Tarantool/Box:
persistent in-memory queues

kostja@tarantool.org

NoSQL Matters, Cologne, Apr 27 2013

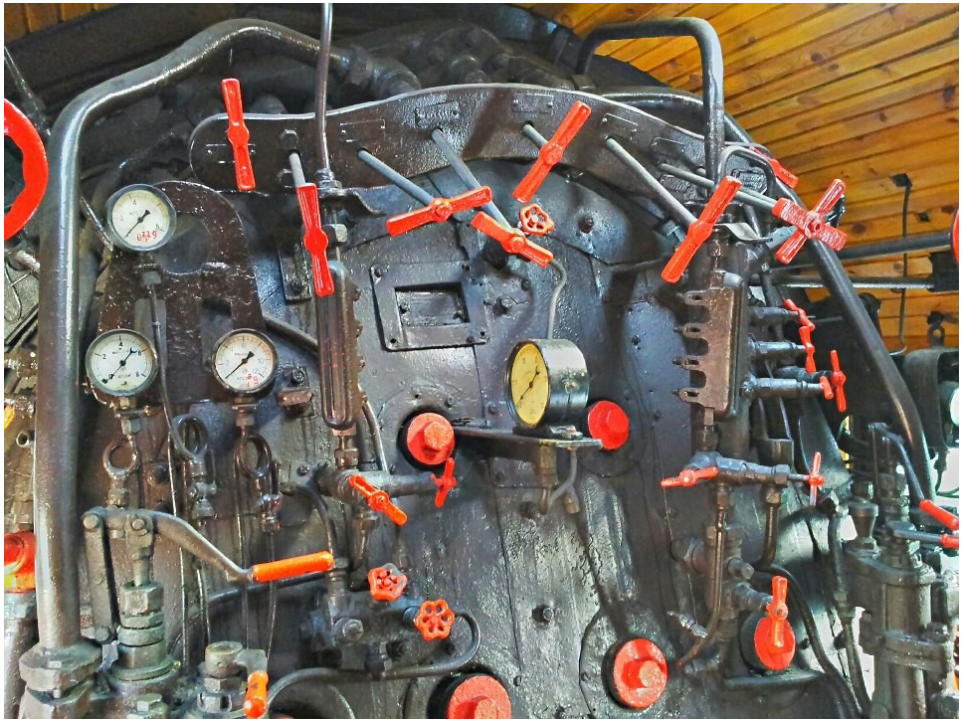
Tarantool/Box: introduction

- data types: NUM, NUM64, BLOB
- TREE, HASH, BITSET index
- Replication & Online backup
- Extensibility with Lua procedures, triggers, events

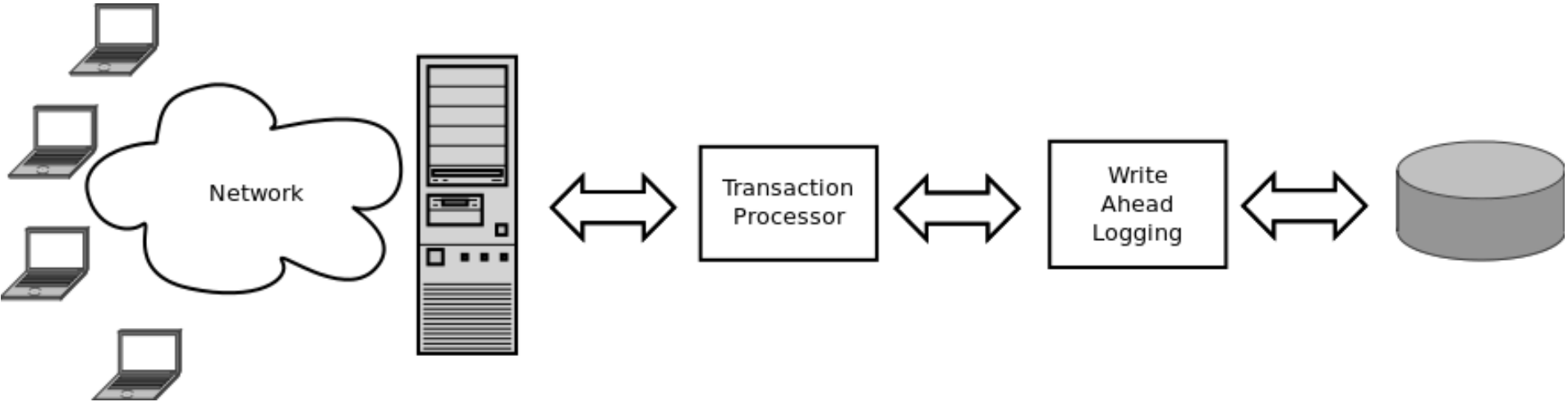
Tarantool: introduction (2)

- All data is maintained in memory, write ahead logged
- Data atom is tuple, <field, field, ...>
- Memcache, iproto, text protocols
- Primary, secondary, multipart keys

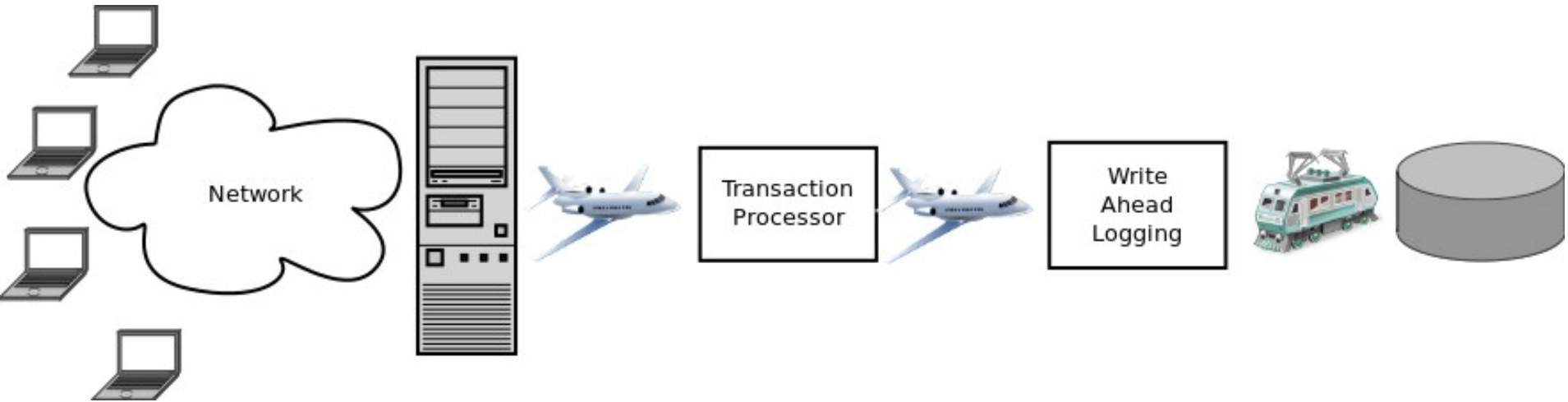
Understanding a database



Tarantool: architecture



Tarantool: architecture (2)

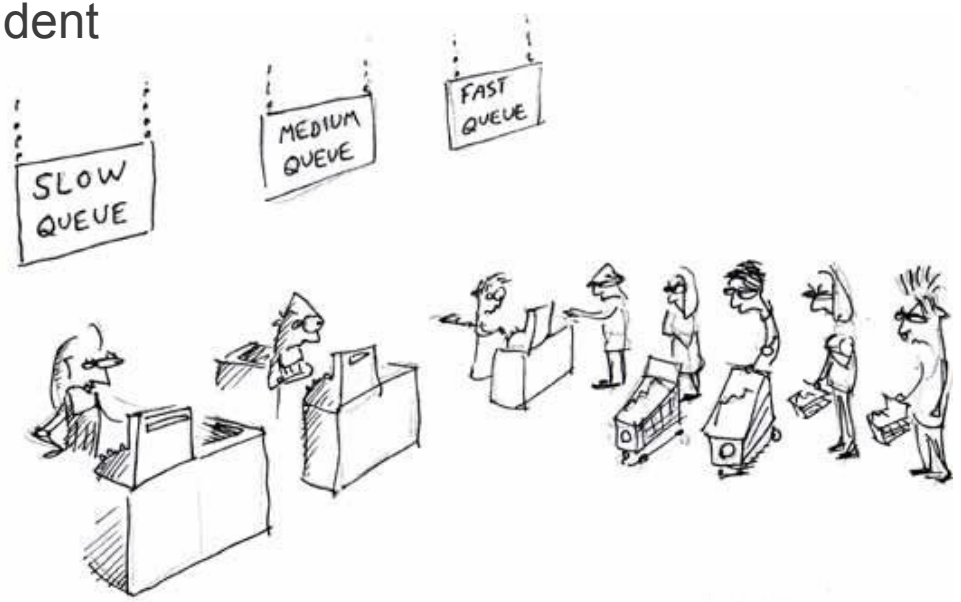


Tarantool/Queues

- RDBMS is one of the first queue engines
- Transactions are a necessity, but there is more to it
- A building block of High Availability & Scalability

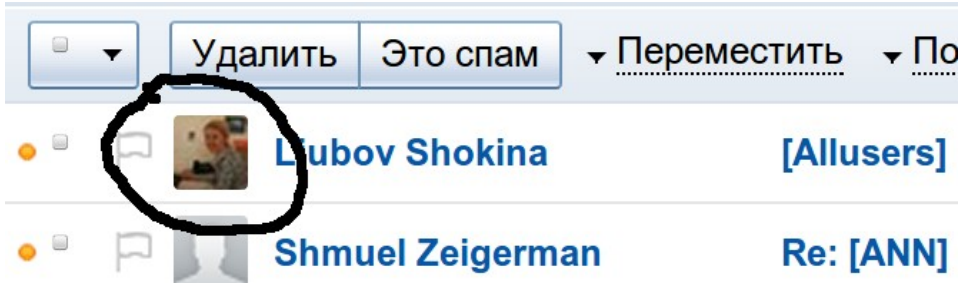
Queues: pro & cons

+ client and the server are independent
+ state of the task is always known
+ can be used for Load Balancing
+ supports task priorities
- an additional point of failure
- added processing overhead



Tarantool/Queues in Web Apps

- Email notifications
- File upload
- Fetching user avatars



Tarantool: features for Queues

- Rich Lua procedures:
  - `box.fiber, box.cfg, box.space`
- Inter-procedure communication
  - `box.ipc.channel`
- Specialized data structures
  - Partial and bitmap keys

Queue API

- A copy-cat of **beanstalkd** API
- `queue.put(), take(), ack(), delete(), release()`
- Important problems of queue management are taken care of:
  - Task priorities
  - Task timeouts
  - Poisoned tasks

Runaway tasks

- box.fiber.id(): session identifier
- box.queue.take() assigns tasks a session id
- on_disconnect trigger puts back runaway tasks to «ready» queue

Waiting on a task

- box.ipc.channel(): inter-procedure communication pipe
- channel:put(), channel:get()
- channel:is_full(), channel:has_waiters()

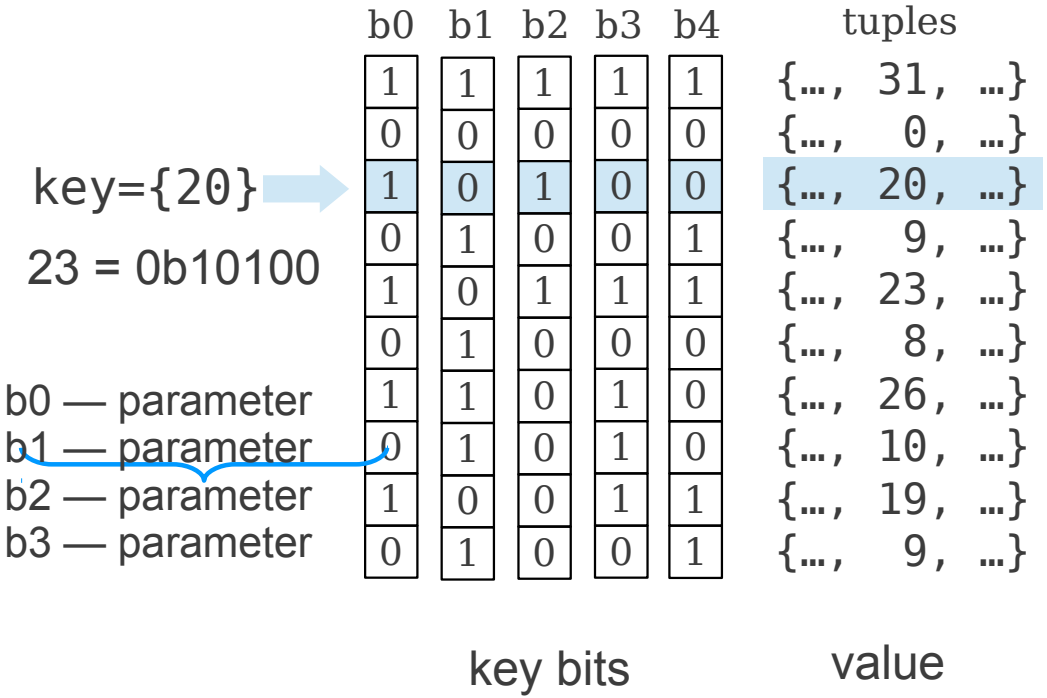
Tarantool: triggers

- Invoked upon a system event
- on_connect(), on_disconnect()
- instead_of()
- pay per use

Partial keys

- index.type = HASH|TREE
- index.where = lua_function
- Only contains data matching a predicate

BITSET keys


b0 b1 b2 b3 b4
tuples
key={20}
23 = 0b10100
b0 — parameter
b1 — parameter
b2 — parameter
b3 — parameter
1 1 1 1 1 {…, 31, …}
0 0 0 0 0 {…, 0, …}
1 0 1 0 0 {…, 20, …}
0 1 0 0 1 {…, 9, …}
1 0 1 1 1 {…, 23, …}
0 1 0 0 0 {…, 8, …}
1 1 0 1 0 {…, 26, …}
0 1 0 1 0 {…, 10, …}
1 0 0 1 1 {…, 19, …}
0 1 0 0 1 {…, 9, …}
key bits
value

Conclusion

- Tarantool/Box — high-performance data engine
- Tarnatool/Lua — a building block for a scalable web appilcation
- Tarantool/Queue – a case for a versatile in-memory database

Refs

http://github.com/mailru/tarantool - source code
http://github.com/mailru/tntlua - stored procedure repository
http://nosql-databases.org - alternative NoSQL solutions
http://groups.google.com/group/tarantool-ru - mailing list
http://tarantool.org/dist/ - always fresh .tar.gz and .rpm

kostja@tarantool.org

http://tarantool.org

?

Refs

http://github.com/mailru/tarantool - source code
http://github.com/mailru/tntlua - stored procedure repository
http://nosql-databases.org - alternative NoSQL solutions
http://groups.google.com/group/tarantool-ru - mailing list
http://tarantool.org/dist/ - always fresh .tar.gz and .rpm

kostja@tarantool.org

http://tarantool.org