

Tarantool/Box: persistent in-memory queues

kostja@tarantool.org

NoSQL Matters, Cologne, Apr 27 2013

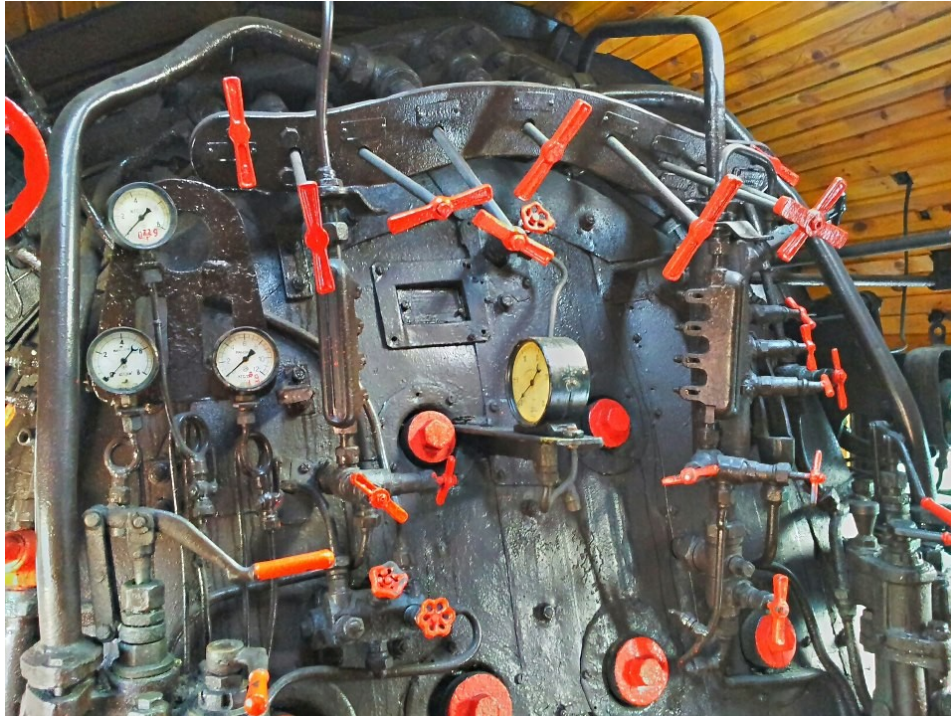
Tarantool/Box: introduction

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

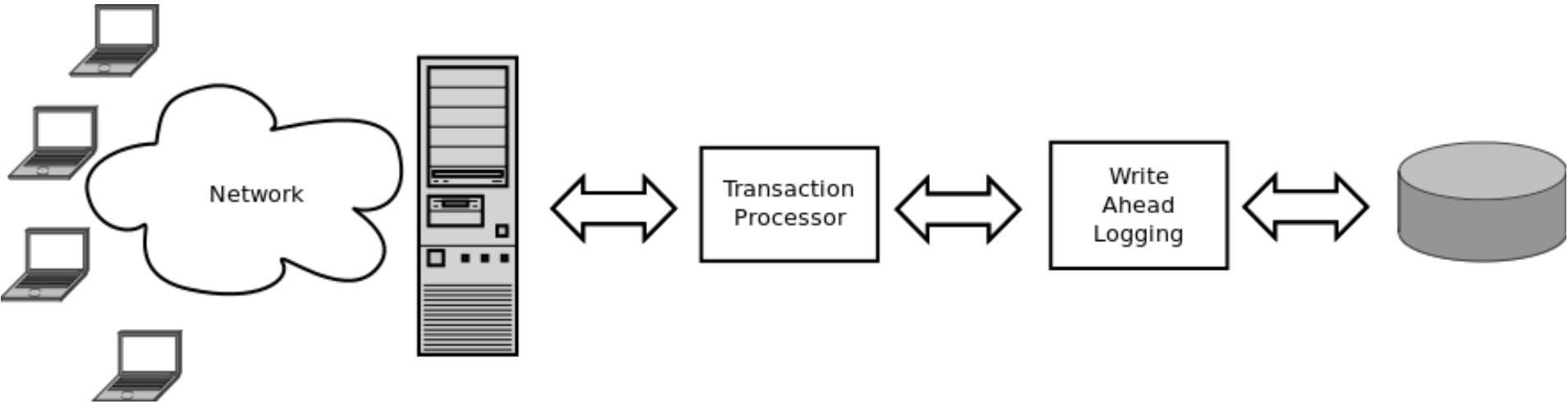
Tarantool: introduction (2)

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

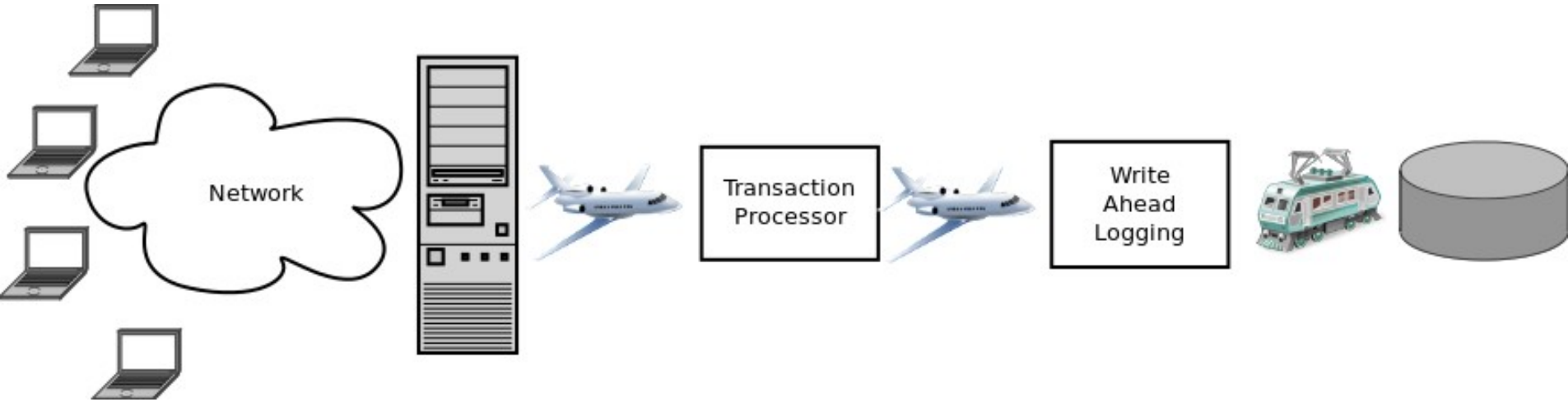
Understanding a database



Tarantool: architecture



Tarantool: architecture (2)



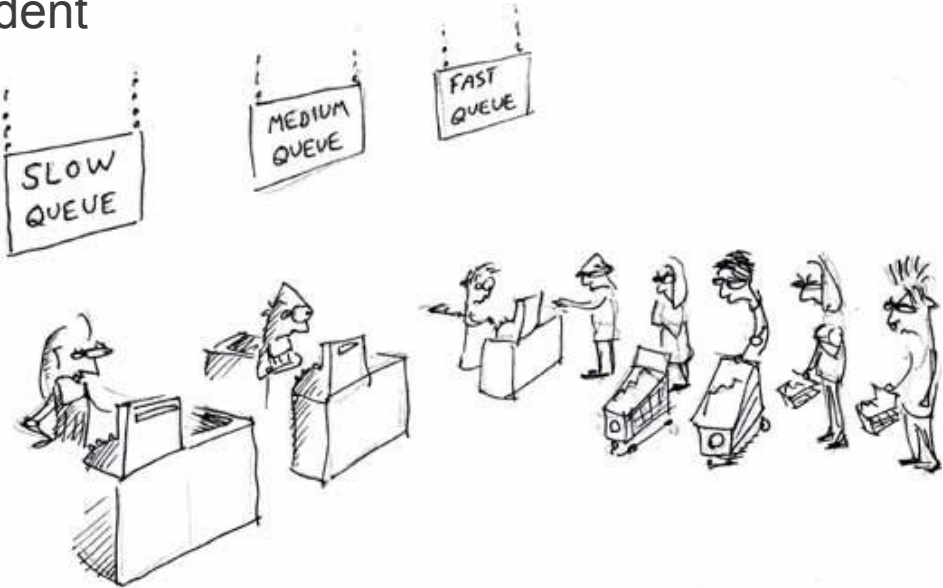
Tarantool/Queues

- 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



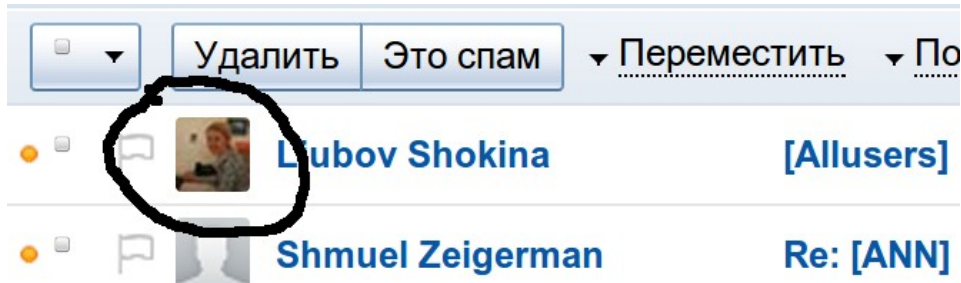
Queues: pro & cons

- + 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

```
181
182 function box.fqueue.put(sno, tube, prio, ...)
183     sno, tube, prio= tonumber(sno), tonumber(tube), tonumber(prio)
184     if not box.fqueue.enabled[sno] then error("Space '.. sno .. ' queue not enabled") end
185     if prio == nil then prio = 0x7fff end
186     if tube == nil then tube = 0 end
187     local status = 'R'
188     local id = box.fqueue.id(sno)
189     --             id, tube, prio, status, taken, buried,
190     return box.insert(sno, id, tube, prio, status, 0, 0, ...)
191 end
192
193 function box.fqueue.delete(sno, id)
194     sno, id = tonumber(sno), tonumber(id)
195     if not box.fqueue.enabled[sno] then error("Space '.. sno .. ' queue not enabled") end
196     box.fqueue.taken[box.pack('l', id)] = nil
197     return box.space[sno].delete(id)
198 end
199
200     -- box.ipc.channel
201
202 function box.fqueue.take(sno, tube)
203     if not box.fqueue.enabled[sno] then error("Space '.. sno .. ' queue not enabled") end
204     if tube == nil then tube = 0 end
205     local x, one_ready = box.space[sno].select{ where="key <= ? and type=key", tuple_equal( tube, 'R' ) }
206     if one_ready == nil then return end
207     print("Selected '.. tostring( box.unpack('l', one_ready[0]) )..' for take ")
208     box.fqueue.taken[ one_ready[0][1] = box.fiber.id()
```

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
	1	1	1	1	1	{..., 31, ...}
	0	0	0	0	0	{..., 0, ...}
key={20} →	1	0	1	0	0	{..., 20, ...}
	0	1	0	0	1	{..., 9, ...}
23 = 0b10100	1	0	1	1	1	{..., 23, ...}
	0	1	0	0	0	{..., 8, ...}
b0 — parameter	1	1	0	1	0	{..., 26, ...}
b1 — parameter	0	1	0	1	0	{..., 10, ...}
b2 — parameter	1	0	0	1	1	{..., 19, ...}
b3 — parameter	0	1	0	0	1	{..., 9, ...}

key bits value

Conclusion

- Tarantool/Box — high-performance data engine
- Tarantool/Lua — a building block for a scalable web application
- 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>

