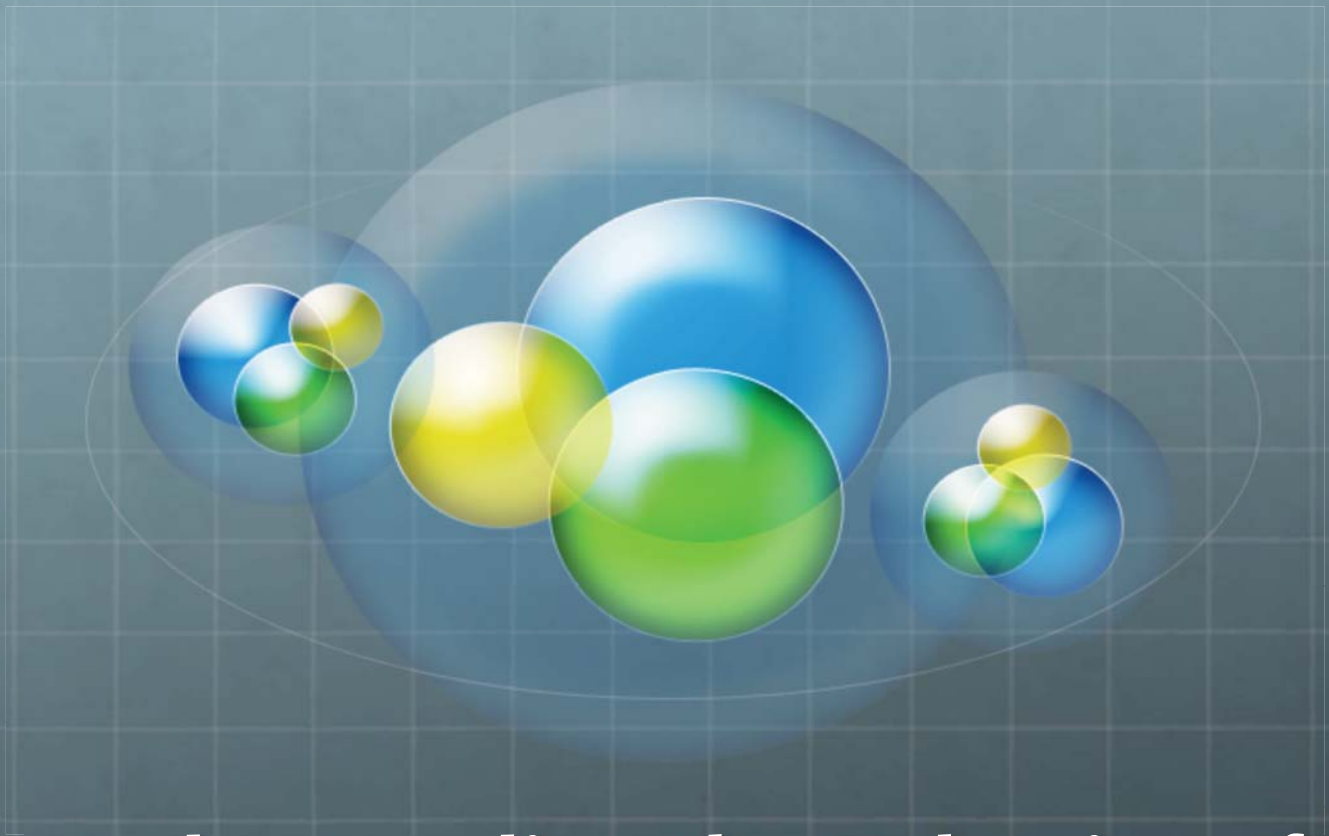


Understanding the Behavior of Transactional Memory Applications

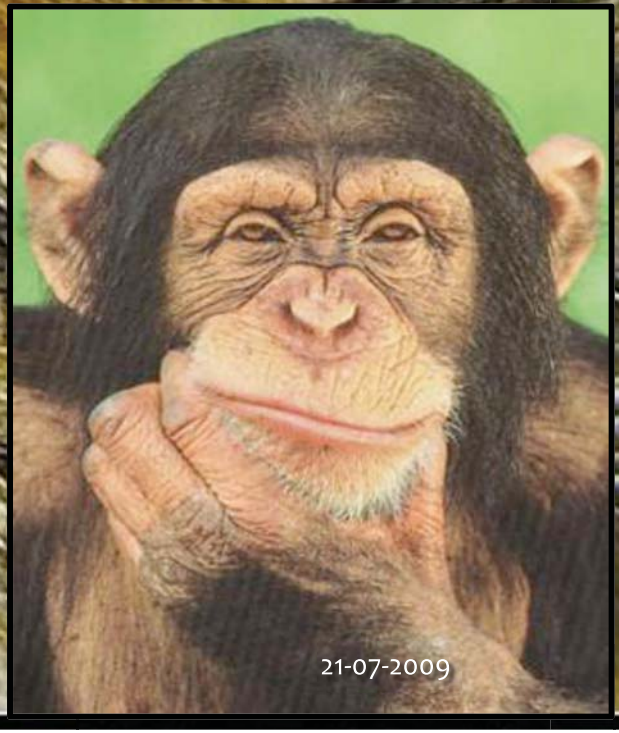
João Loureço Ricardo Dias João Luís
Miguel Rebelo Vasco Pessanha

Universidade Nova de Lisboa, Portugal

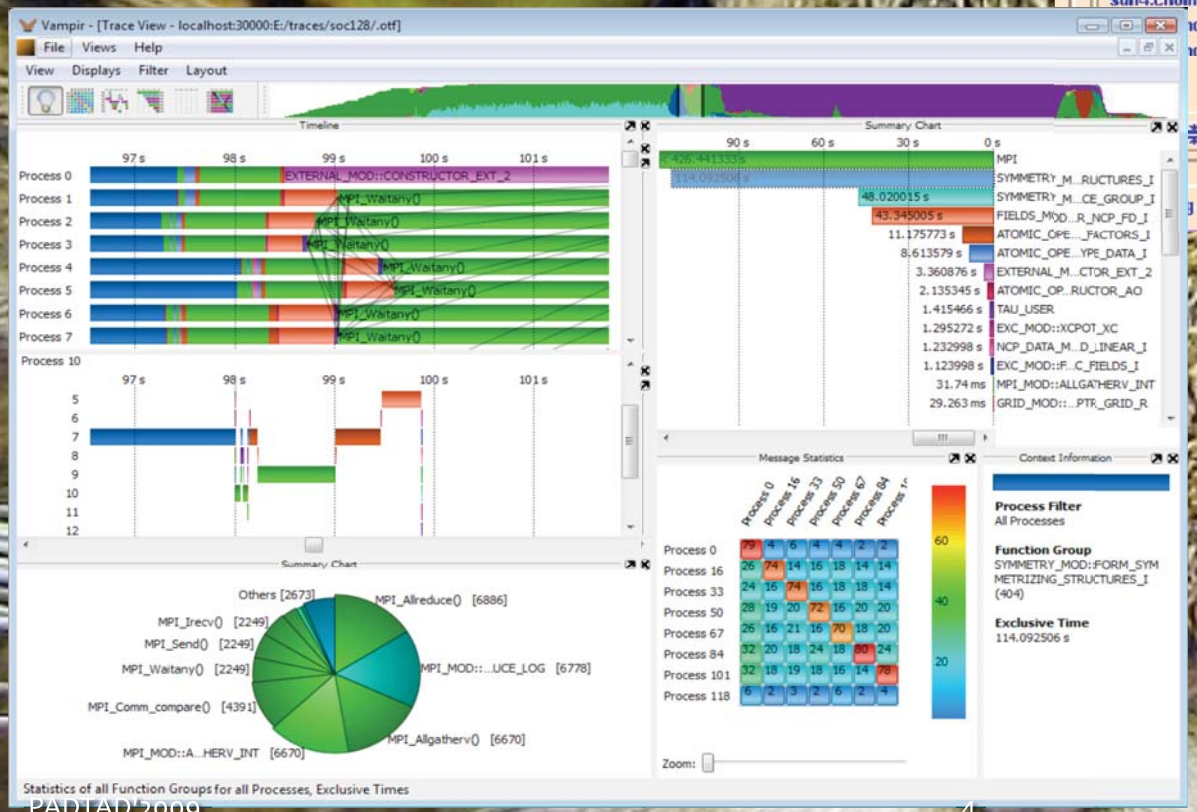
Parallel Programming…
"so what? "

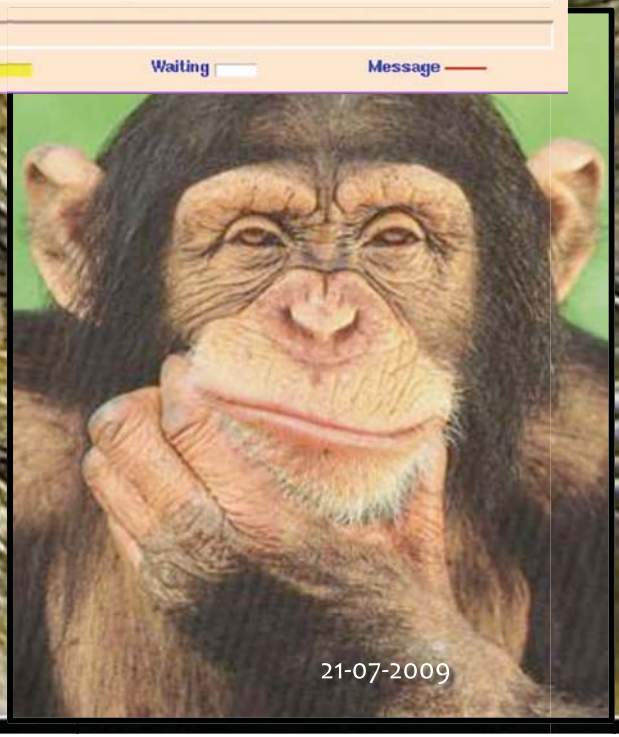
Shared Memory…
“how did I used to do? ”

Lack of tools



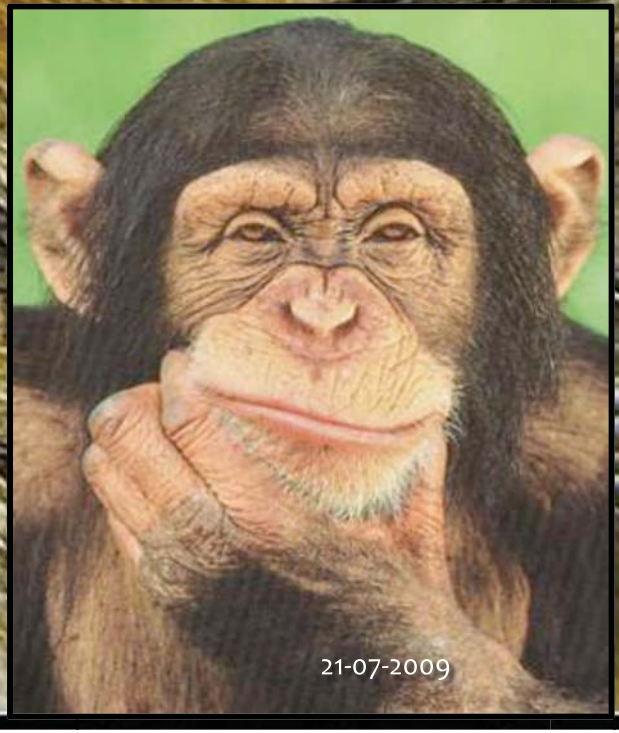
Message Passing…
“how did I used to do?”



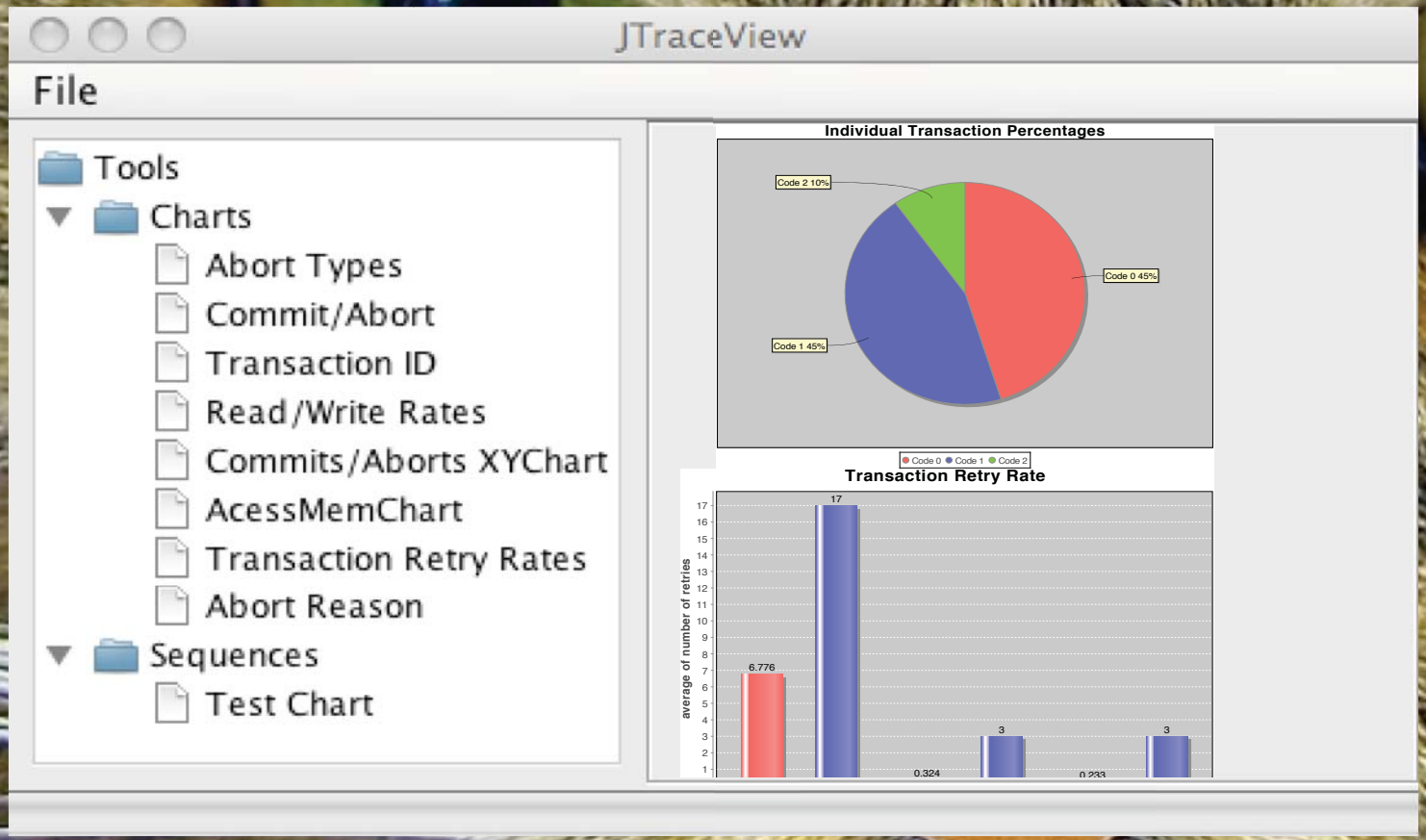


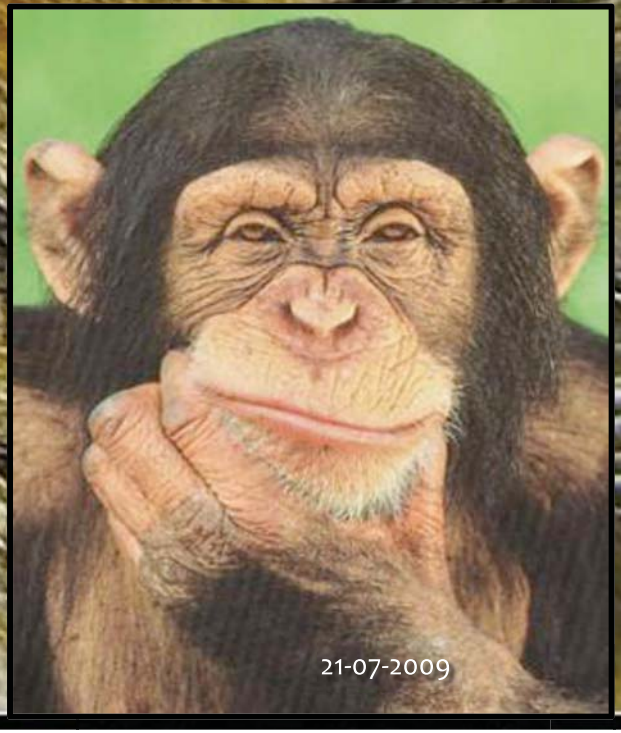
Transactional Memory…
“how did I used to do?”

Lack of tools



Transactional Memory...
"how do I do now!"

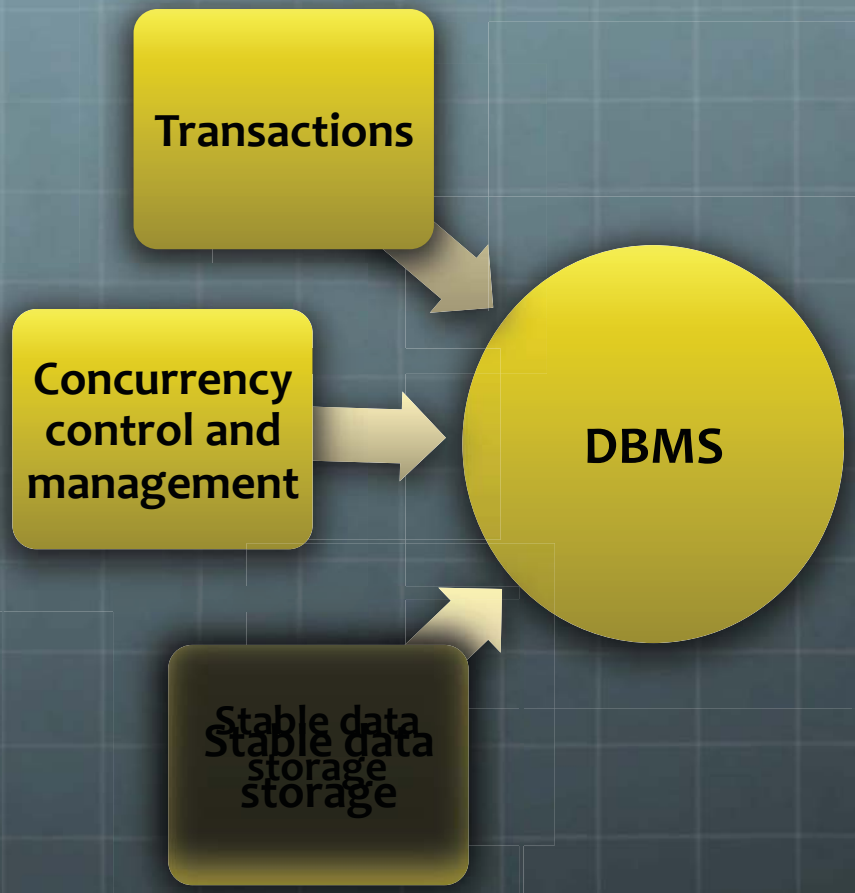


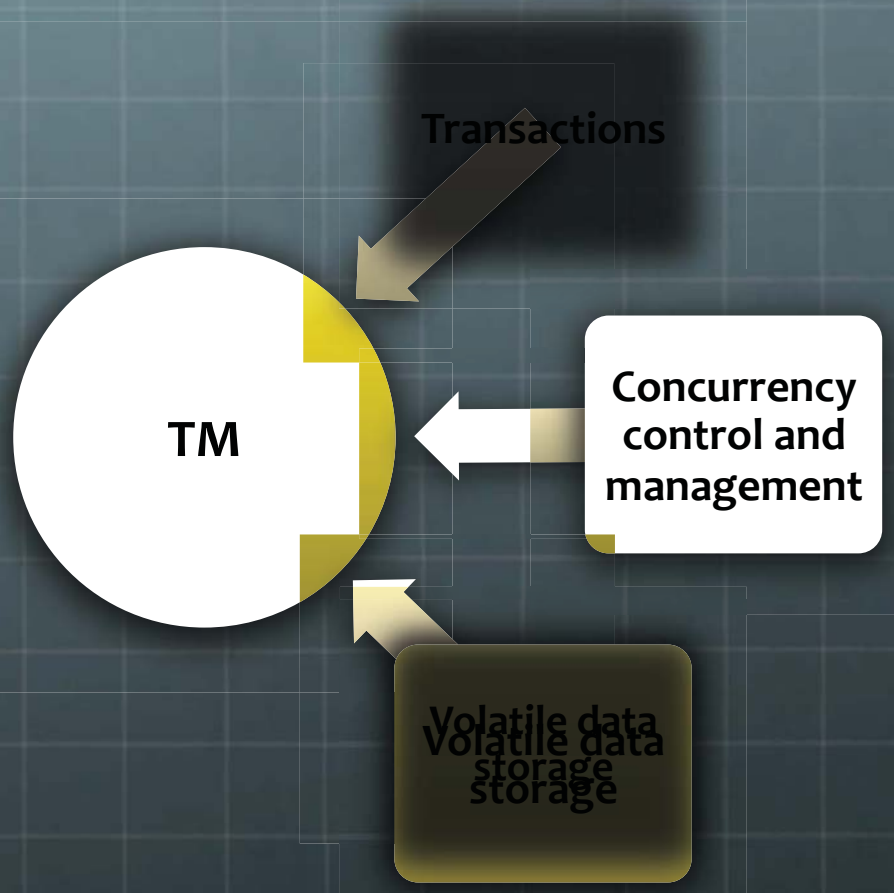


Agenda

- What is TM?
- Specification for a TM event logging format
- A *low overhead* monitoring framework for TM
- A set of charts to visualize the logged data
- Concluding remarks

What is Transactional Memory?





Behavior of Memory Transactions

- **Specification:**
  Tx_Start (Tx_Read | Tx_Write)*
  (Tx_Abort | Tx_Commit)

- **Run-time:**
  Tx_Start (Tx_Read | Tx_Write)*
  (Tx_Abort_Read | Tx_Abort_Write
  | Tx_Abort_Commit | Tx_Abort_User
  | Tx_Commit)

Events for TM Tracing

- All Events

timestamp	eventID	threadID	transID	[?]
• When?	• What?	• Where?	• Whom?	
3576882676744	Start | Load | Store | ...	0 | 1 | ...	0 | 1 | ...	

- Tx_Read | Tx_Write

varID

- Memory location

0x84ef2e8

- Tx_Abort

evType

- Why?

User | Abort | Other

Requirements for Tracing in JTraceView

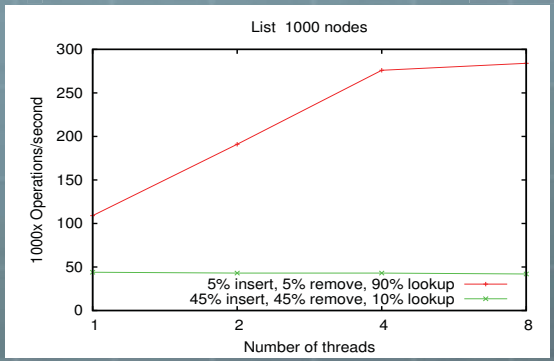
- Keep hardware requirements
  - Different hardware may mask some behaviors and trigger new ones
- Log only well defined time-slots
  - When to start to collect events and for how long
  - Huge log files
- Keep the global application behavior
  - Same behavior pattern
  - Low and constant overhead
  - No additional synchronizations

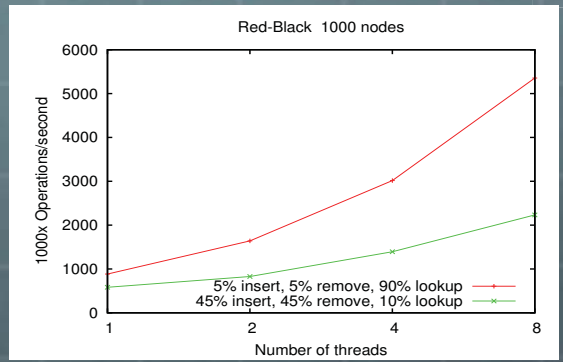
Our Proposal for Tracing in JTraceView

- Keep events in separate buffers
  - One buffer per thread
  - Avoids additional synchronizations
- Use RDTSC (or similar) registers as a global clock
  - Clock drifting could be a problem, but it's not (in general)
  - Provides info on real execution time for transactions
- Keep global application behavior

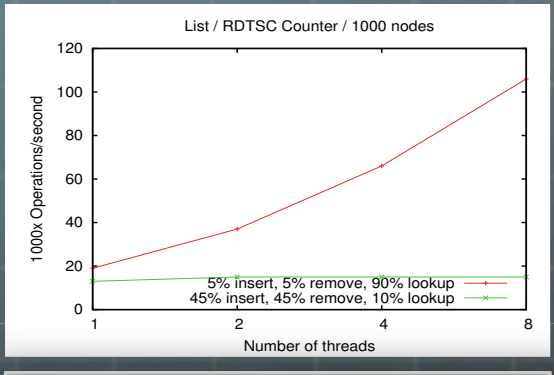
Keeping App Behavior

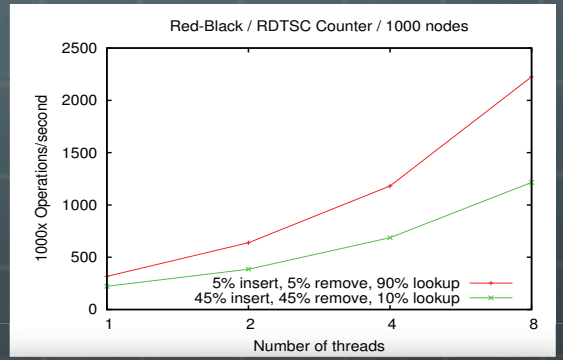
- Linked List
- Red-Black Tree



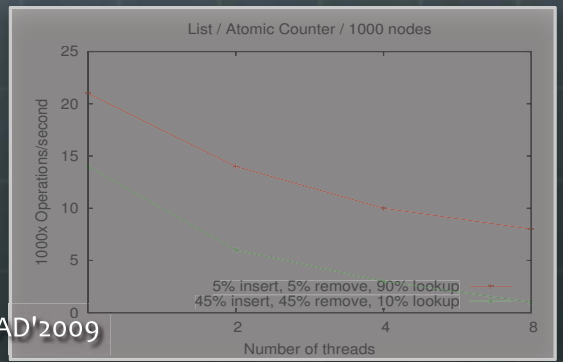


Unmonitored

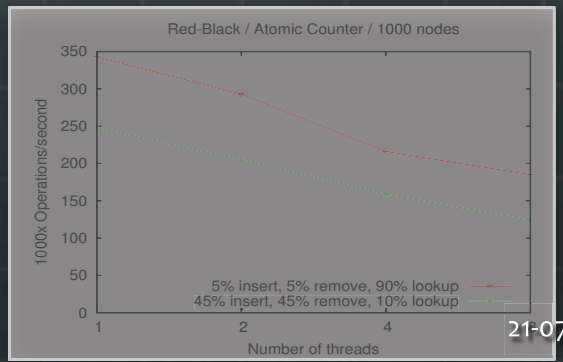




JTraceView tracing



Naïve tracing



Tracing System Output

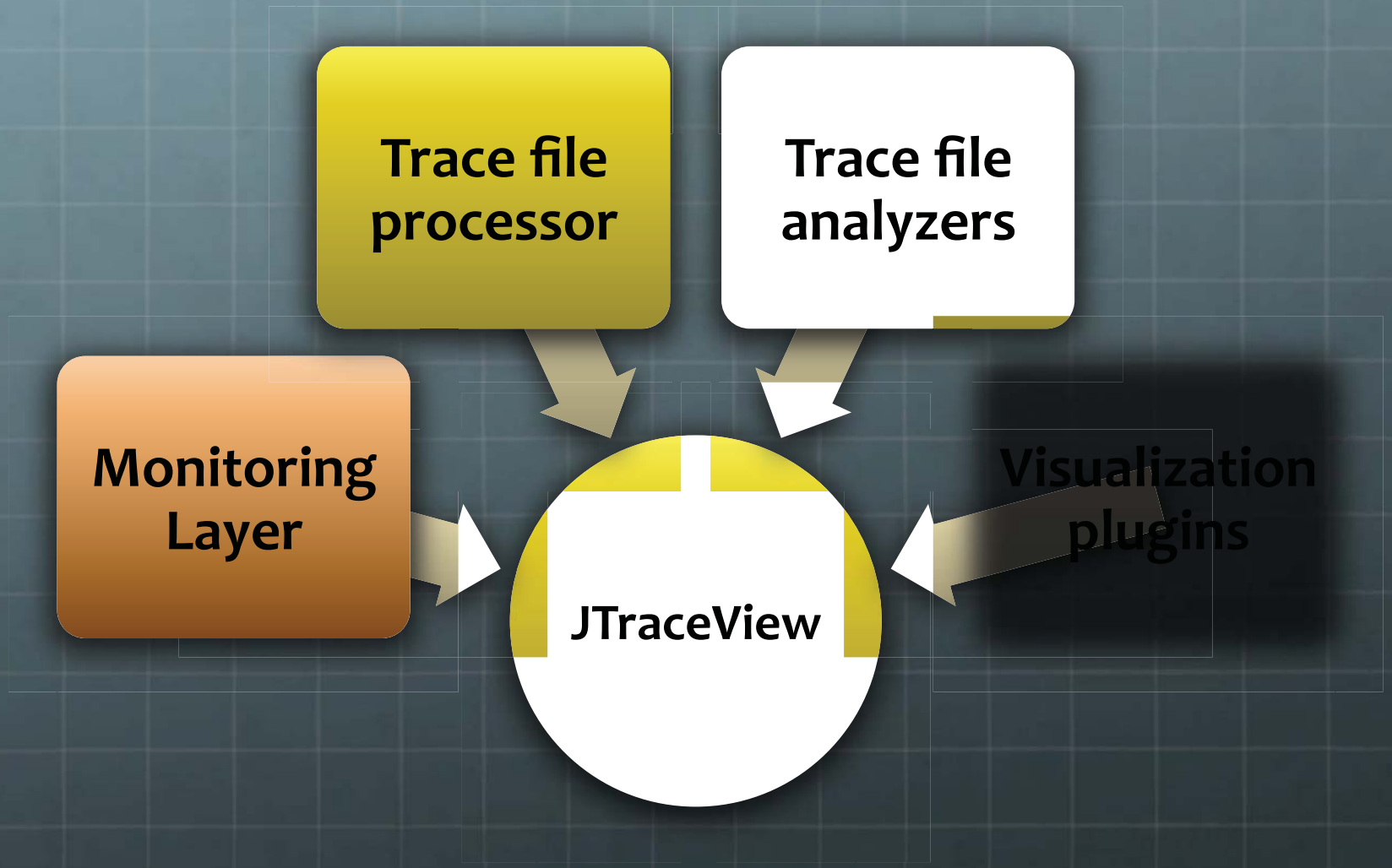
3576882676772344	tx_read	T2	1	0x84ef2e8
3576882676773517	tx_read	T2	1	0x84f0d50
3576882676774549	tx_read	T2	1	0x84ef2a0
3576882676775661	tx_read	T2	1	0x0
3576882676776340	tx_read	T2	1	0x128
3576882676777018	tx_write	T2	1	0x127
3576882676777932	tx_read	T2	1	0x18c
3576882676778553	tx_write	T2	1	0x18b
3576882676779150	tx_read	T2	1	0x0
3576882676780071	tx_read	T2	1	0x0
3576882676780703	tx_read	T2	1	0x84f0d50
3576882676781647	tx_read	T2	1	0x0
3576882676782276	tx_read	T2	1	0x84f0[illegible]
3576882676782893	tx_read	T2		0x84e[illegible]
3576882676783823	tx_write	T2		0x84[illegible]
3576882676784422	tx_write	T2	1	
3576882676785040	tx_commit	T2		
3576882676791481	tx_start	T2		
3576882676792077	tx_read	T2		
3576882676793013	tx_read	T2		
3576882676793651	tx_read	T2		
3576882676794275	tx_read	T2	2	
3576882676795214	tx_read	T2	2	84[illegible]
3576882676795835	tx_read	T2	2	0xb0
3576882676796464	tx_read	T2	2	0x84ef[illegible]
3576882676797411	tx_read	T2	2	0x8f
3576882676798039	tx_read	T2	2	0x84ef268
3576882676799270	tx_read	T2	2	0x80
3576882676800314	tx_read	T2	2	0x84f0ea0
3576882676801265	tx_read	T2	2	0x86
3576882676802315	tx_read	T2	2	0x84f0c18
3576882676802934	tx_read	T2	2	0x87
3576882676803569	tx_read	T2	2	0x0
3576882676804519	tx_commit	T2	2	
3576882676807718	tx_start	T2	2	
3576882676808301	tx_read	T2	2	0x84dd600
3576882676808920	tx_read	T2	2	0xd3
3576882676809561	tx_read	T2	2	0x84eb408
3576882676810472	tx_read	T2	2	0x78
3576882676811097	tx_read	T2	2	0x84f0fc0
3576882676811709	tx_read	T2	2	0xb0

RB-Tree in CTL generates 1 KB per processor per milisecond

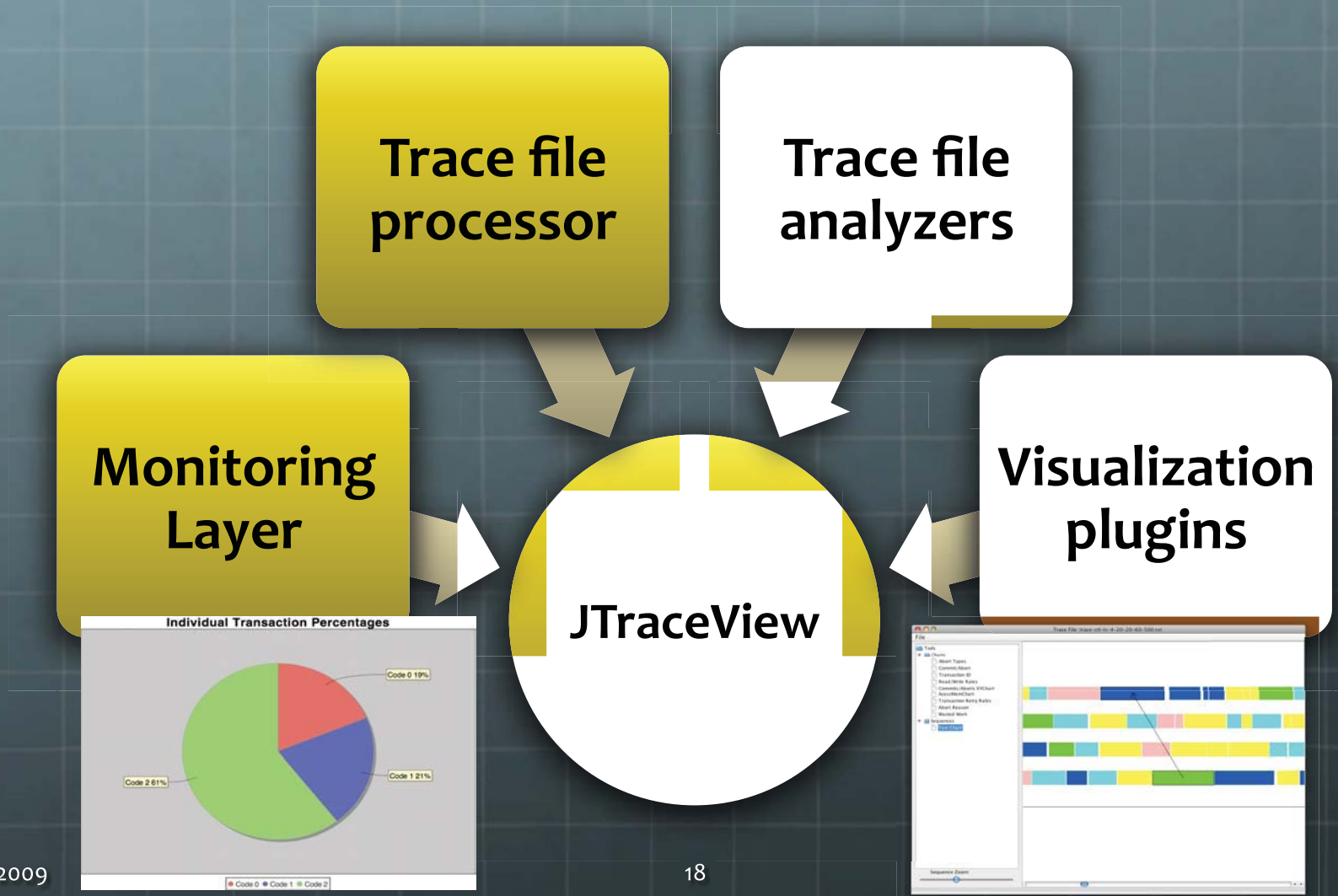
Human Unmanageable

even when converted to text

Visualization Tool for TM

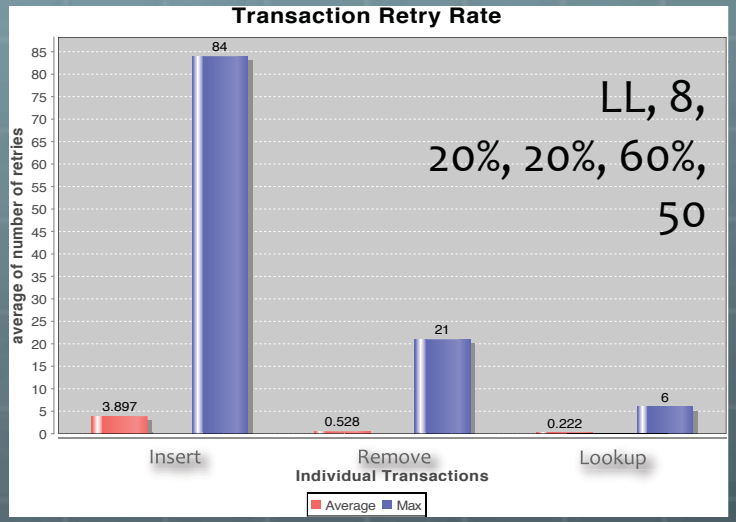


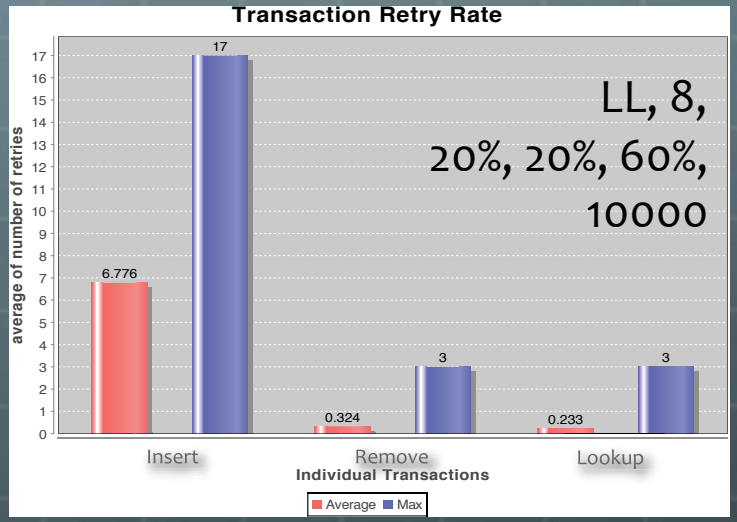
Visualization Tool for TM

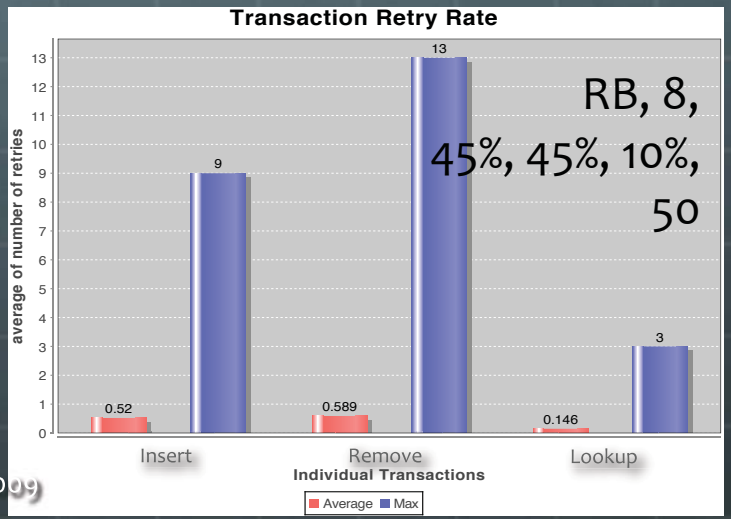


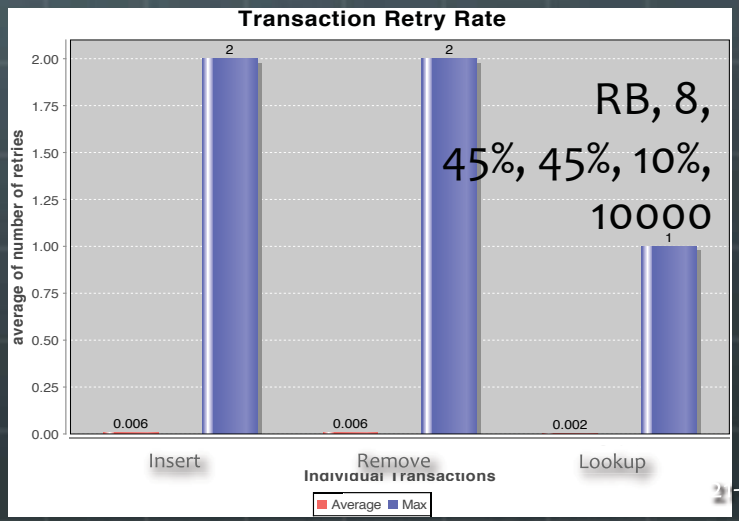
Visualization Plugins

Example

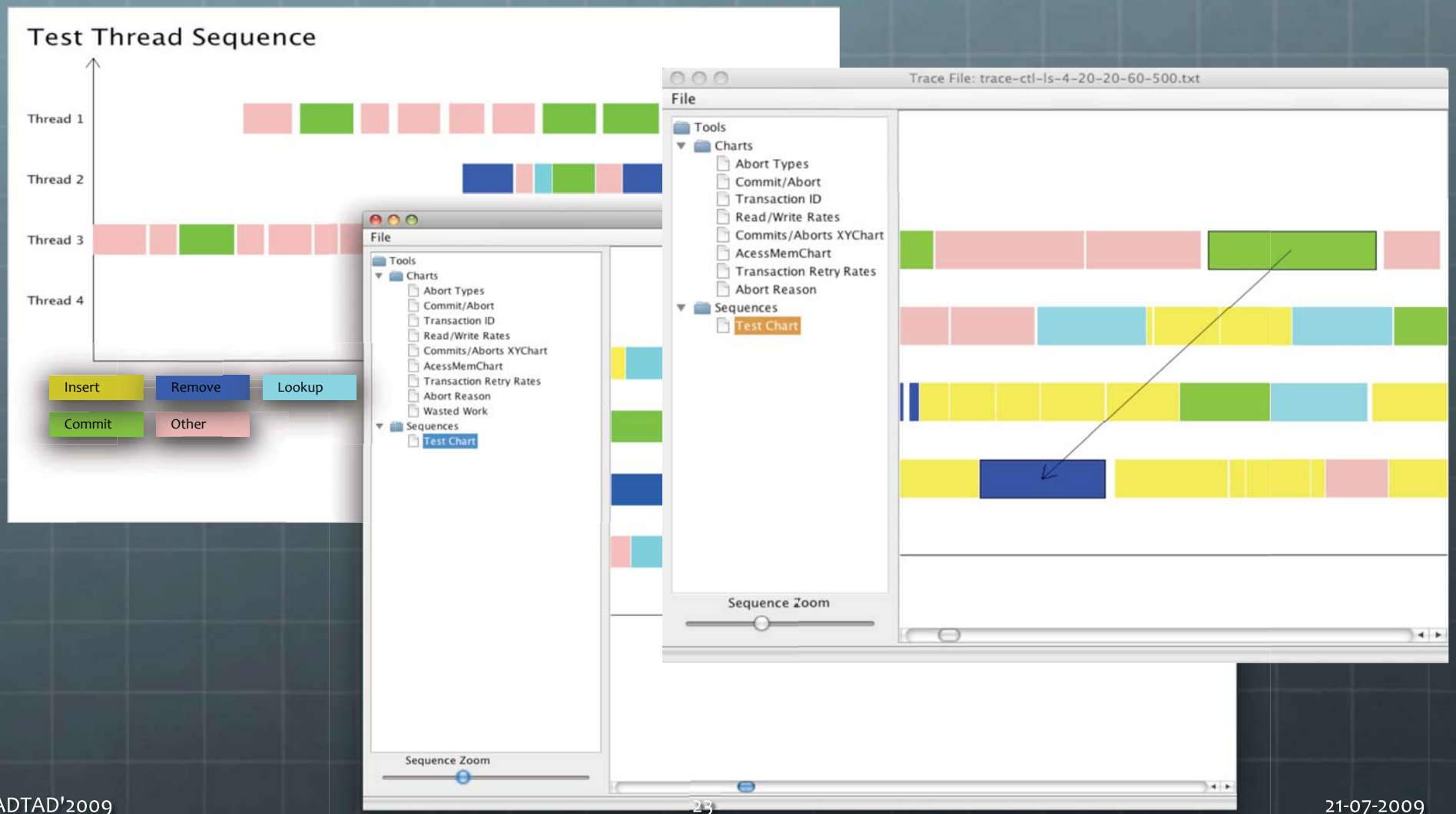

Transaction Retry Rate
average of number of retries
3.897
84
0.528
21
0.222
6
Insert
Remove
Lookup
Individual Transactions
Average
Max
LL, 8,
20%, 20%, 60%,
50


Transaction Retry Rate
average of number of retries
6.776
17
0.324
3
0.233
3
Insert
Remove
Lookup
Individual Transactions
Average
Max
LL, 8,
20%, 20%, 60%,
10000


Transaction Retry Rate
average of number of retries
0.52
9
0.589
13
0.146
3
Insert
Remove
Lookup
Individual Transactions
Average
Max
RB, 8,
45%, 45%, 10%,
50


Transaction Retry Rate
average of number of retries
0.006
2
0.006
2
0.002
1
Insert
Remove
Lookup
Individual Transactions
Average
Max
RB, 8,
45%, 45%, 10%,
10000

Another example



Concluding Remarks

- JTraceView includes a low intrusion monitoring layer
- Deals with the very high frequency of events and huge logs
- The huge logs can be visually synthesized
- Time-space view can also help in debugging
- Visualization tools such as JTraceView are a good approach to better understand the behavior of transactional memory applications

Future Work

- Analyzing more complex applications
  - Use traces from STAMP apps (and others)
  - Support for online visualization and state-based debugging
  - Identify highly-conflicting transactions and apply alternative synchronization methods
- The tool is still in development:
  - Heuristics for correcting clock-drifting
  - More plugins with charts
  - More interactions in the time-space view
  - New types of visualization
  - Possible integration with an IDE (eclipse?)

Thank you!