Running a 24x7 system at Kreditor

Architecture and Experiences

What is Kreditor?

Kreditor = creative billing solutions

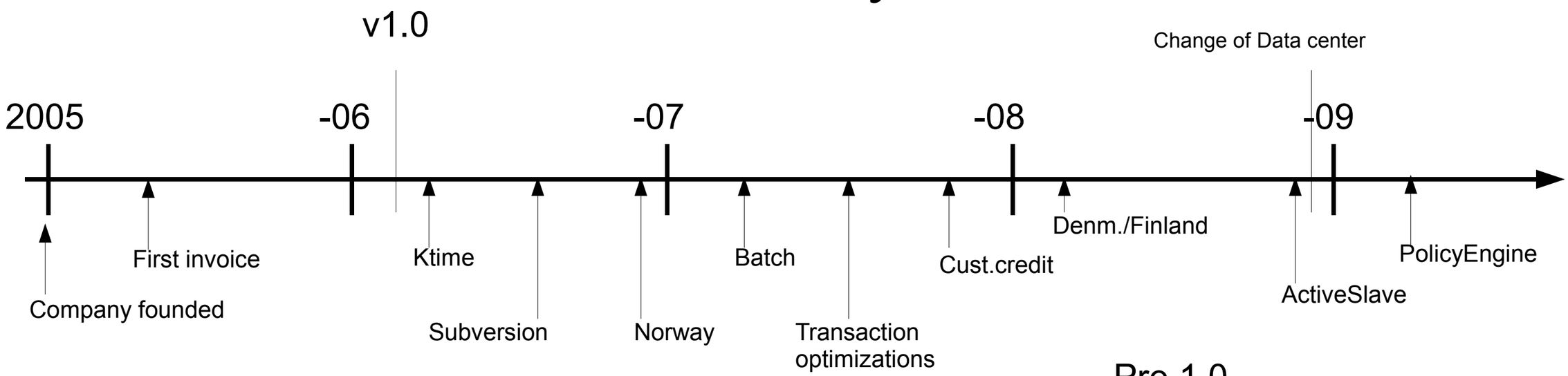
- Offer your customers to pay by bill or installment without taking any risks or increased administration for your company.
- Customer is offered the convenience of payment after delivery.
- Easier and safer compared to credit card payments.

Also:

Kreditor is Sweden's fastest growing company

Kreditor is a 100% Erlang shop.

History



- Company started January 2005
- First invoice went through 10 April 2005
- SW by 4 former Bluetail guys.
- Extreme timeschedule => Straightforward arch.
- Amount of work predicted to be small (!?...)
- SW guys leaves and forms Tail-f.
- Kreditor hires 2 SW devs. in end of 2005
- Tail-f gives support beginning of 2006
- Switch to Subversion (after disk-crash...)

KREDITOR

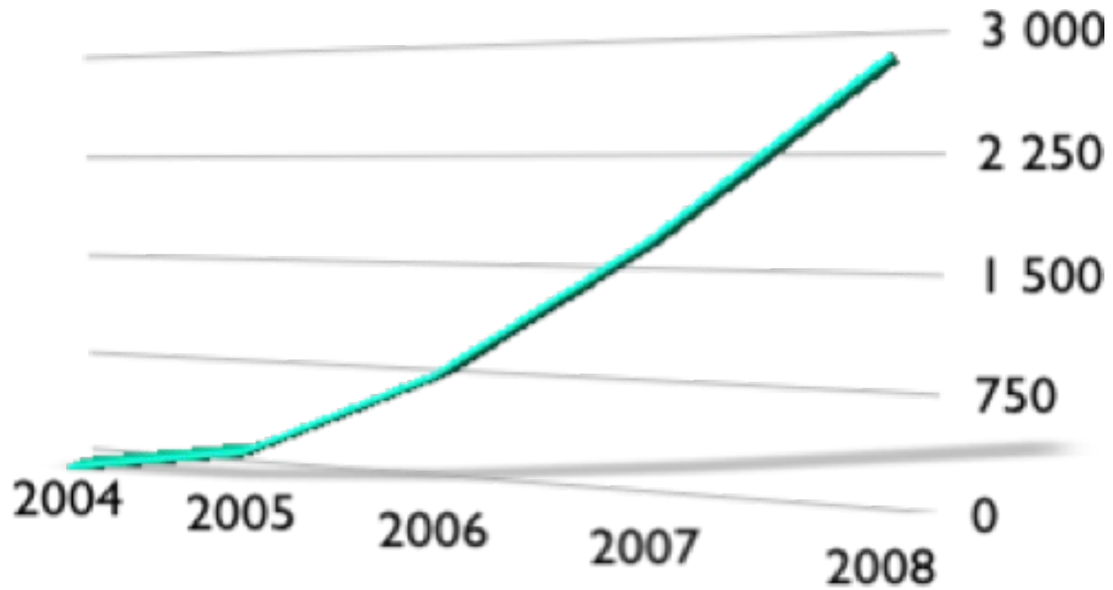
<u>Pre 1.0</u>
- Basic invoicing
- Sweden only

<u>Post 1.0</u>
- Ktime (virtual time)
- Enforcement
- Other countries
- Customer credit
- Batch API
- Test framework
- Master/ActiveSlave
- Payment plans
- Bookkeeping 2 Comp.
- SMS gateway
- ....lots of stuff.....

Historical facts

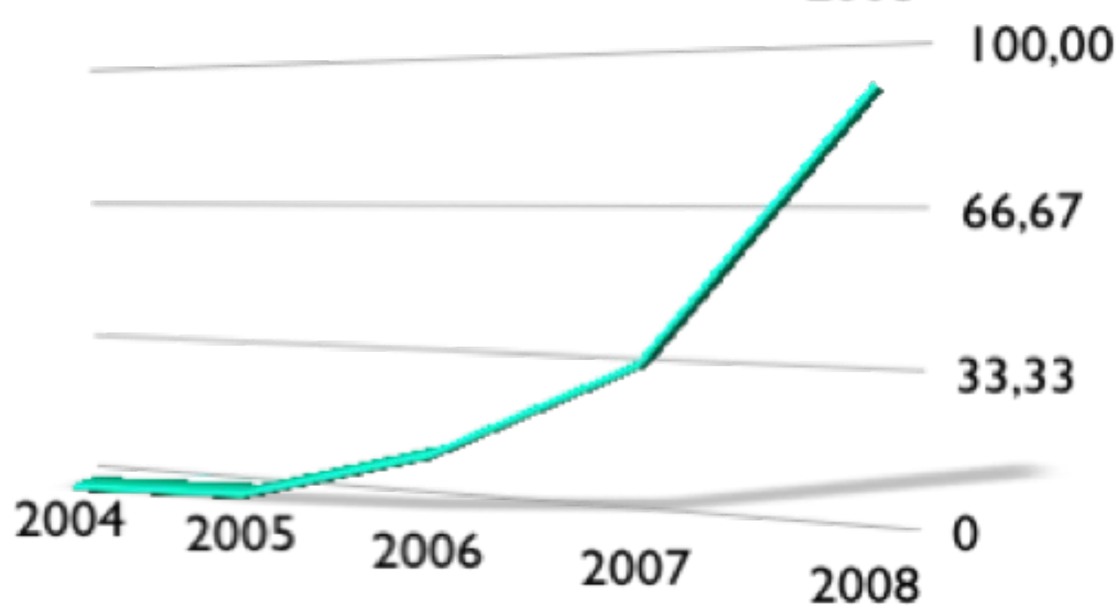
Number of connected stores:

2004: 0
2005: ~200
2006: ~800
2007: ~1700
2008: ~2800



Turnover:

2004: ~0 SEK
2005: 1.5 million SEK
2006: 13.5 million SEK
2007: 35 million SEK
2008: 90.8 million SEK

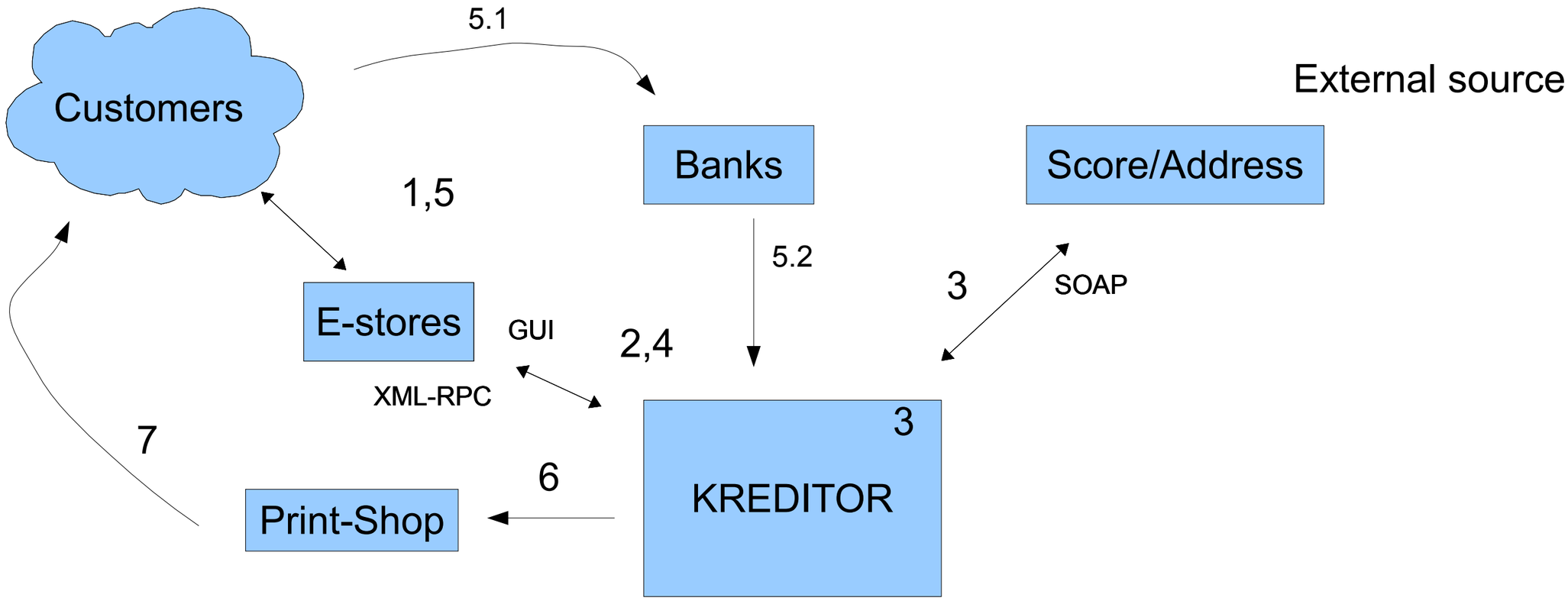


Current staff: ~100

Typical use cases

KREDITOR

Use case continue



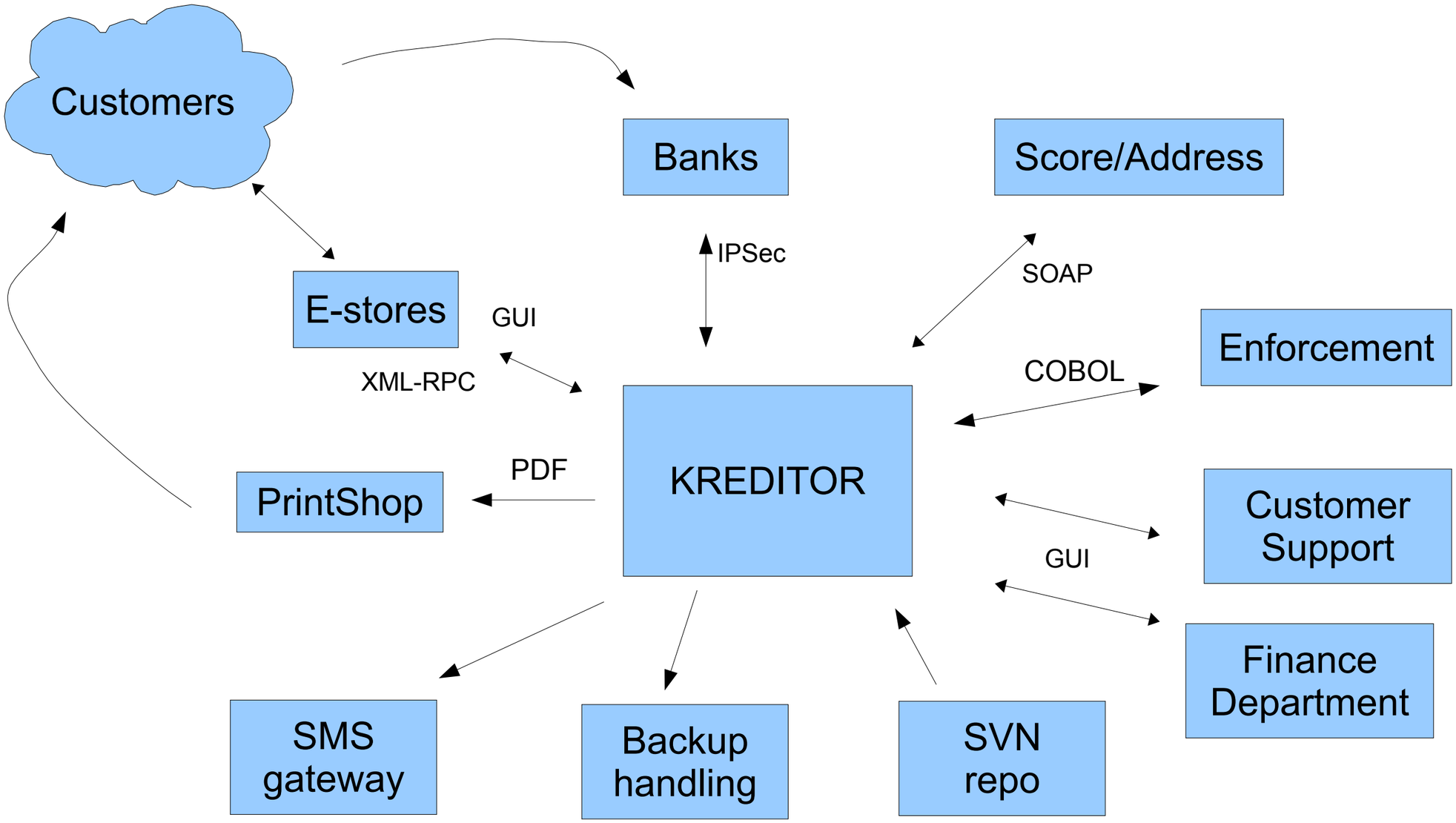
1. Customer buys perfume
2. E-store places order
3. Kreditor verifies that customer is OK

IMMEDIATELY

4. E-store activates/modifies order
5. E-store ships goods + invoice (PDF). (5.1 + 5.2 Customer pay bill)
6. Kreditor (possibly) sends reminder (PDF) to be printed.
7. Print-Shop prints and mail the reminder to Customer

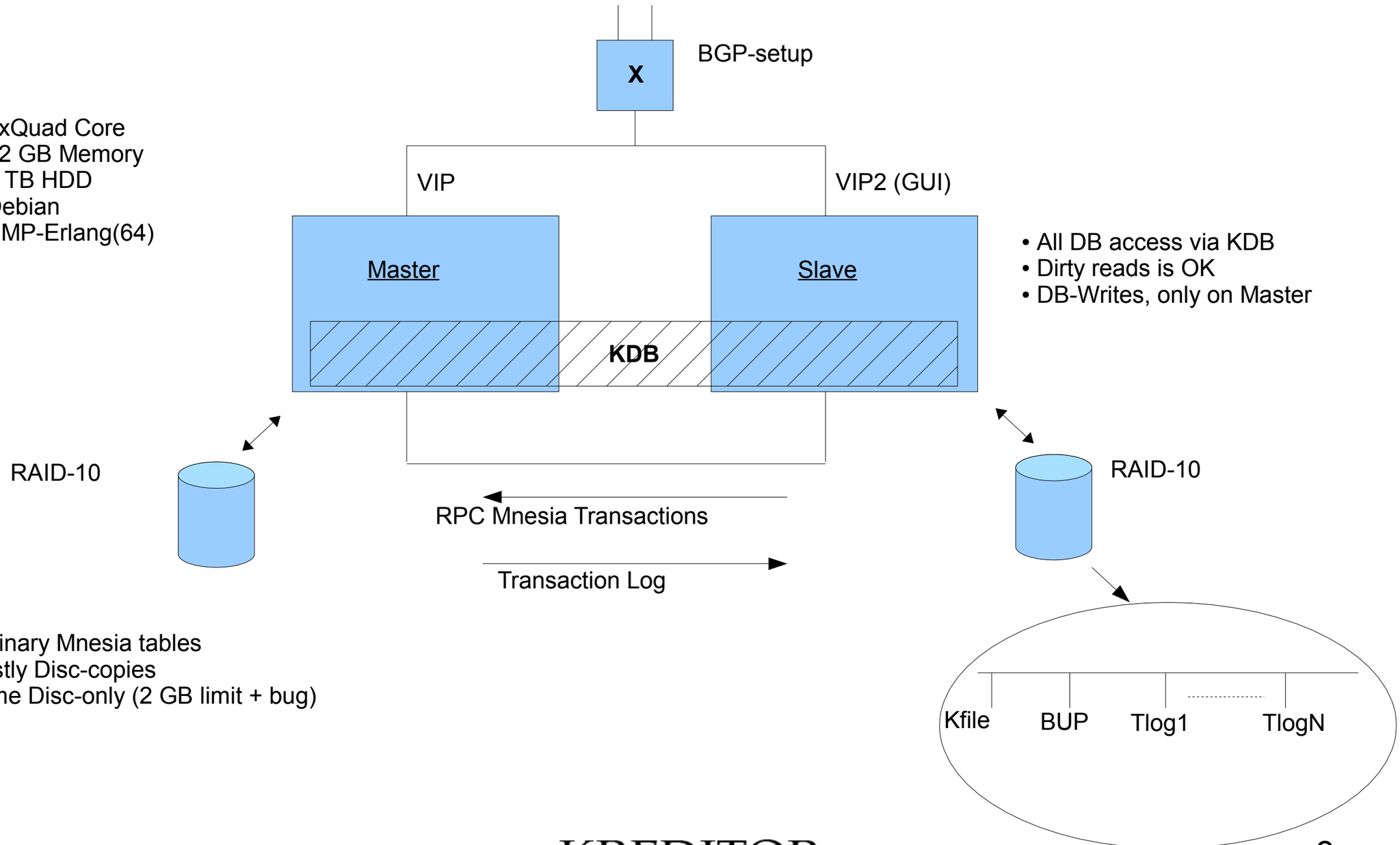
LATER

External architecture



Internal Architecture

- 2xQuad Core
- 32 GB Memory
- 1 TB HDD
- Debian
- SMP-Erlang(64)



- All DB access via KDB
- Dirty reads is OK
- DB-Writes, only on Master

- Ordinary Mnesia tables
- Mostly Disc-copies
- Some Disc-only (2 GB limit + bug)

The KDB layer

- All Mnesia operations has to go through KDB
- Encapsulates *mnesia:transaction/1* with:
  - Transaction timeouts
  - Execution on Master (via Erlang RPC)
  - Transaction Log handling
- Dirty reads OK (still via KDB though)

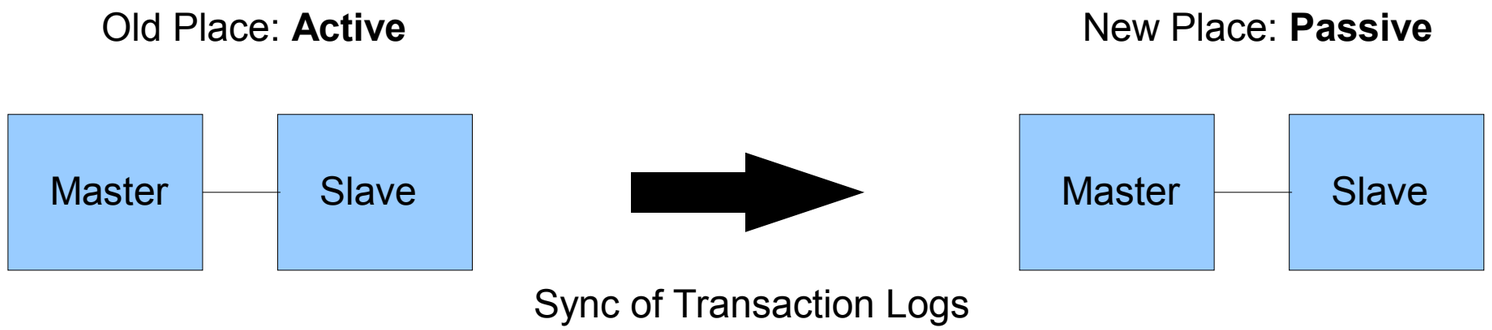
Major components

- Invoice handling (xml-rpc, batch, GUI)
- Installment plans (detailed invoice to customer, interest calculations)
- Credit granting and Address lookup.
- Cronjobs (bookeeping,print shop,bank files,settlements,enforcement,...)
- Bookeeping (both internal and for E-Stores)
- KDB layer (transaction logs, daily backup, failure recovery)
- Kcases (handling of erroneous payments, etc...)
- GUI (65000 loc, Erlang+ehtml)
- Alarm and Log handling

Slave takeover (simplified)

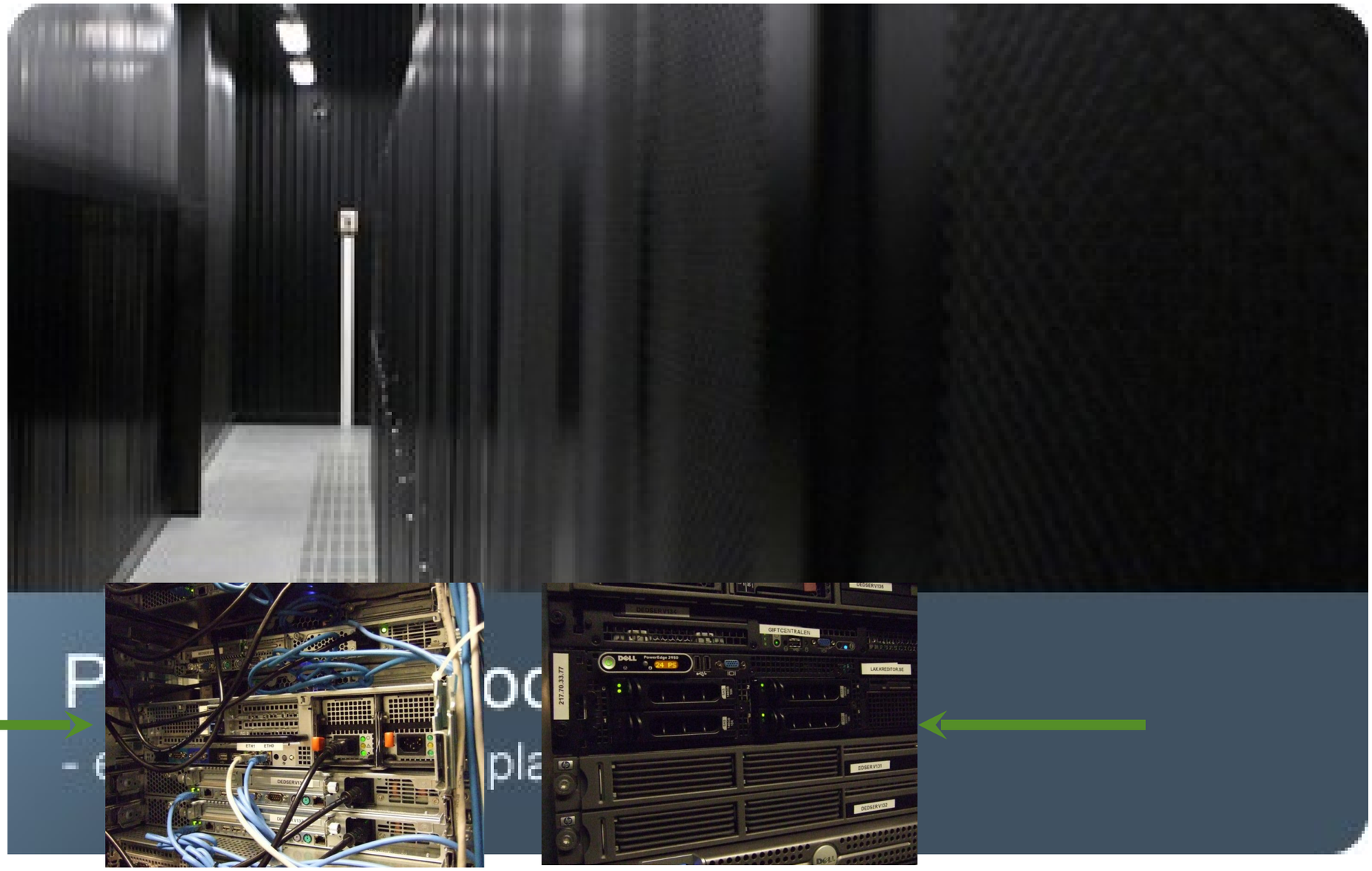
- Master: NodeDown => Slave
- Slave: ping(DefGW)=true AND ping(MasterVIP)=false
- Slave:
  - Setup VIP
  - Start Racoon (IPSec key server)
  - Do GARP (announcing where VIP is)
  - Setup auto-downer (of VIP)
- Slave: Start some Master specific apps.
- Takeover takes around 1 sec.

Change of Data Center



- A new *Cluster Type* was introduced
- Very short TTL in DNS
- Continuous sync of Trans.Log
- Now:
  - Stop active cluster
  - Changed Passive to Active
  - Change IP's in DNS
  - Redirect (iptables) old IP's to new IP's (to handle cached DNS)
- Downtime: the above took 36 seconds

The new shiny Data Center



Problems experienced

- Dets (2GB limit, lurking bug)
- Mnesia transactions (now with timeout)
- Erlang code in Yaws pages (not good for SW upgrades)
- Performance (THG for SMP-Erlang-64-bits)
- Architecture == *“Those parts that are hard to change”*
- Need to do more SW dev. to grow, but can't hire until grown a bit more...

Success experiences

- Robust: Sys. works even with Mnesia corrupting tables
- Upgrade of SW: many times every day
- Change of HW: due to Master-Slave architecture
- Change of data center: with 36 seconds down-time
- Upgrade to 64-bit Erlang: 4GB -> 32GB memory
- Upgrade to SMP: 1 -> 8 active cores, with no SW change
- Active Slave: 8 -> 16 active cores, with only minor SW changes

Ongoing architectural change

- Break out the part dealing with granting of customer credits.
  - Before: Inconsistent, Spaghetti-style
  - After: PolicyEngine, PolicyRules (DSL), No transactions

- Break out address handling.
  - Before: Inconsistent, Address-record littering
  - After: Tracability, Coherent interface, Possible to analyze

- Re-write interfaces toward external score and address sources.
  - Before: Inconsistent, Inflexible
  - After: Common framework, Flexible

How we do SW-development

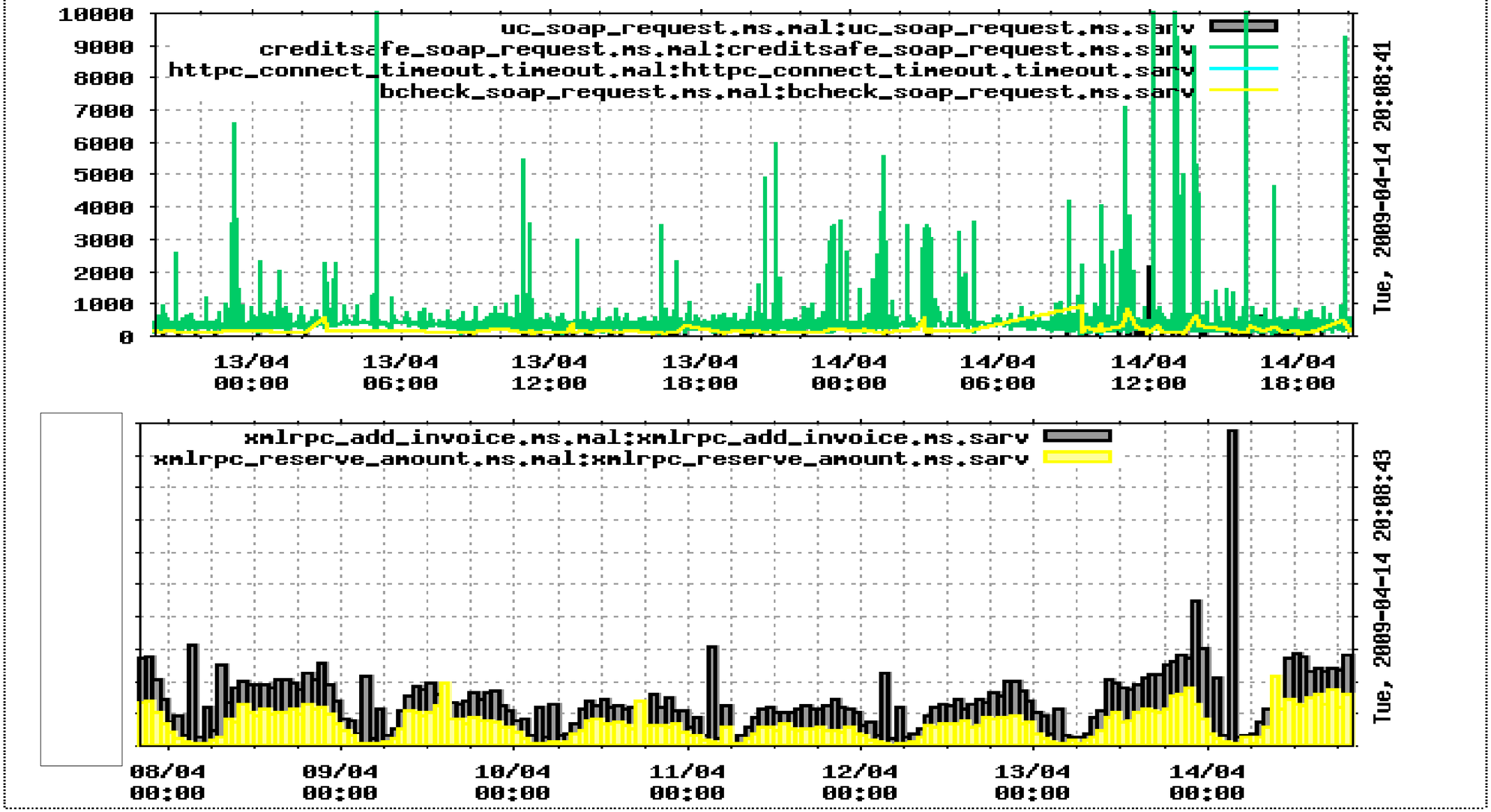
- Scrum style
- Subversion, one production branch
- Yatsy test server, ~700 test cases
- CruiseControl, build+test at commit
- Trunk => ProdBranch ==> svn update on live
- Upgrade instructions in README (going towards prog. upgrades)
- One developer is always on a 24h call (serious alarms via SMS)

CruiseControl

Project	Status *(since)*	Last failure	Last successful	Label	
trunk	*waiting (14.04.09 20:05)*		14.04.09 17:30	build.609	Build
CDON-light	*waiting (14.04.09 20:06)*	**14.04.09 18:54**	14.04.09 13:43	build.133	Build
CDON	*waiting (14.04.09 19:41)*	**14.04.09 19:23**	14.04.09 13:34	build.2	Build
kred-7-0	*waiting (14.04.09 20:05)*		14.04.09 11:48	build.296	Build
trunk-master_slave	*waiting (14.04.09 19:56)*	**14.04.09 17:55**	09.04.09 18:28	build.384	Build
alpha	*waiting (14.04.09 20:05)*		09.04.09 09:10	build.19	Build
CDON-tobias-micke	*waiting (14.04.09 20:06)*		08.04.09 13:41	build.3	Build
kred-payment-remake	*waiting (14.04.09 20:05)*	**09.04.09 09:37**	06.04.09 09:49	build.1	Build
trunk-nightly	*waiting (14.04.09 20:01)*	**14.04.09 05:54**	04.04.09 03:02	build.171	Build
kred-7-0-nightly	*waiting (14.04.09 20:01)*	**14.04.09 03:24**			Build
beta	*waiting (14.04.09 20:05)*				Build
devel-pacc-files	*waiting (14.04.09 20:05)*				Build
devel-fraud-return	*waiting (14.04.09 20:05)*				Build
CDON-nightly	*waiting (14.04.09 20:01)*	**14.04.09 01:13**			Build
devel-p85-2	*waiting (14.04.09 20:05)*				Build

RSS

Statistics


uc_soap_request.ms.mal:uc_soap_request.ms.sarv
creditsafe_soap_request.ms.mal:creditsafe_soap_request.ms.sarv
httpc_connect_timeout.timeout.mal:httpc_connect_timeout.timeout.sarv
bcheck_soap_request.ms.mal:bcheck_soap_request.ms.sarv
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
13/04 00:00
13/04 06:00
13/04 12:00
13/04 18:00
14/04 00:00
14/04 06:00
14/04 12:00
14/04 18:00
Tue, 2009-04-14 20:08:41
xmlrpc_add_invoice.ms.mal:xmlrpc_add_invoice.ms.sarv
xmlrpc_reserve_amount.ms.mal:xmlrpc_reserve_amount.ms.sarv
08/04 00:00
09/04 00:00
10/04 00:00
11/04 00:00
12/04 00:00
13/04 00:00
14/04 00:00
Tue, 2009-04-14 20:08:43

Open Source Usage

- XML-RPC library from jungerl
- Erlguten (for PDF generation of invoices)
- Gettext from jungerl (i18n)
- Erlsom (xml schema, sax parser, for the Batch API)
- Egeoip (geolocation from Google code)
- Eper (redbug tracing and stats collecting)
- Yatsy testserver (Google code)
- Yaws
- Erlang/OTP :-)

Overall Experiences

- Initial architecture has served well.
- Have been possible to evolve architecture
- Architecture will need to scale further in the future
- Obstacles: get away from transactions, break out major components
- Perhaps need to choose the CAP path...
  - Consistency (eventual)
  - Availability
  - Partitioning

Erlang is amazingly good for running a 24/7 system!

Future

- Kreditor will become a proper bank.
- Opens up for some exciting applications.
- Interesting architectural challenges ahead.
- More ??????????
- We are hiring (?)