Erlang is Our Superpower

Or, How Collecta Uses Erlang

Jack Moffitt – CTO, Collecta

jack@metajack.im
@metajack
http://metajack.im

Part I: We are **bitten** by a **spider.**

It all **started** as a

game.

XMPP makes a great

platform.

We were **totally** in love with

Python.

We were even **more** in love with

Twisted.

We were **not** in love with our

XMPP server.

We decided to

shop around.

We heard a lot about

ejabberd.

But

WTF

is **Erlang?**

At least it’s

better than

Java.

You can

learn a lot

from

Joe.

In the end, we made the

users happy.

Part II: We meet our

arch-enemy.

Say there's an

earthquake.

Google gives you

plate

techtonics.

But you wanted information about the quake that

just happened.

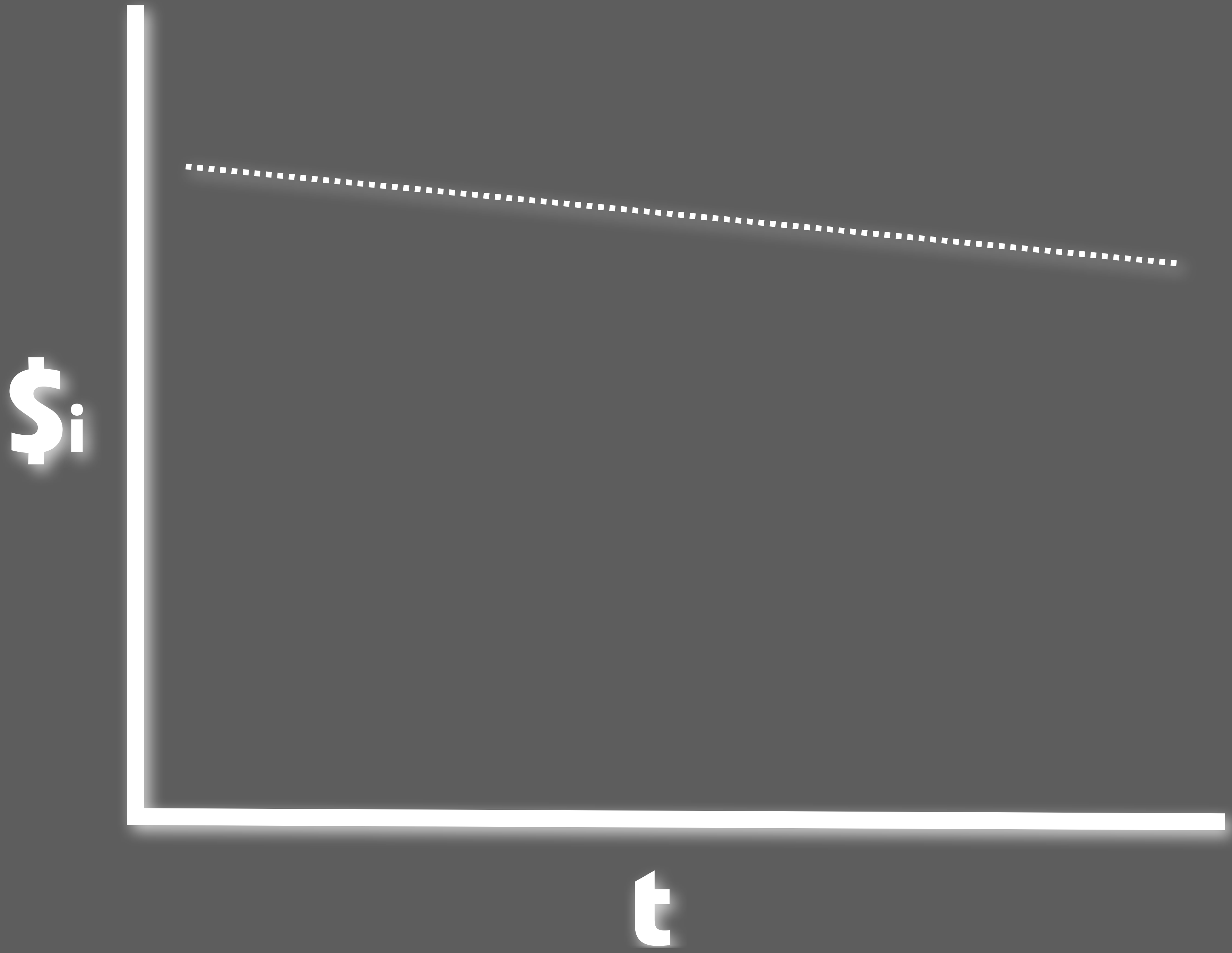
Traditional search

devalues time.

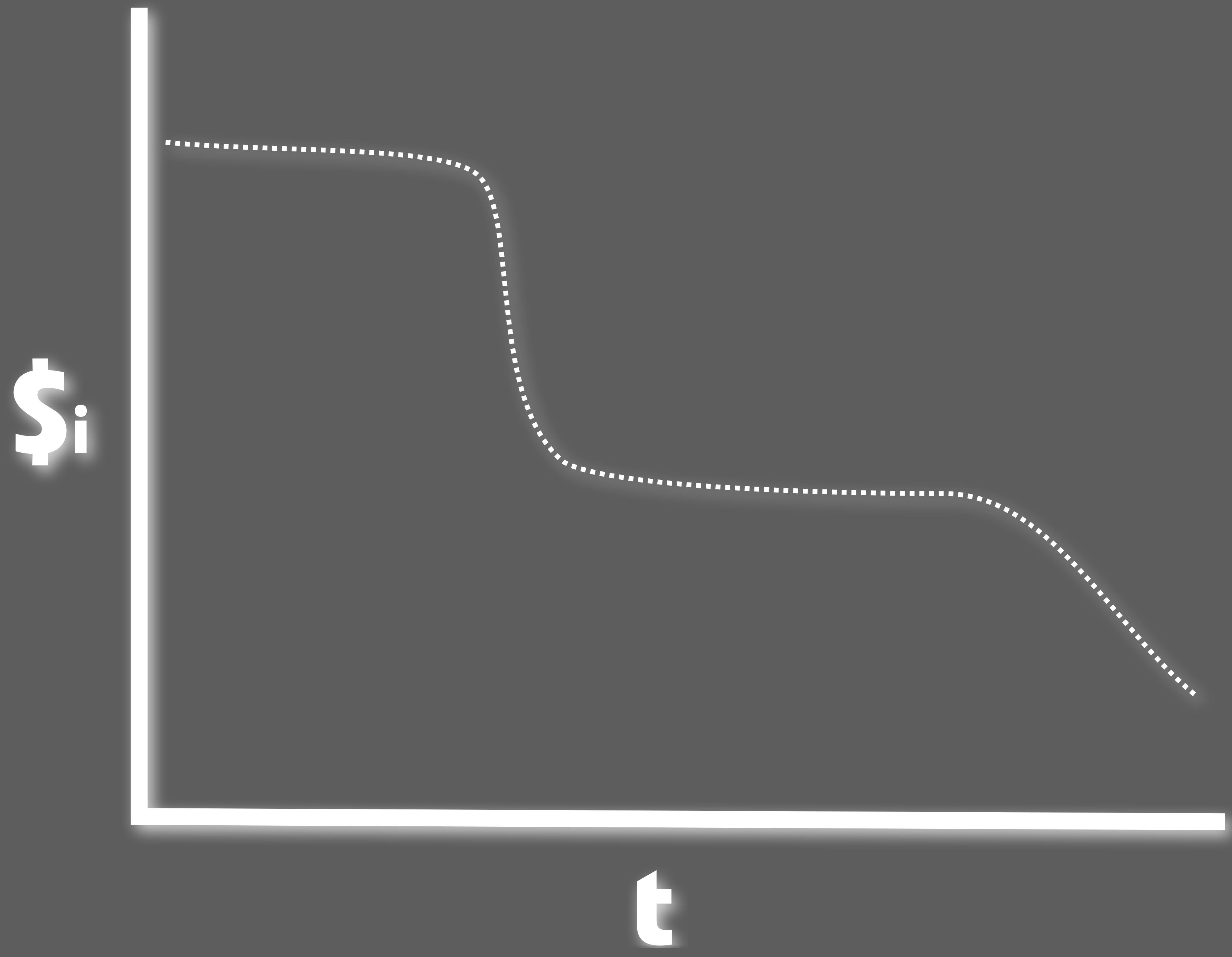
$$\$_i = f(t)$$

Information's **value** is a

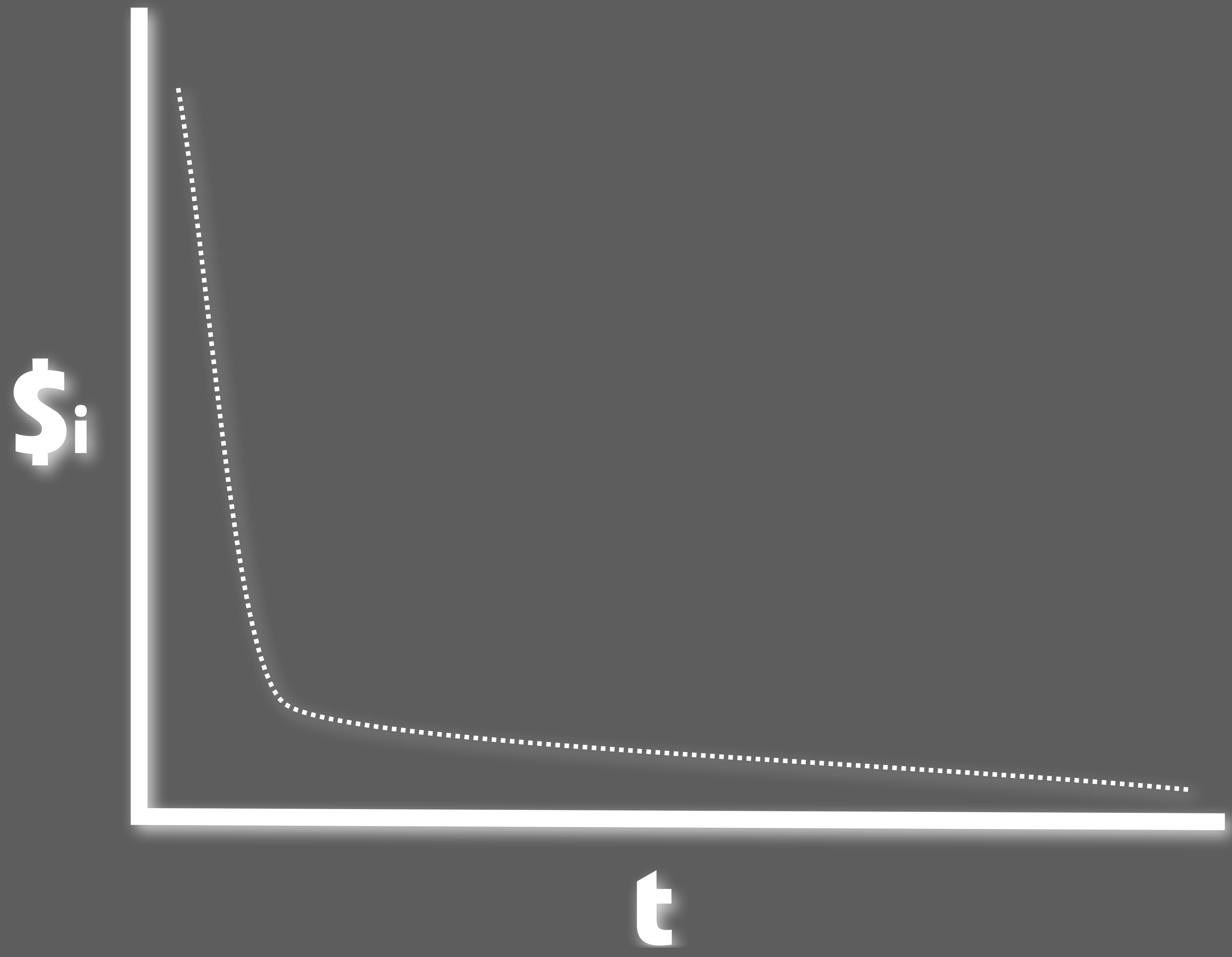
function of time.



Encyclopedia entry



Apple announcement



Customer complaint

Part III: We draw up our

battle plans.

Publishers

push data

to us.

Data is

aggregated

into a single

firehose.

Data is

transformed

and

filtered

as it passes through.

Collecta does
spam filtering,
language annotation and filtering,
classification,
and **keyword filtering.**

Data is

persisted to disk

at the **end of the chain.**

Results are

streamed

to clients.

Part IV: We show off our **costume.**

Part V: We recognize our

new power.

We knew the

shape

of the **problem.**

Quantity of data is

large

and

growing rapidly.

We must plan for user demand to be

massive.

The

scale

mandates

distributed

solutions.

I’ve heard of a

language

that could help.

We went from

Erlang WTF

to

Erlang FTW!

Part VI: Our battle **unfolds.**

ejabberd

It forms our

distributed messaging

pipeline.

Rich semantics

are available via

XMPP and Pubsub.

It is

Extremely

and

easily

extensible.

You can extend it at the

protocol layer with **XMPP**

and in

code with **modules**.

It’s very easy to

filter and modify

packets.

It contains very good

built-in support

for

bi-directional Web.

Webmachine

Webmachine is used for

HTTP push

intake.

It serves the

web site.

It forms the

API glue.

We use it for the

client-side analytics

gathering.

CouchDB

CouchDB provides

raw data storage.

We also store

API keys

and

user accounts.

We fetch documents

by key.

RabbitMQ

RabbitMQ handles the final

persistance chain.

It manages an

async but ordered

set of operations.

Riak & Riak Search

Riak is

linearly scalable.

There is **no** app-level sharding.

Solr and Lucene

don’t scale.

Part VII: We come to the **happy ending.**

Built-in
distribution and **message passing**
is a
huge win.

Code is

concise.

Development is

rapid.

Erlang applications tend to be

best in class.

Hot

code upgrades

are easy;

you'll **actually do them.**

Rich set of

libraries

for building

distributed and scalable

systems.

It actually

works.

Haiti needed

help.

MySpace launched

***our* new product.**

Woke up to

3x traffic.

Started to worry at

5x.

Really worried at

8x.

Then the concert

started.

It fell down at

12x.

One

bug fix needed.

Now our normal traffic is

40x.

We’re still

learning.

Most common issue is

overflowing mailboxes.

The End

http://professionalxmpp.com
http://metajack.im
jack@metajack.im
@metajack