Tarantool/Box: 2 billion queries a day at Mail.Ru

Konstantin Osipov, Mail.Ru

Talk agenda

A step back: what's wrong with relational databases?

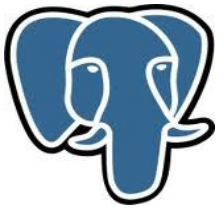
Tarantool/Box: an in-memory DBMS

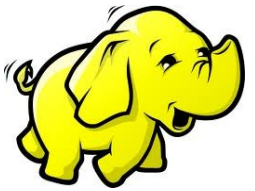
Data access patterns

Scalability patterns

Plans & projections

Database landscape: 2012





What's wrong with relational databases?

- difficulty of horizontal scaling
- rigid schema
- overhead of outdated architectures

NoSQL is often about a rewrite from scratch:

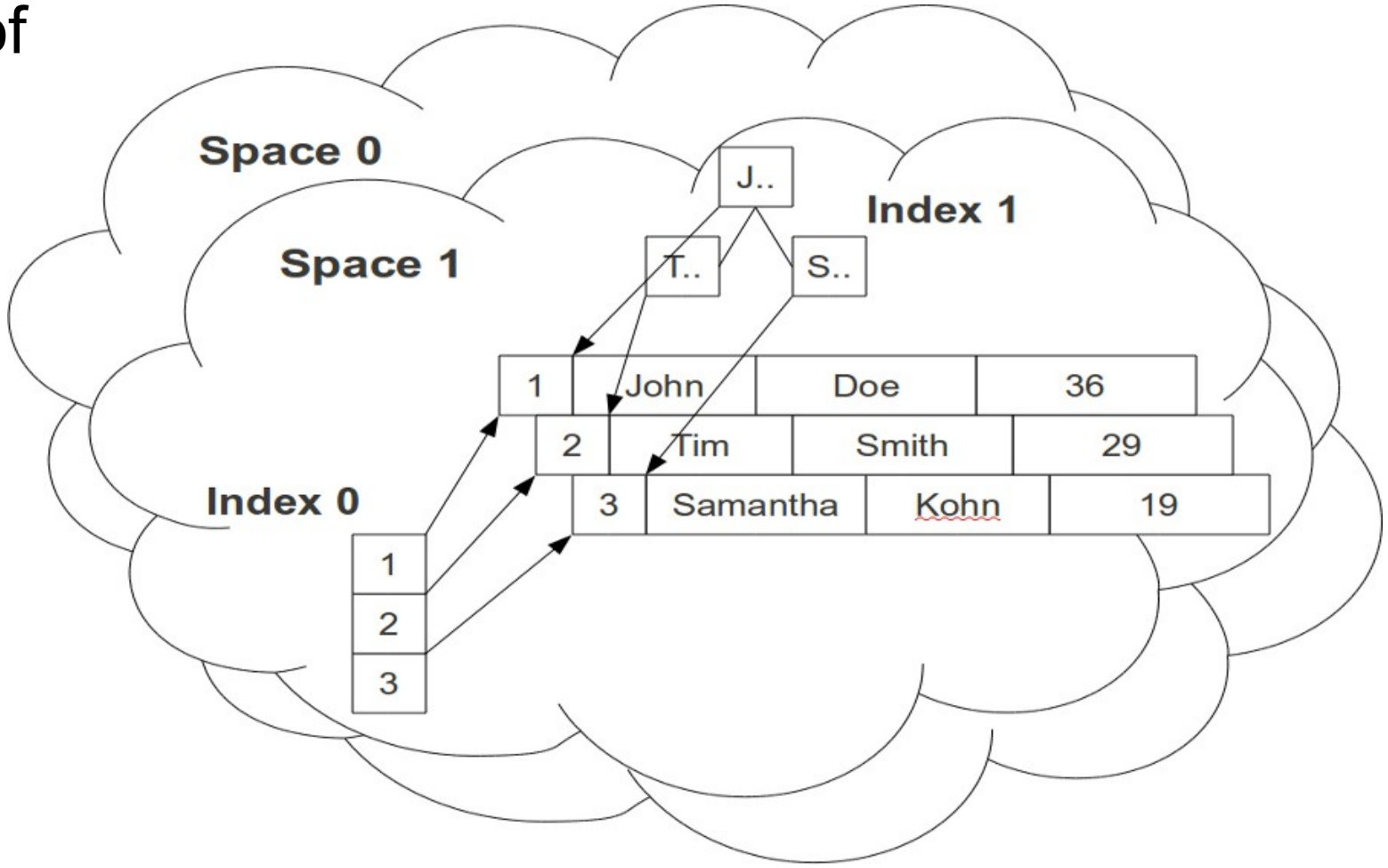
- new data models, data consistency models
- new data access languages
- new algorithms for data storage
- lots of new approaches to scalability

Tarantool/Box: an overview

- fully cached: 100% of data is cached in RAM
- data is persistent on disk
- flexible data model
- log shipping replication and online backup
- stored procedures in Lua
- memcached mode and a compact binary protocol

Data model

A game of spaces, tuples and keys.



Data model (2)

- HASH and TREE indexes
- multipart, unique, non-unique indexes
- STRING, NUM, NUM64 data types

uid	channel_id	atime	new	list			
kostja@	36	03/29/12	0				
avs@gma.	124	03/30/12	11	Fish Hooks ...	MMA Live	Games and...	Oakley Big...
anna@en	511	03/31/12	0				
devo4ka@	338	04/01/12	0				
leok@mail	334	04/02/12	20	Natsuiro Kis...	Monokone	Eureka Sev...	Natsuiro Kis.
mfk007@y	511	04/03/12	3	Dancing with...	The Hard Times	Impact World	
natalka@y	36	04/04/12	1	Animal Planet			
dm@yand	334	04/05/12	0				

Data model: summary

Terminology:

Tarantool	RDBMS
Space	Table
Field	Column

Data types:

NUM
NUM64
STRING

Index types:

HASH
TREE

Ruby API

```
DB = Tarantool.new host: 'localhost', port: 33013
space = DB.space 0

space.insert 'prepor', 'Andrew', 'ceo@prepor.ru'
res = space.select 'prepor'
puts "Name: #{res.tuple[1].to_s}; Email:
#{res.tuple[2].to_s}"
space.delete 'prepor'
```

ActiveModel and **EventMachine** are supported as well.

PHP API

```
define('SNO',0);        //  space id
$key = 12345
$tuple = array($key,'spb','Hello Word');
$res = $tnt->insert(SNO, $tuple);
$res = $tnt->delete(SNO, $key, [$flag]);
$data = array(1 => 'Cologne', 2 => 'NoSQL-Matters');
$res = $tnt->update(SNO, $key, $data);
# $key - always the primary key
# $data - PHP array field no => new value
```

PHP API: SELECT

```
$count = $tnt->select(SNO, $index, $key,[$limit,
$offset]);
# $key - used index, multiple keys are allowed;
# $index - index id, used for querying
# default $limit = 0xFFFFFFFF, $offset = 0;
# Returns the number of found tuples
$tuple = $tnt->getTuple();
```

Lua API

Redis	Tarantool (Lua)
redis.set(key, value)	box.insert(space, key, value)
redis.get(key)	box.select(space, 0, key, value)
redis.getset(key, newkey)	box.update(space, key, '=p', 0, newkey)
redis.incr(key)	box.update(space, key, '+p', 1, 1)
redis.lpush(key, value)	box.update(space, key, '!p', 1, value)
redis.rpush(key, value)	You guess it...

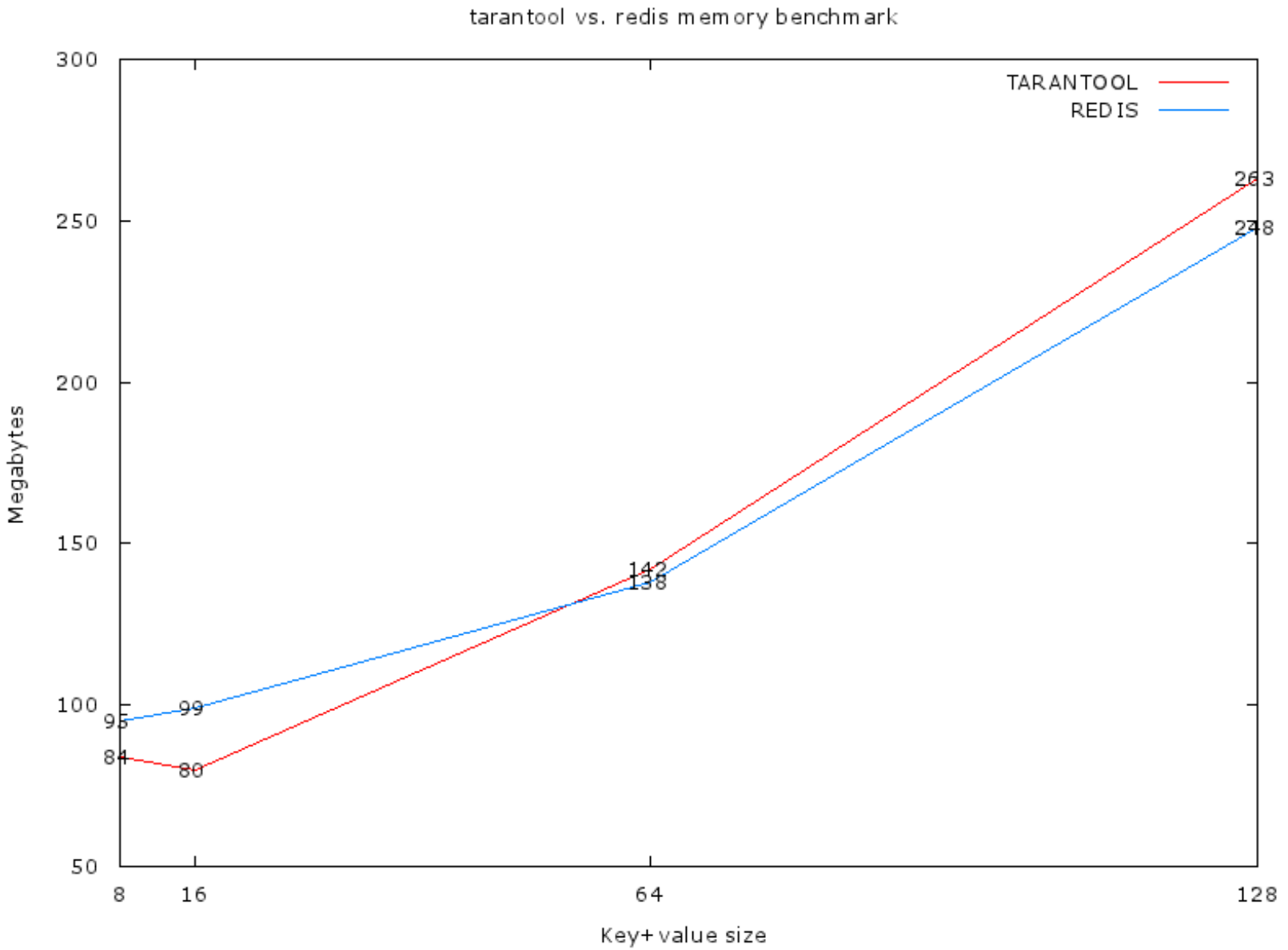
Performance

Intel I5 , 4G RAM, 7200 RPM SATA

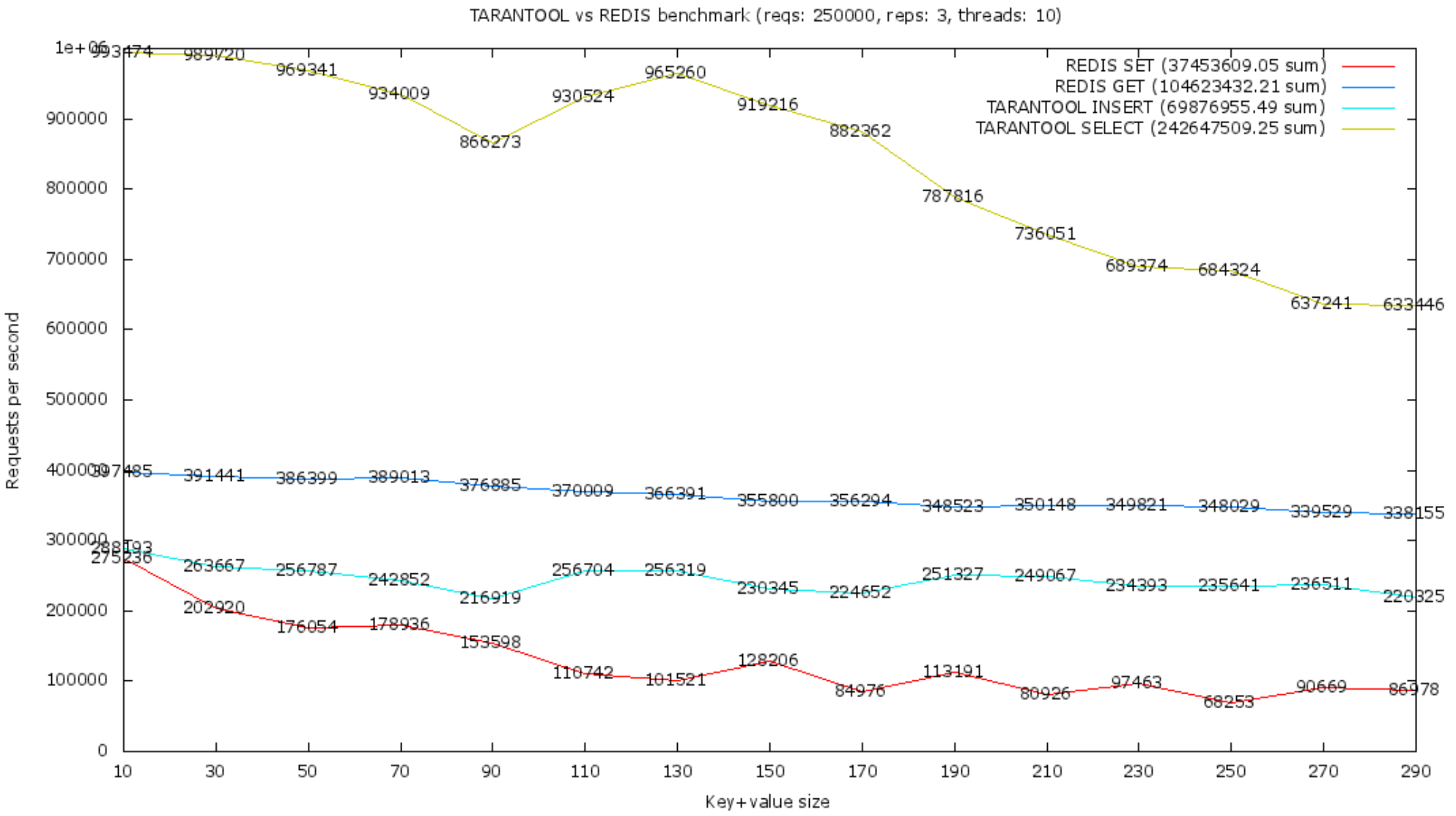
10 threads, 200-300 bytes per tuple

Tarantool 1.4.6: 170k writes, 260k reads

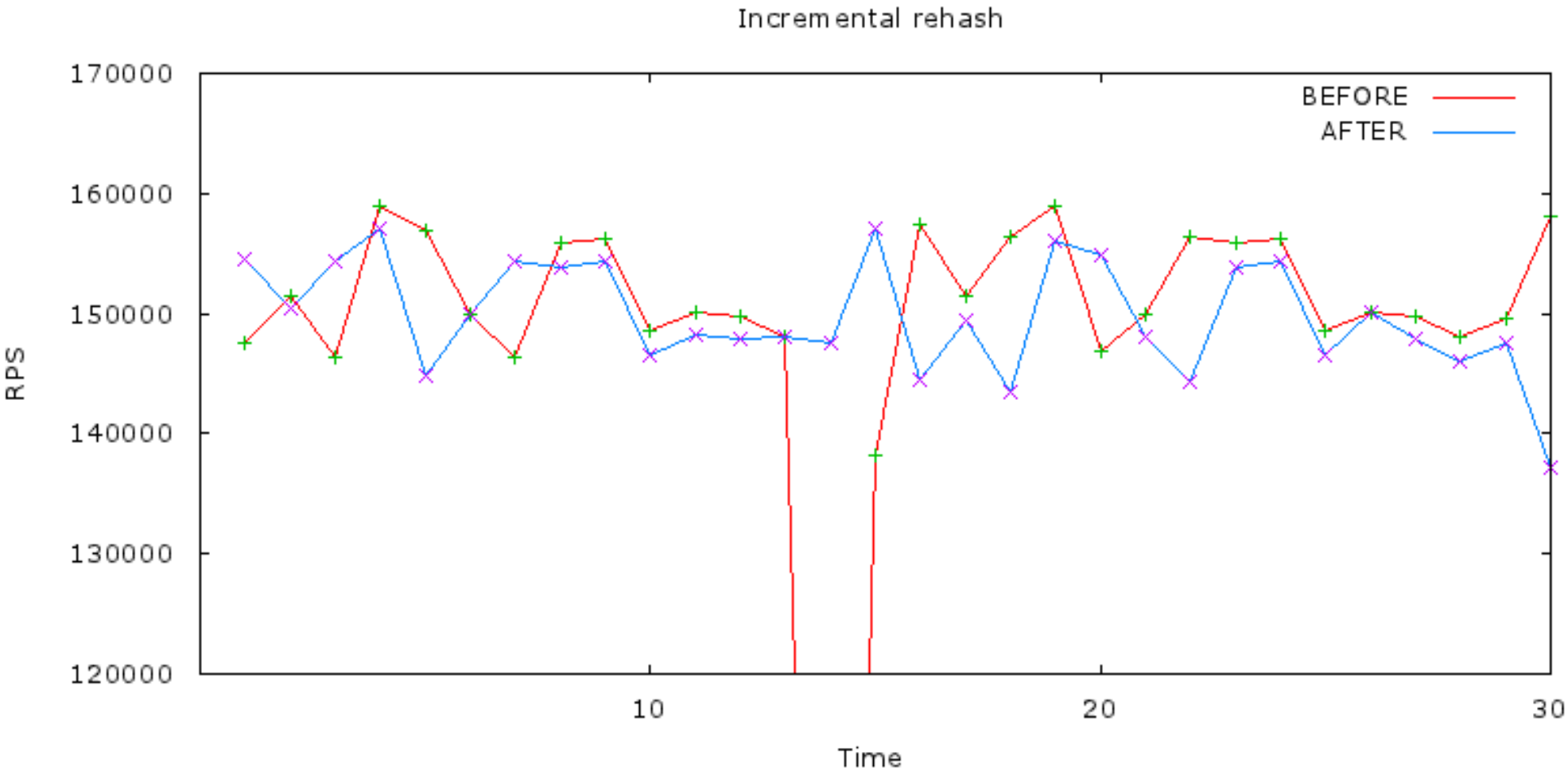
Memory footprint



A simple GET/SET benchmark



Stuff you don't see: inc. rehash



Auto-increment: PHP

```
define(S_USER, 1); // space number USER
define(INC_NO,1); // index number
define(COUNTER,1); // field number

$key = $tnt->inc(SNO, INC_NO, COUNTER,
                 [1, true]);
$tnt->insert(NS_USER, $key, $data);
```

Auto-increment: Lua

```
function box.auto_increment(spaceno, ...)
    max_tuple = box.space[spaceno].index[0].idx:max()
    if max_tuple ~= nil then
        max = box.unpack('i', max_tuple[0])
    else
        max = -1
    end
    return box.insert(spaceno, max + 1, ...)
end
```

```
$tnt->call(SNO, 'box.auto_increment', $data);
```

FIFO pattern: Lua

```
function fifo_push(name, val)
    fifo = find_or_create_fifo(name)
    top = box.unpack('i', fifo[1])
    bottom = box.unpack('i', fifo[2])
    if top == fifomax+2 then -- % size
        top = 3
    …
    end
    return box.update(0, name, '=p=p=p', 1, top,
                      2, bottom, top, val)
end
```

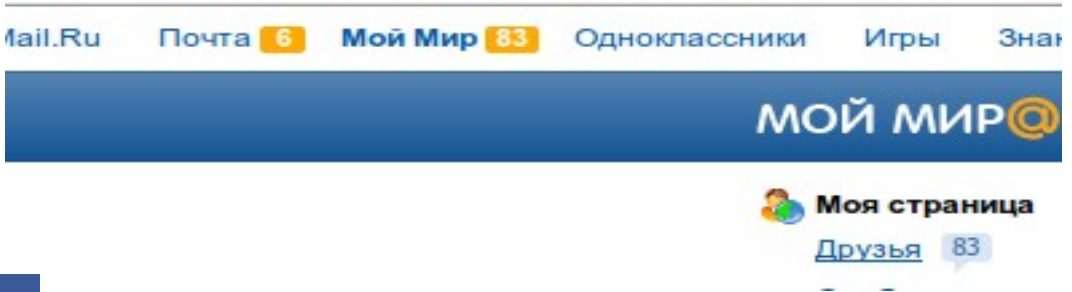
Memcache pattern

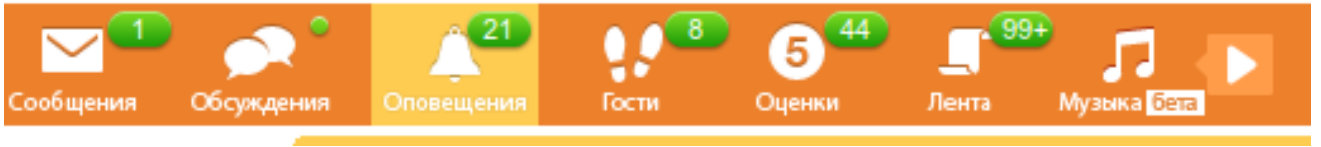
- You can create your own background fibers:
- **`box.fiber.create(), box.fiber.yield()`**
- A background fiber can be a *custom expire process*

- Used to store user sessions in **mail.ru:**
- 4 units, 2 Tarantool/Box instances each, 2 masters and 2 replicas

- 40-60k requests/second, CPU usage < 20%

Notifications pattern





- 6 units, running 4 Tarantool/Box instances each
- **`notification_push()`, `notification_read()`**
- 250 GB of constantly changing data

Notificaitons (2)

- 5 notification types
- a ring buffer for each type stores 20 most recent notifications
- a counter of unread notifications for each type
- duplicate notifications are automatically removed

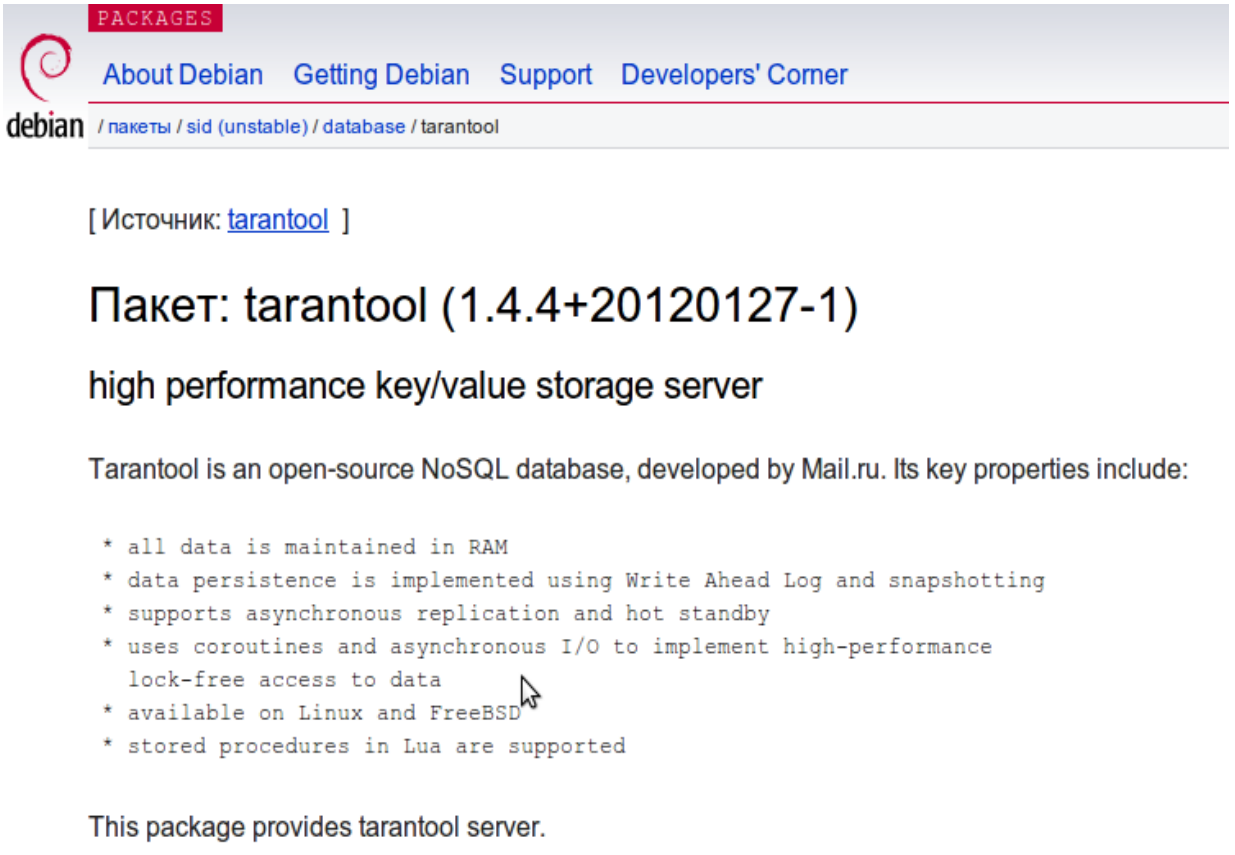
User API:

```
notification_push(user_id, type_id,
        notification)
notification_read(user_id, type_id)
```

Notifications (3)

```
table.insert(args, box.pack('ppp', dup_offset, #id, ""))
local unread_count = box.unpack('i', tuple[fieldno])
if unread_count < ring_max and
       dup_offset > unread_count * #id then
    format = '+p+p'..format
    local args1 = args
    args = { 1, 1, fieldno, 1 }
    for _, v in ipairs(args1) do
        table.insert(args, v)
    end
end
return box.update(space_no, user_id, format,
unpack(args))
```

Tarantool is in Debian “Sid”



PACKAGES

About Debian Getting Debian Support Developers' Corner

debian / пакеты / sid (unstable) / database / tarantool

[Источник: tarantool]

Пакет: tarantool (1.4.4+20120127-1)

high performance key/value storage server

Tarantool is an open-source NoSQL database, developed by Mail.ru. Its key properties include:

* all data is maintained in RAM
* data persistence is implemented using Write Ahead Log and snapshotting
* supports asynchronous replication and hot standby
* uses coroutines and asynchronous I/O to implement high-performance lock-free access to data
* available on Linux and FreeBSD
* stored procedures in Lua are supported

This package provides tarantool server.

Thank you,
Dmitry Oboukhov!

We're looking for
SuSE and **Fedora**
packagers

What's cooking in 1.5

- quick-start
- disk-based backend
- synchronous multi-master replication
- authenticaiton

Thank you!

Links

http://github.com/mailru/tarantool - source code
http://github.com/mailru/tntlua - open source stored procedures repository
http://groups.google.com/group/tarantool - mailing list
http://tarantool.org/dist/ - **always fresh** .tar.gz and .rpm

kostja@tarantool.org

http://tarantool.org