MetaCentrum

Harvesting Logs and Events Using MetaCentrum Virtualization Services

Radoslav Bodó, Daniel Kouřil
CESNET

Campus network monitoring and security workshop
Prague 2014

Agenda

- Introduction
- Collecting logs
- Log Processing
- Advanced analysis
- Resume

Introduction

- Status
  - NGI MetaCentrum.cz
    - approx. 750 worker nodes
    - web servers
    - support services

- Motivation
  - central logging services for
    - security
    - operations

Goals

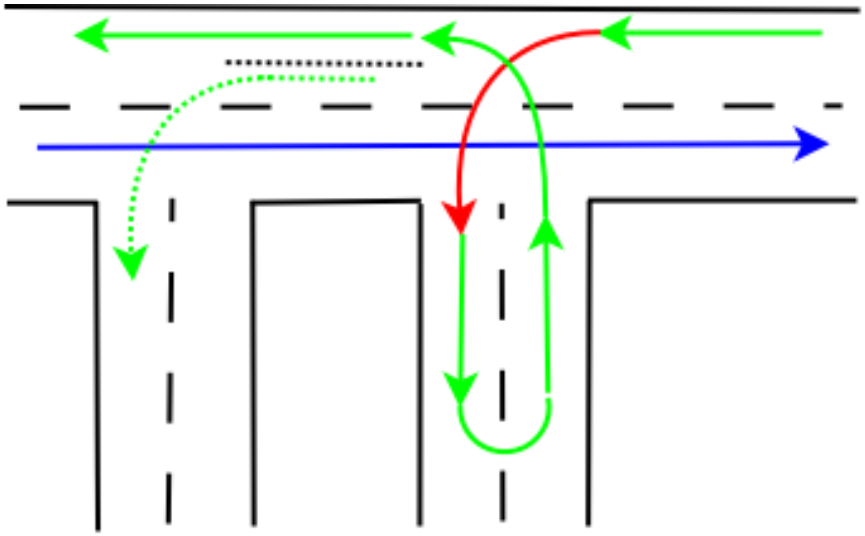
- secure and reliable delivery
  - encrypted, authenticated channel
- scalability
  - system handling lots of logs on demand
  - scaling up, scaling down
- flexibility
  - system which can handle "any" data ...

Collecting logs

- linux + logging = syslog
  - forwarding logs with syslog protocol
    - UDP, TCP, RELP
    - TLS, GSS-API

- NGI Metacentrum
  - Debian environment
  - Kerberized environment
    - rsyslogd forwarding logs over GSS-API protected channel

rsyslogd shipper



- **omgssapi.so -- client**
  - forwarding is action
    - action queue must be non direct
    - queue must be limited
    - full queue must not block main queue

```
$ActionQueueType LinkedList          # use asynchronous processing
$ActionQueueFileName srvrfwd1        # set file name, also enables disk mode
$ActionResumeRetryCount -1           # infinite retries on insert failure
$ActionQueueSaveOnShutdown on        # save in-memory data if rsyslog shuts down
$ActionQueueMaxDiskSpace 100m        # limit disk cache
$ActionQueueTimeoutEnqueue 100       # dont block worker indefinitely when cache fills up
*.* :omgssapi:<server_name>:<port>   # deliver all messages to central server using GSS-API protection
```

rsyslogd server

- **imgssapi.so -- server**
  - nothing really special
    - listener
    - per IP layout
    - service logs

```
$ModLoad imgssapi
$InputGSSServerServiceName host
$InputGSSServerPermitPlainTCP off
$InputGSSServerRun 515
$InputGSSServerMaxSessions 2000
```

```
$template PerHostLogsSyslog,"/var/log/hosts/%$YEAR%/%$MONTH%/%fromhost-ip%/syslog"
$template PerHostLogsAuthlog,"/var/log/hosts/%$YEAR%/%$MONTH%/%fromhost-ip%/auth.log"
$template PerHostLogsKernlog,"/var/log/hosts/%$YEAR%/%$MONTH%/%fromhost-ip%/kern.log"

auth.*,authpriv.* -?PerHostLogsAuthlog
kern.* -?PerHostLogsKernlog
*.*;kern,auth,authpriv.none -?PerHostLogsSyslog

$template PerServiceLogsSyslog,"/var/log/hosts/auth/%$YEAR%/%$MONTH%/auth.log.%$YEAR%%$MONTH%%$DAY%"
auth.*,authpriv.* -?PerServiceLogsSyslog

$template PbsService,"/var/log/hosts/pbs/%$YEAR%/%$MONTH%/log.%$YEAR%%$MONTH%%$DAY%"
:programname, contains, "pbs_mom" -?PbsService
```

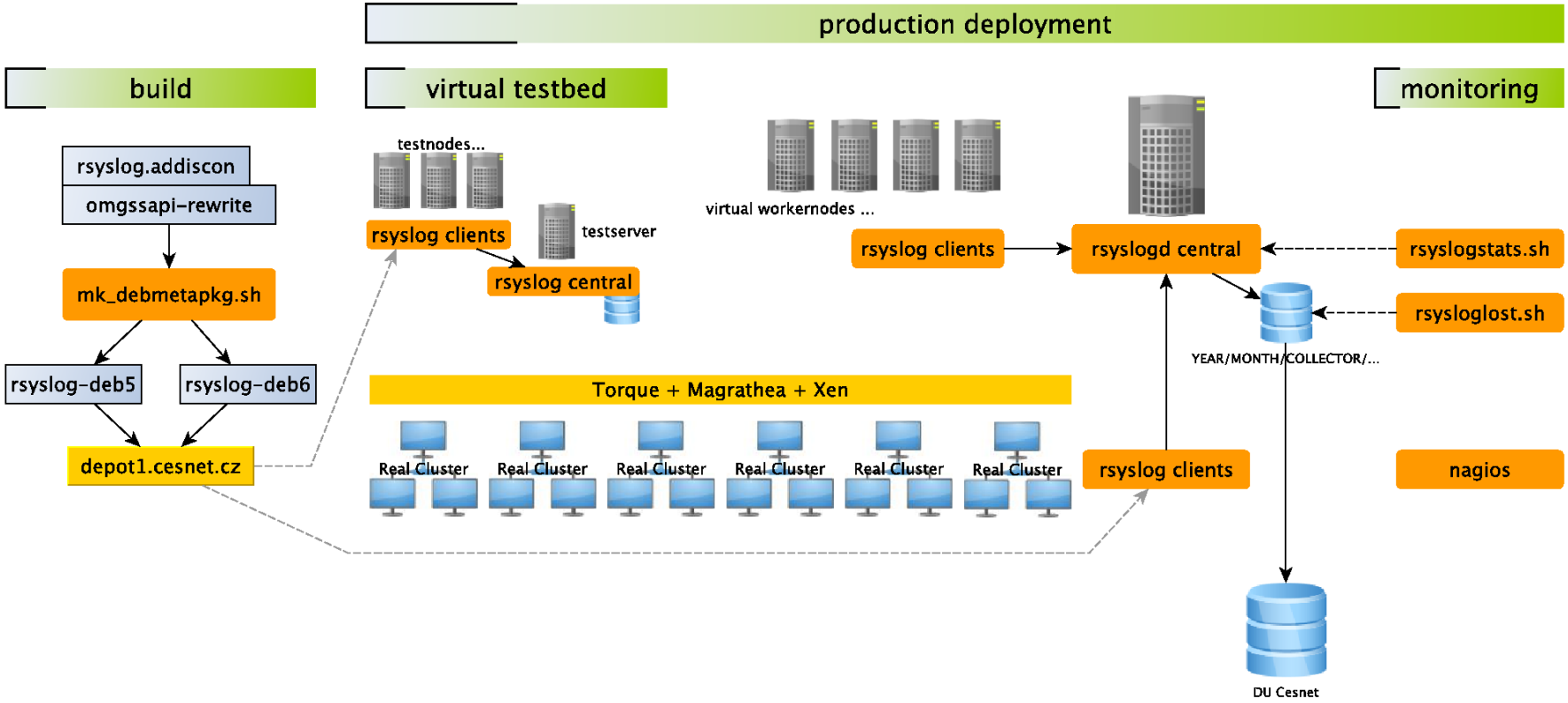
rsyslogd GSS patches

- original GSS-API plugins are not maintained since 3.x
  - plugin does not reflect internal changes in rsyslogd >> occasional segfaults/asserts
    - not quite nice even after upstream hotfix
      - no more segfaults, but SYN storms (v5,v6,?v7,?v8)

- a new omgssapi based on
  - old one + actual omfwd (tcp forward)
  - contributed to public domain but not merged yet
    - we'll try to push it again into v8

rsyslogd testbed

- development of multithreaded application working with strings and networking is error prone process .. everytime
  - virtual testbed used to test produced builds



rsyslogd wrapup

- in production about a 2 years
- approx. 90% nodes coverage (700 nodes)
- 50 - 100GB per month
  - 2GB compressed with 7zip
- monitoring
  - nagios
  - cron scripts

Log processing

- why centralized logging ?
  - having logs on single place allows us to do centralized do_magic_here

- classic approach
  - grep, perl, cron, tail -f

Log processing

- classic approach
  - grep, perl, cron, tail -f
  - alerting from PBS logs
    - jobs_too_long

```
# du -sh .
105G .
# time grep -R "realuser" * > search.txt
real 18m39.447s
user 1m7.796s
sys 1m24.565s
# wc search.txt
   81636  1773549 21777400 search.txt
```

- perl is fine but not quite fast for 100GB of data
  - example:
    - search for login from evil IPs

- for analytics a database must be used
  - but planning first ...

The size

- the grid scales
  - logs growing more and more
    - a scaling DB must be used

- clustering, partitioning
  - MySQL, PostgreSQL, ...

The structure strikes back

- logs are not just text lines, but rather a nested structure

```
LOG ::= TIMESTAMP DATA
DATA ::= LOGSOURCE PROGRAM PID MESSAGE
MESSAGE ::= M1 | M2
```

- logs differ a lot between products
  - kernel, mta, httpd, ssh, kdc, ...

- and that does not play well with RDBMS (with fixed data structures)

A new hope ?

- NoSQL databases
  - emerging technology
  - cloud technology
  - scaling technology
  - c00l technology

- focused on
  - ElasticSearch
  - MongoDB

elasticsearch.

- ElasticSearch is a full-text search engine
  built on the top of the Lucene library

  - it is meant to be distributed
    - autodiscovery
    - automatic sharding/partitioning,
    - dynamic replica (re)allocation,

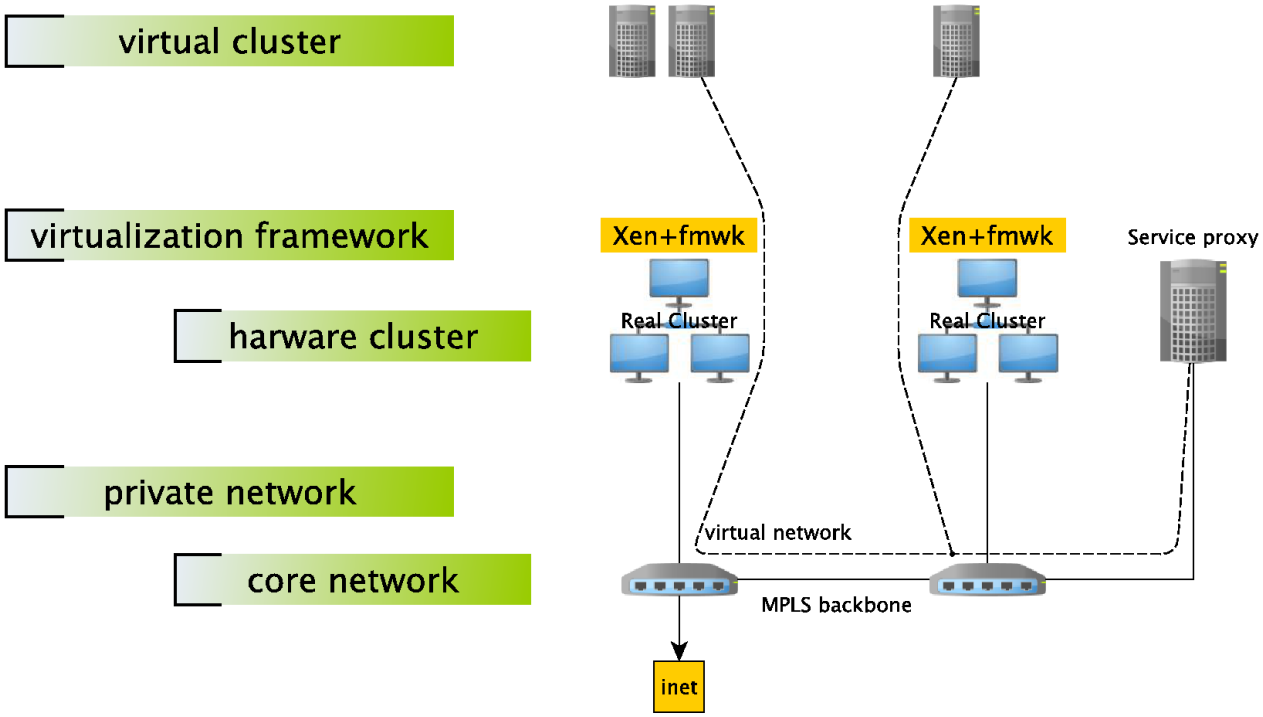
    - various clients already

elasticsearch.

- REST or native protocol
  - PUT indexname&data (json documents)
  - GET _search?DSL_query...
    - index will speed up the query
- ElasticSearch is not meant to be facing public world
  - no authentication
  - no encryption
  - no problem !!

~~rsyslog testbed~~ Private cloud

- a private cloud has to be created in the grid
  - cluster members are created as jobs
  - cluster is interconnected by private VLAN
  - proxy is handling traffic in and out



Turning logs into structures

- rsyslogd
  - omelasticsearch, ommongodb

LOG ::= TIMESTAMP DATA

DATA ::= LOGSOURCE PROGRAM PID MESSAGE

MESSAGE ::= **M1** | **M2** | **...**

- Logstash
  - grok
  - flexible architecture

logstash -- libgrok

- reusable regular expressions language and parsing library by Jordan Sissel

```
Nov  1 21:14:23 scorn kernel: pid 84558 (expect), uid 30206: exited on signal 3
```

In order, your brain reads a timestamp, a hostname, a process or other identifying name, a number, a program name, a uid, and an exit message. You might represent this in words as:

```
TIMESTAMP HOST PROGRAM: pid NUMBER (PROGRAM), uid NUMBER: exited on signal NUMBER
```

All of these can be represented by regular expressions. Grok comes with a bunch of pre-defined patterns to make getting started easier, including syslog patterns that help with the above. In grok, this pattern looks like:

```
%{SYSLOGBASE} pid %{NUMBER:pid} \(%{WORD:program}\), uid %{NUMBER:uid}: exited on signal %{NUMBER:signal}
```

All of the base grok patterns are in uppercase for style consistency. Each thing in %{} is evaluated and replaced with the regular expression it represents.

Grokked syslog


```
{
    _index: "logstash-2013.01.13",
    _type: "syslog",
    _id: "jTo_ymGdSaWluwom332bvw",
    _version: 1,
    _score: 1,
      _source: {
          @tags: [ ],
          @fields: {
              coll: [
                "160.217.209.82"
           ],
             logsource: [
                "hildor23-1.prf.jcu.cz"
           ],
             program: [
                "CRON"
           ],
             pid: [
                "3849"
           ],
             message: [
                "pam_unix(cron:session): session closed for user root"
           ]
        },
        @timestamp: "2013-01-13T23:20:01.000Z",
        @type: "syslog"
    }
}
```

logstash -- arch

- event processing pipeline
  - input | filter | output
- many IO plugins
- flexible ...

inputs

- amqp
- drupal_dblog
- eventlog
- exec
- file
- ganglia
- gelf
- gemfire
- generator
- heroku
- irc
- log4j
- lumberjack
- pipe
- redis
- relp
- sqs
- stdin
- stomp
- syslog
- tcp
- twitter
- udp
- xmpp
- zenoss
- zeromq

filters

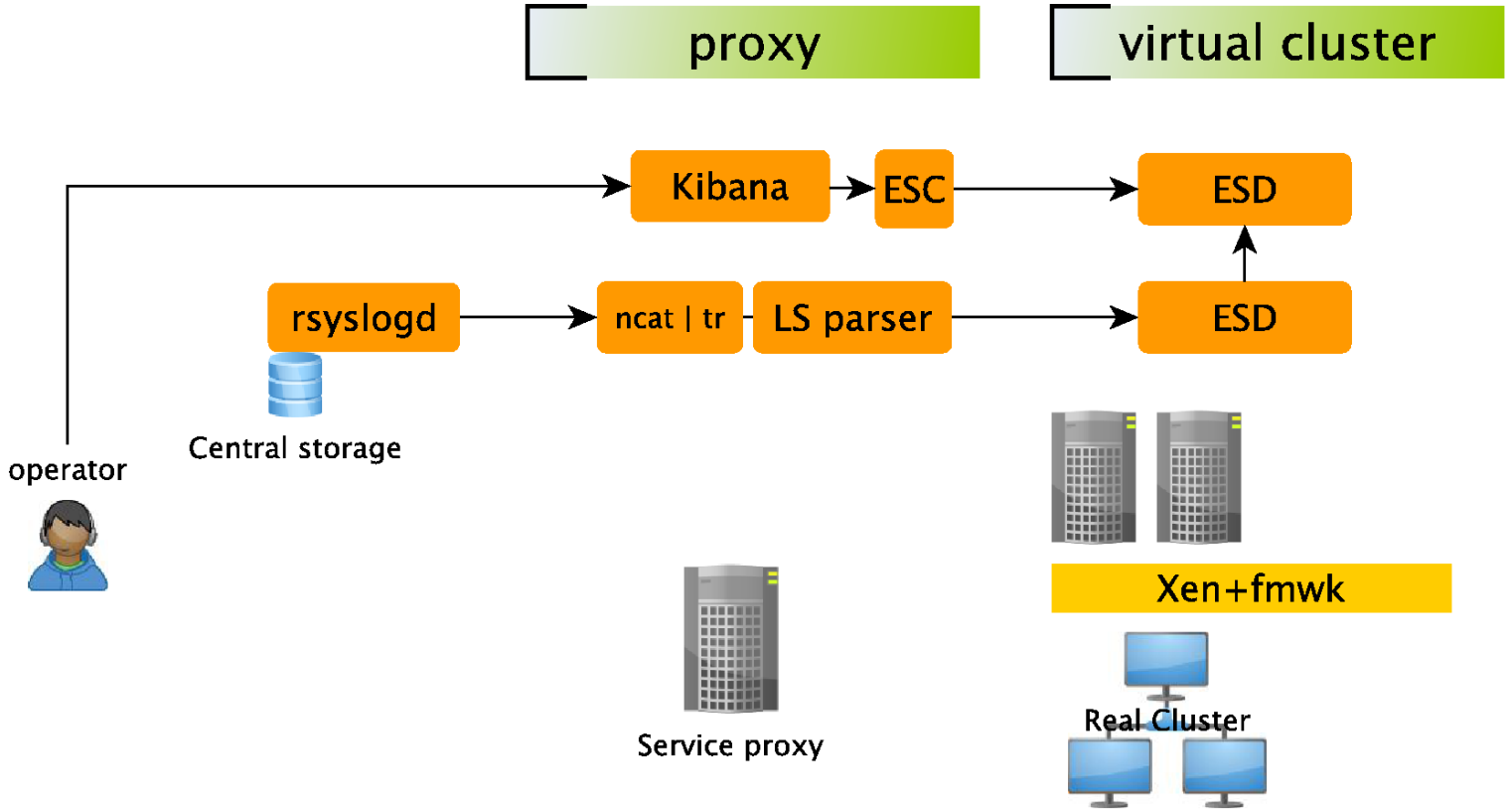
- alter
- anonymize
- checksum
- csv
- date
- dns
- environment
- gelfify
- geoip
- grep
- grok
- grokdiscovery
- json
- kv
- metrics
- multiline
- mutate
- noop
- split
- syslog_pri
- urldecode
- xml
- zeromq

outputs

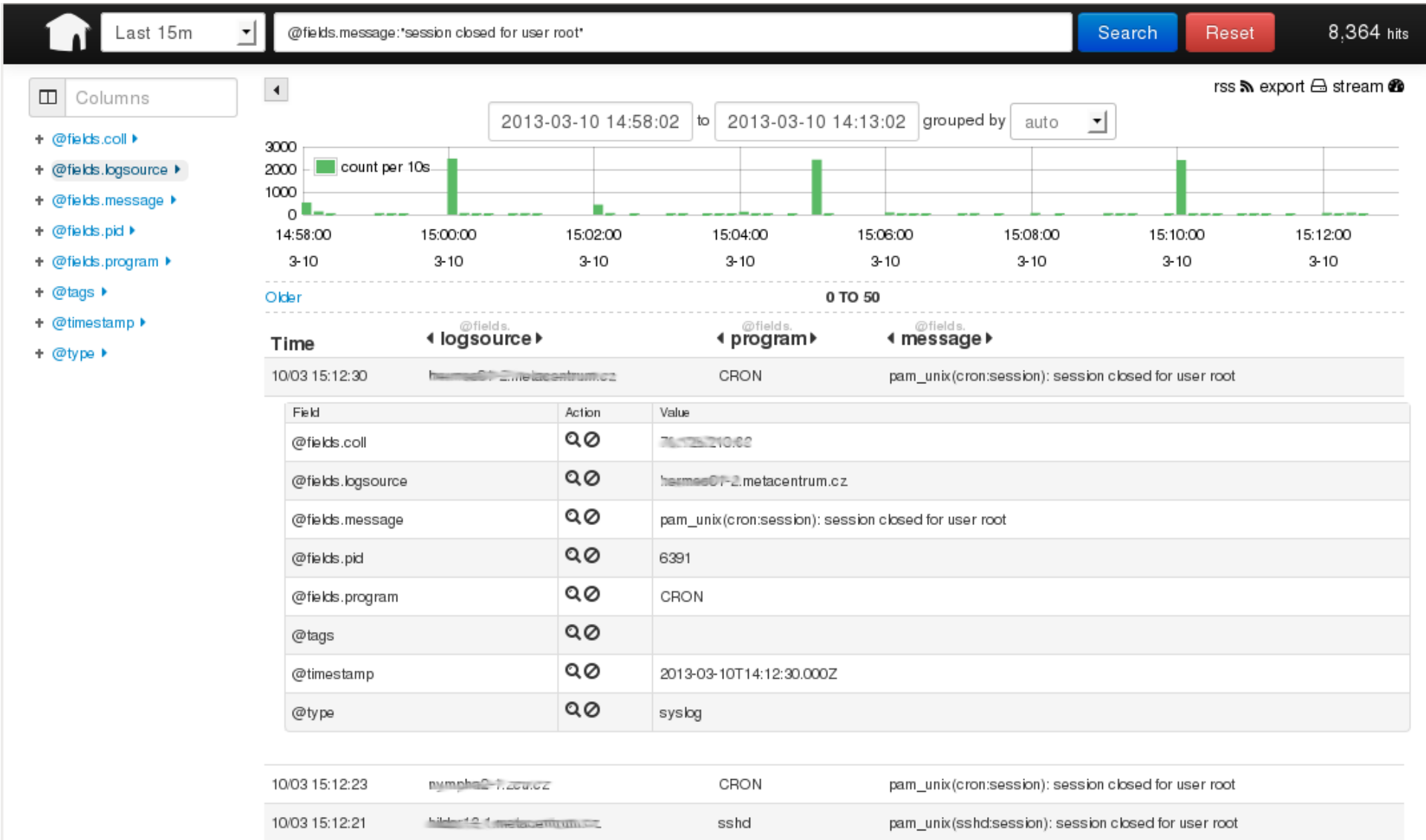
- amqp
- boundary
- circonus
- cloudwatch
- datadog
- elasticsearch
- elasticsearch_http
- elasticsearch_river
- email
- exec
- file
- ganglia
- gelf
- gemfire
- graphite
- graphtastic
- http
- internal
- irc
- juggernaut
- librato
- loggly
- lumberjack
- metriccatcher
- mongodb
- nagios
- nagios_nsca
- null
- opentsdb
- pagerduty
- pipe
- redis
- riak
- riemann
- sns
- sqs
- statsd
- stdout
- stomp
- syslog
- tcp
- websocket

Log processing proxy

- ES + LS + Kibana
  - ... or even simpler (ES embedded in LS)



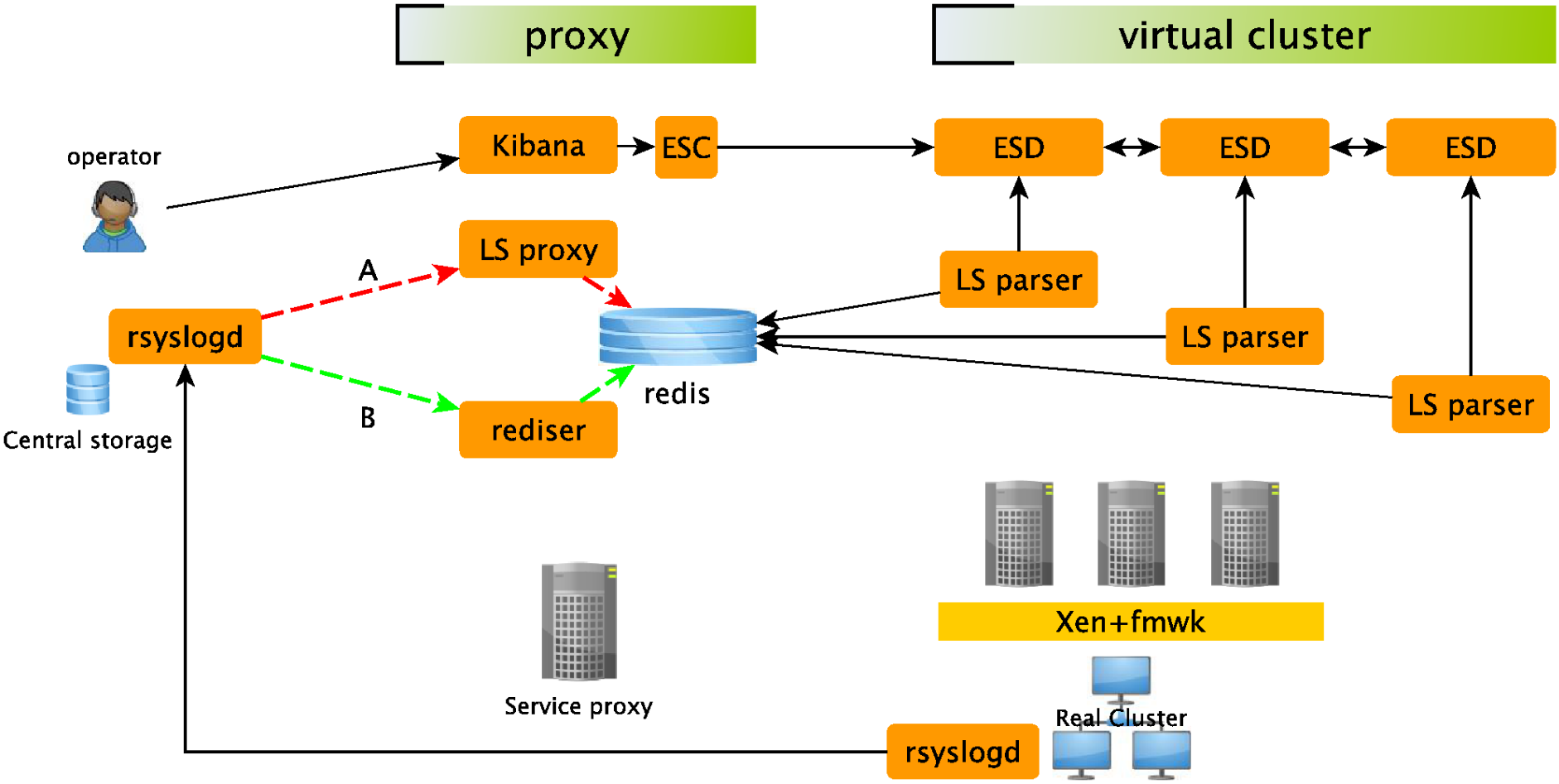
btw Kibana


Last 15m
@fields.message:"session closed for user root"
Search
Reset
8,364 hits
Columns
@fields.coll
@fields.logsource
@fields.message
@fields.pid
@fields.program
@tags
@timestamp
@type
rss export stream
2013-03-10 14:58:02
to
2013-03-10 14:13:02
grouped by
auto
count per 10s
3000
2000
1000
0
14:58:00
15:00:00
15:02:00
15:04:00
15:06:00
15:08:00
15:10:00
15:12:00
3-10
Older
0 TO 50
Time
@fields. logsource
@fields. program
@fields. message
10/03 15:12:30
CRON
pam_unix(cron:session): session closed for user root
Field
Action
Value
@fields.coll
@fields.logsource
metacentrum.cz
@fields.message
pam_unix(cron:session): session closed for user root
@fields.pid
6391
@fields.program
CRON
@tags
@timestamp
2013-03-10T14:12:30.000Z
@type
syslog
10/03 15:12:23
CRON
pam_unix(cron:session): session closed for user root
10/03 15:12:21
sshd
pam_unix(sshd:session): session closed for user root

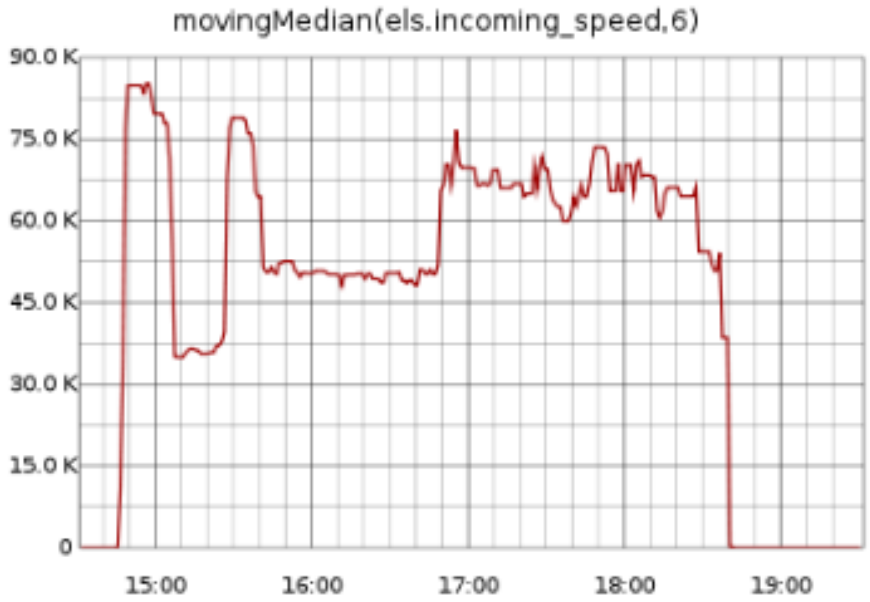
Performance

- Proxy parser might not be enough for grid logs ..
  - creating cloud service is easy with LS, all we need is a spooling service >> redis
- Speeding things up
  - batching, bulk indexing
  - rediser
    - bypassing logstash internals overhead on a hot spot (proxy)
- ~~Logstash does not implement all necessary features yet~~
  - ~~http time flush, synchronized queue ...~~
    - ~~custom plugins, working with upstream ...~~

Cloud parser



LS + ES wrapup



- upload
  - testdata
    - logs from January 2013
    - 105GB -- cca 800M events
  - uploaded in 4h
    - 8 nodes ESD cluster
    - 16 shared parsers (LS on ESD)
    - 4 nodes cluster - 8h
  - speed vary because of the data (lots of small msgs)
    - during normal operations a large cluster is not needed

LS + ES wrapup

- Speed of ES upload depends on
  - size of grokked data and final documents,
  - batch/flush size of input and output processing,
  - filters used during processing,
  - LS outputs share sized queue which can block processing (lanes:),
  - elasticsearch index (template) setting.
  - ...
  - ...
  - tuning for top speed is manual job (graphite, ...)

LS + ES wrapup

- search speed ~

```
# du -sh .
105G .
# time grep -R "realuser" * > search.txt
real 18m39.447s
user 1m7.796s
sys 1m24.565s
# wc search.txt
   81636  1773549 21777400 search.txt
#

# ./el_listnodes.py
10.0.0.31 id HRf5TJebQw6_cYxAsS2mtQ indices.docs.count 388345192
10.0.0.3 id BBcHTUk9SkWxhynzoPafog indices.docs.count 409604004
10.0.0.1 id 1pTq45TKTGavwnZGKXPcfQ indices.docs.count 0
# time sh curltest.sh > search1
real 0m34.944s
user 0m0.536s

# grep '"_id"' search1 |wc
  81636  244908 3265440
```

Advanced log analysis

- ES is a fulltext SE, not a database
  - but for analytics a DB is necessary

- Document-Oriented Storage
  - Schemaless document storage
  - Auto-Sharding
  - Mapreduce and aggregation framework

Advanced log analysis

- MongoDB
  - Can be fed with grokked data by Logstash
    - sshd log analysis

```
AAARESULT (?:Accepted|Failed|Authorized|identification|Invalid|disconnect|tried|refused)
METHOD (?:[a-z-]+|correct key)
PRINCIPAL [a-zA-Z0-9_/-]+@%{HOSTNAME}

AUTHN %{AAARESULT:result} %{METHOD:method} for (invalid user )?%{USER:user} from %{IPORHOST:rem
AUTHZ %{AAARESULT:result} to %{USER:user}, krb5 principal %{PRINCIPAL:principal} \(krb5_kuserok
SCAN Did not receive %{AAARESULT:result} string from %{IPORHOST:remote}
INVALID %{AAARESULT:result} user %{USER:user} from %{IPORHOST:remote}
DISCONNECT Received %{AAARESULT:result} from %{IPORHOST:remote}: 11: disconnected by user
WRONGKEY Authentication %{AAARESULT:result} for %{USER:user} with %{METHOD:method} but not from
REFUSED %{AAARESULT:result} connect from %{IPORHOST:remote} \(%{IPORHOST:remote}\)

SSHATTEMPT (?:%{AUTHN}|%{AUTHZ}|%{SCAN}|%{INVALID}|%{DISCONNECT}|%{WRONGKEY}|%{REFUSED})

SSHBASE3 (%{SYSLOGTIMESTAMP} (%{IP:coll} )?%{SYSLOGHOST:logsource} )?%{SYSLOGTIMESTAMP:timestam
SSHLINE %{SSHBASE3} %{SSHATTEMPT:message}
```

MapReduce

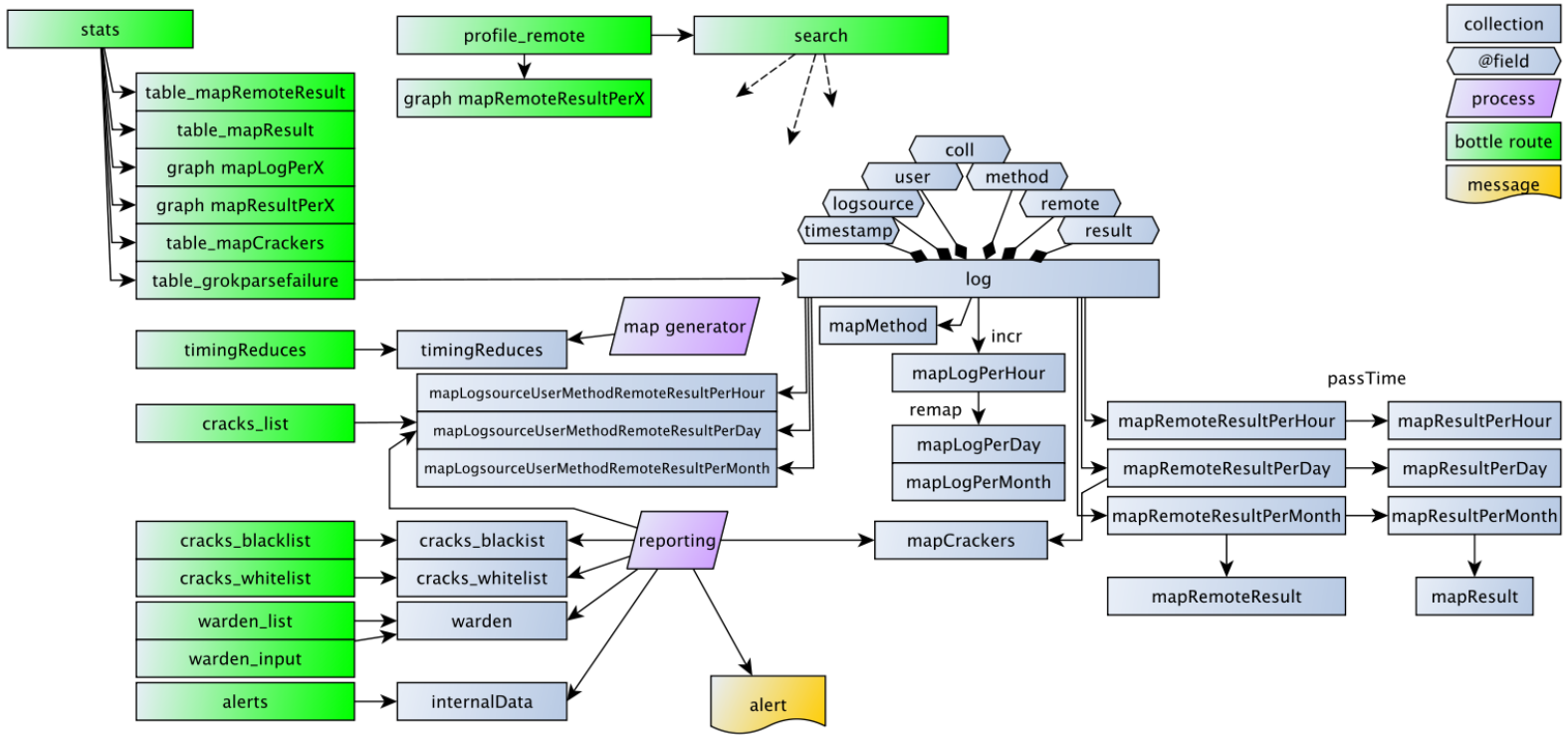
```
'map': Code("
                function() {
                        var a = new Date(
                                this.@timestamp.getFullYear(),
                                this.@timestamp.getMonth(),
                                this.@timestamp.getDate(),
                                this.@timestamp.getHours(),
                                0, 0, 0);
                        emit(
                                { t: a,
                                logsource: (this.@fields.logsource ? this.@fields.logsource.toString() : 'NULL'),
                                user: (this.@fields.user ? this.@fields.user.toString() : 'NULL'),
                                method: (this.@fields.method ? this.@fields.method.toString() : 'NULL'),
                                remote: (this.@fields.remote ? this.@fields.remote.toString() : 'NULL'),
                                result: (this.@fields.result ? this.@fields.result.toString() : 'NULL'),
                                },
                                {count: (this.value ? this.value.count : 1)}
                        );
                }",
        {}),

'reduce': Code('
                function(k,v) {
                        var r = { count:0 } ;
                        v.forEach(function(v) {r.count+=v.count });
                        return r;
                }',
        {}),
```

```
{
   "_id": ObjectId("51379dd1e4b0fad32a766fa7"),
   "@timestamp": ISODate("2013-02-03T13:12:44.0Z"),
   "@tags": [],
   "@fields": {
     "coll": {
       "0": "ddd.aaa.bbb.ccc"
    },
     "logsource": {
       "0": "wknode23.sub.domain.cz"
    },
     "result": {
       "0": "Invalid"
    },
     "user": {
       "0": "prueba"
    },
     "remote": {
       "0": "218.85.135.29"
    }
  },
   "@message": "Feb 3 14:12:42 ddd.aaa.bbb.ccc wknode23.sub.domai
   "@type": "ssh"
}
```

Mongomine

- on the top of created collection
  - time based aggregations (profiling, browsing)
  - custom views (mapCrackers)
    - `mapRemoteResultsPerDay.find( {time= last 14days, result={fail}, count>20} )`
  - external data (Warden, torlist)



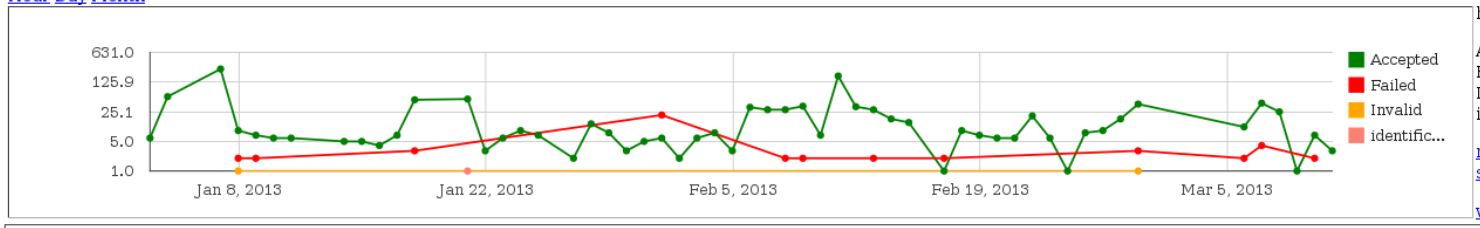
Mongomine

- Logstash + MongoDB application
  - sshd log analysis
    - security events analysis
      - python bottle webapp
      - ~~Google~~ Highcharts
    - automated reporting
      - successful logins from
        - mapCrackers
        - Warden
        - Tor lists

Mongomine

hostname: [redacted] remote: [redacted] submit

Hour Day Month



remote: [redacted] -- hostname: [redacted] -- []

Accepted -- 1158.0
Failed -- 47.0
Invalid -- 2.0
identification -- 1.0

ripe:: [redacted]
sans.org:: [redacted]

whitelist blacklist

POST:{'remote': ['147.251.17.150'], 'limit': [100], 'collection': ['mapLogsourceUserMethodRemoteResultPerDay']}

hide selectorDiv

timestamp_begin:
timestamp_end:
logsource:
user:
principal:
remote: [redacted]
limit: 100
submit

methods: NULL, correct key, gssapi-with-mic, password, publickey, none — All None

results: Accepted, Authorized, Failed, Invalid, identification, refused, tried — All None

collection: internalData, log, mapCrackers, mapLogPerDay, mapLogPerHour, mapLogPerMonth, **mapLogsourceUserMethodRemoteResultPerDay**, mapLogsourceUserMethodRemoteResultPerHour, mapLogsourceUserMethodRemoteResultPerMonth, mapMethod, mapRemoteResult, mapRemoteResultPerDay, mapRemoteResultPerHour, mapRemoteResultPerMonth, mapResult

COLLECTION:mapLogsourceUserMethodRemoteResultPerDay
QUERY:{'_id.remote': '147.251.17.150', '@tags': {'$not': {'$in': ['grokparsefailure']}}} gorock
FOUND:TODO

Show 100 entries Search:

_id.0t	_id.logsource	_id.method	_id.remote	_id.result	_id.user	value.count
Mon Mar 11 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	1
Mon Mar 11 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	2
Sun Mar 10 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	1
Sun Mar 10 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	1
Sun Mar 10 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	1
Sun Mar 10 00:00:00 2013	[redacted]	publickey	[redacted]	Failed	[redacted]	2
Sun Mar 10 00:00:00 2013	[redacted]	password	[redacted]	Accepted	[redacted]	1
Sun Mar 10 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	1
Sun Mar 10 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	1
Sun Mar 10 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	1
Sat Mar 9 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	1
Fri Mar 8 00:00:00 2013	[redacted]	gssapi-with-mic	[redacted]	Accepted	[redacted]	2

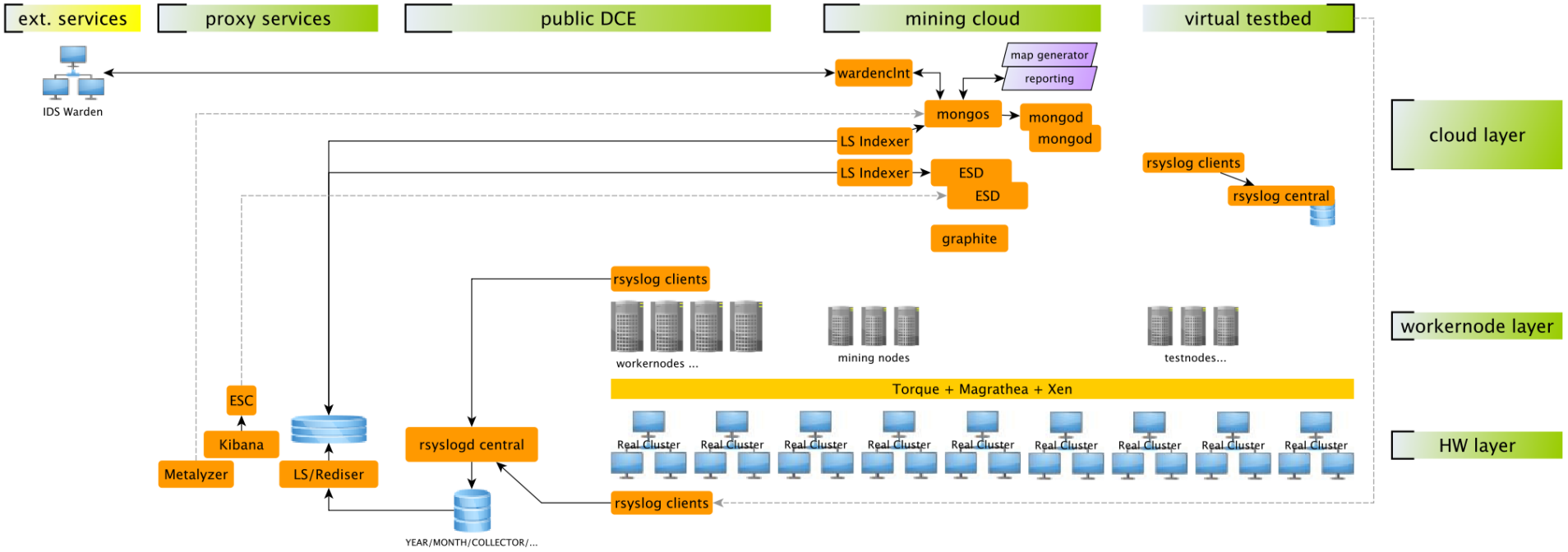
Mongomine wrapup

- testcase
  - 20GB -- January 2013
  - 1 MongoDB node, 24 CPUs, 20 shards
  - 1 parser node, 6 LS parsers

- speed
  - upload -- approx. 8h (no bulk inserts :(
  - 1st MR job -- approx. 4h
  - incremental MR during normal ops -- approx. 10s

Overall schema

- rsyslogd + testbed
- LS + ES
- LS + Mongomine + Ext



Virtual Machine Walkthrough

ESB EGI Technical forum 2013

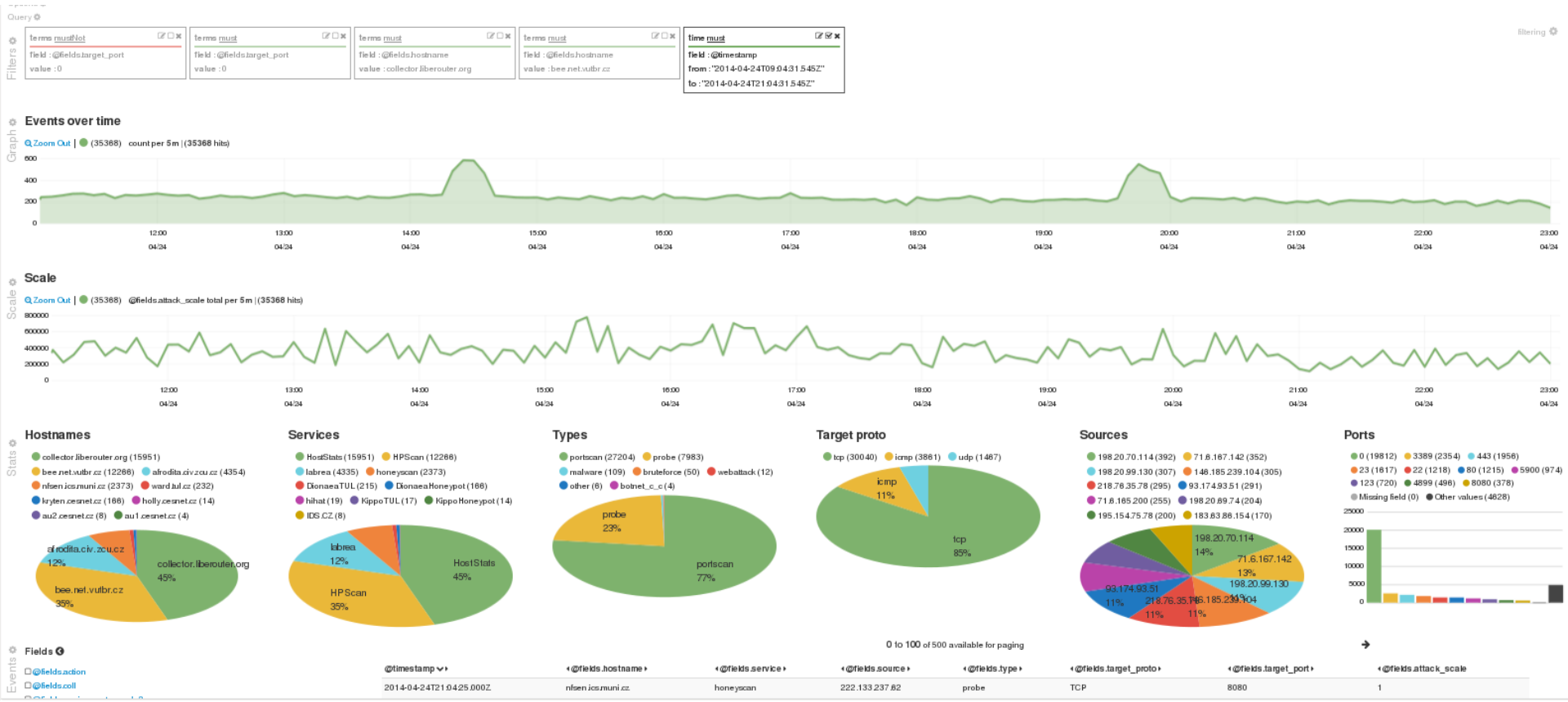
http://home.zcu.cz/~bodik/metasw/esbegitf/

Other Applications - Wardenweb2


```
{
        "priority":null,
        "source":"222.133.237.62",
        "target_proto":"TCP",
        "hostname":"nfsen.ics.muni.cz",
        "service":"honeyscan",
        "note":null,
        "attack_scale":"1",
        "detected":"2014-04-24 21:04:25",
        "timeout":null,
        "source_type":"IP",
        "type":"probe",
        "id":"57341436",
        "target_port":"8080"
}
```

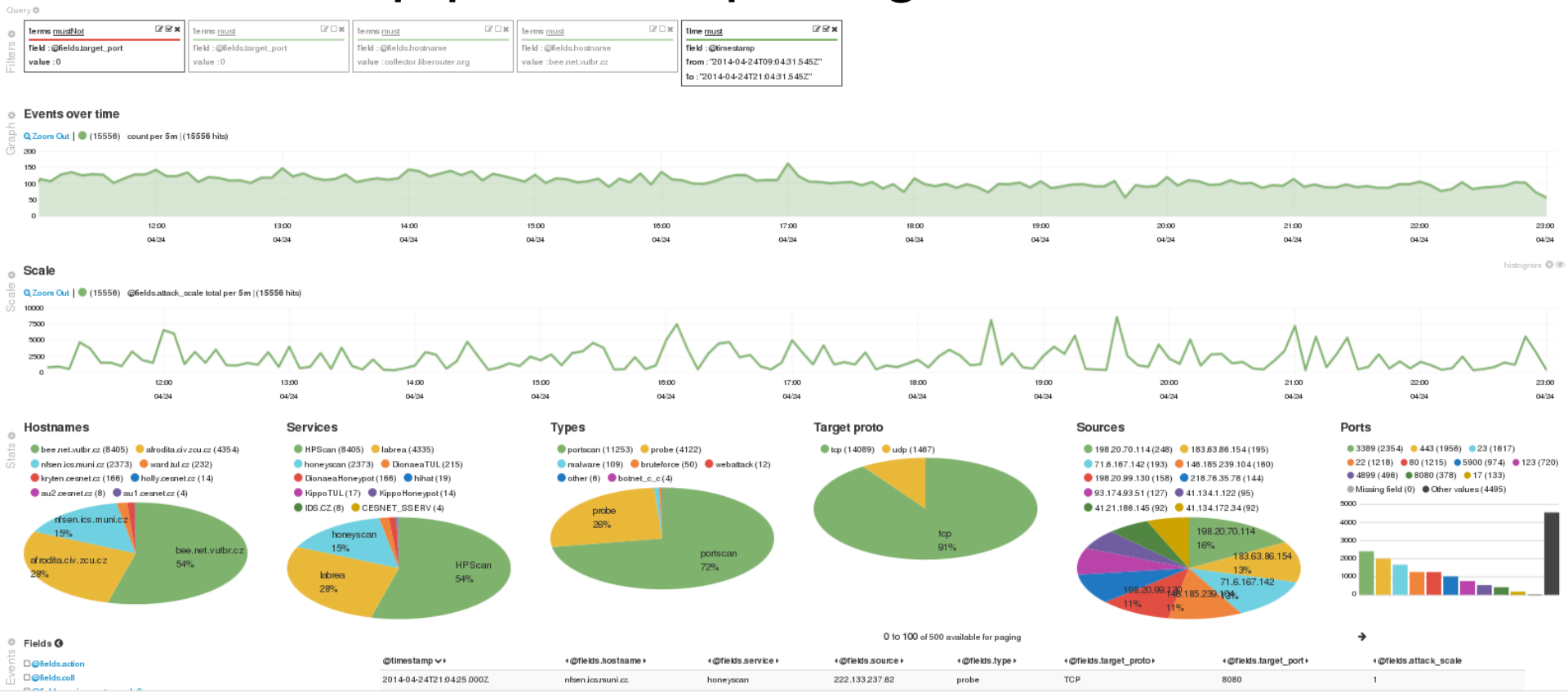
Other applications - wardenweb2

last 12 hours before yesterday's brewery event



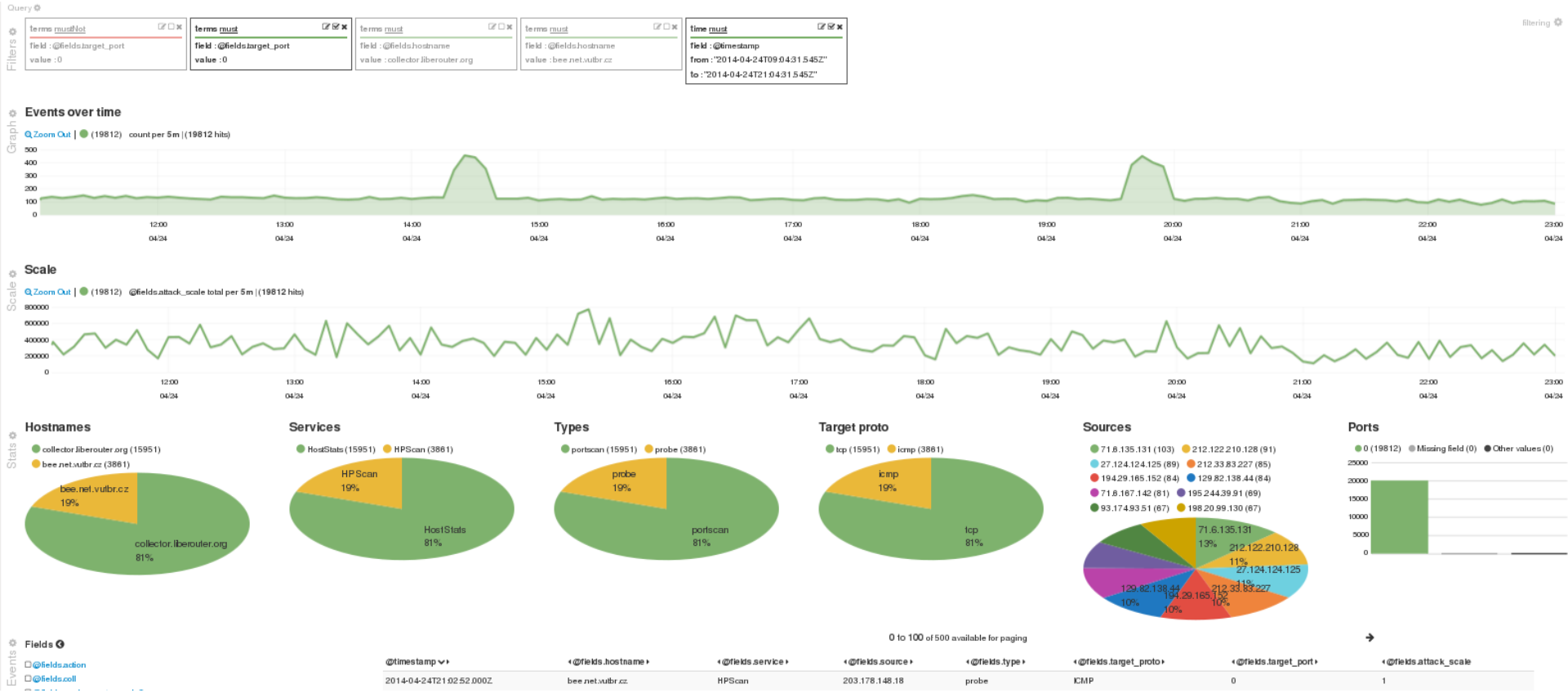
Other applications - wardenweb2

exclude top port 0 >> peak gone



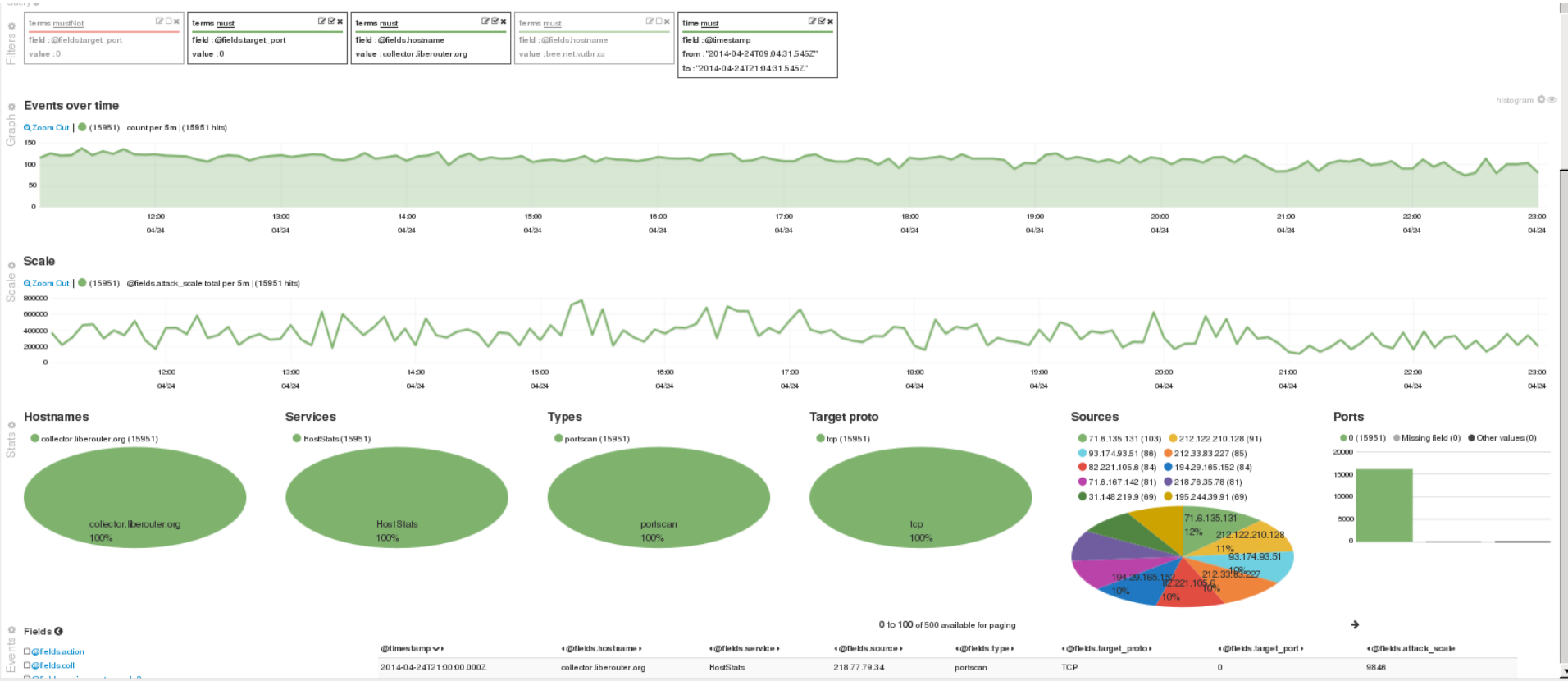
Other applications - wardenweb2

include top port 0 >> just 2 sensors left



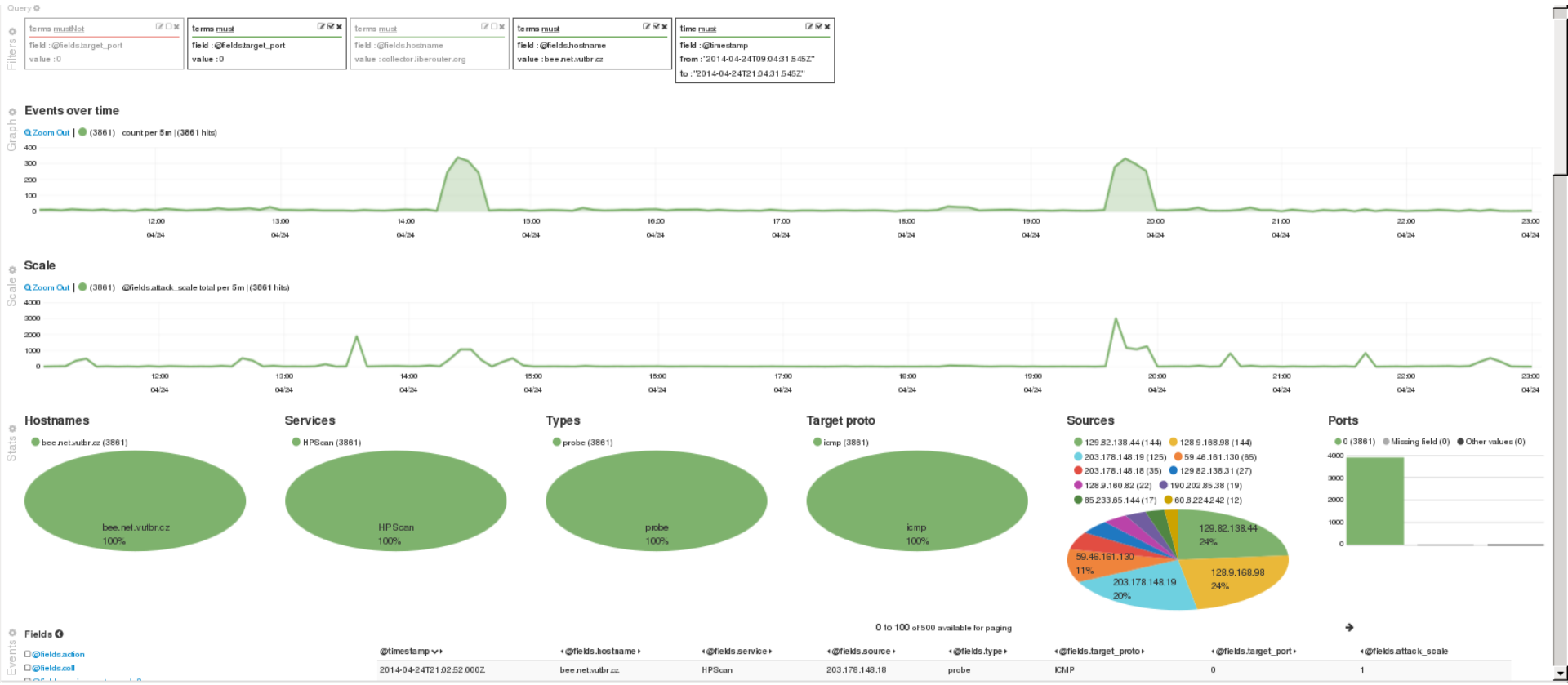
Other applications - wardenweb2

include top collector >> peak gone



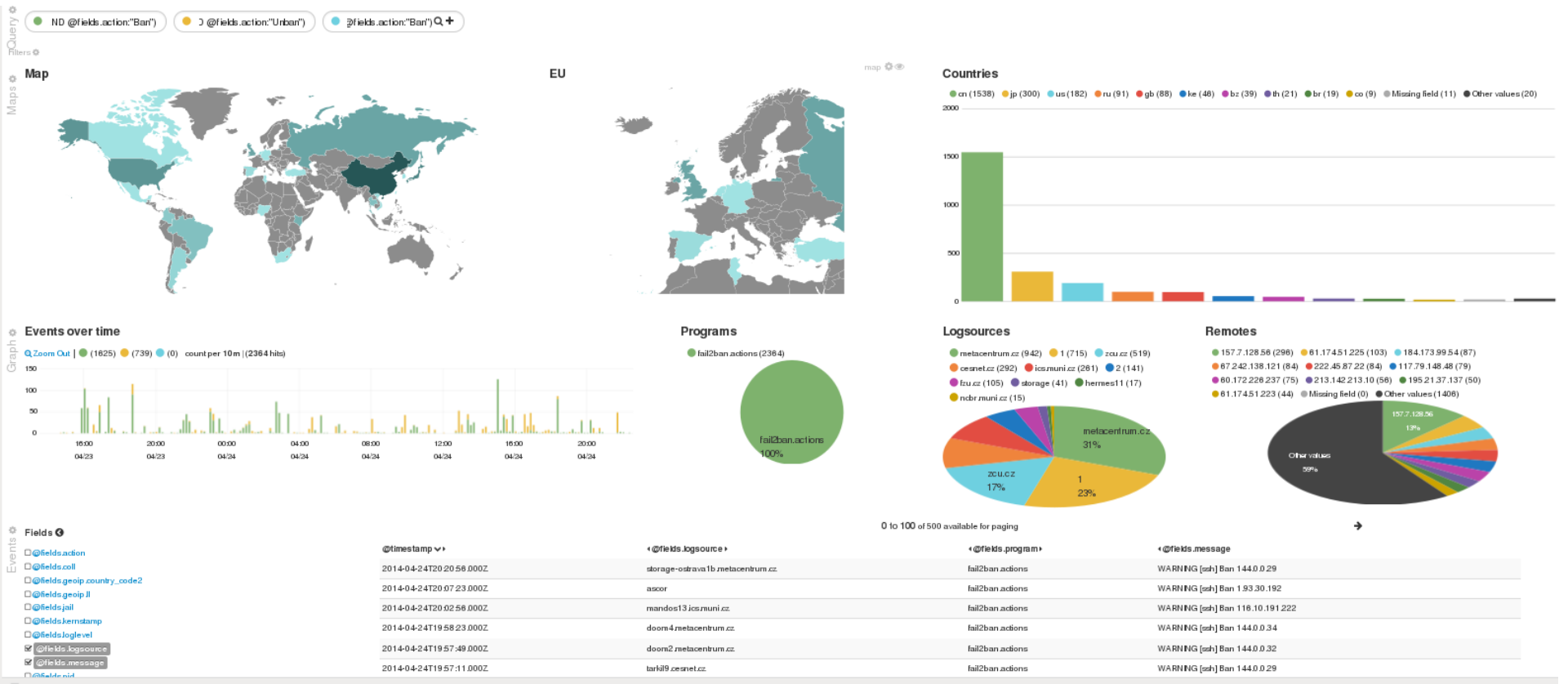
Other applications - wardenweb2

include the other >> peak >> icmp scan



Other applications - fail2ban + geoip

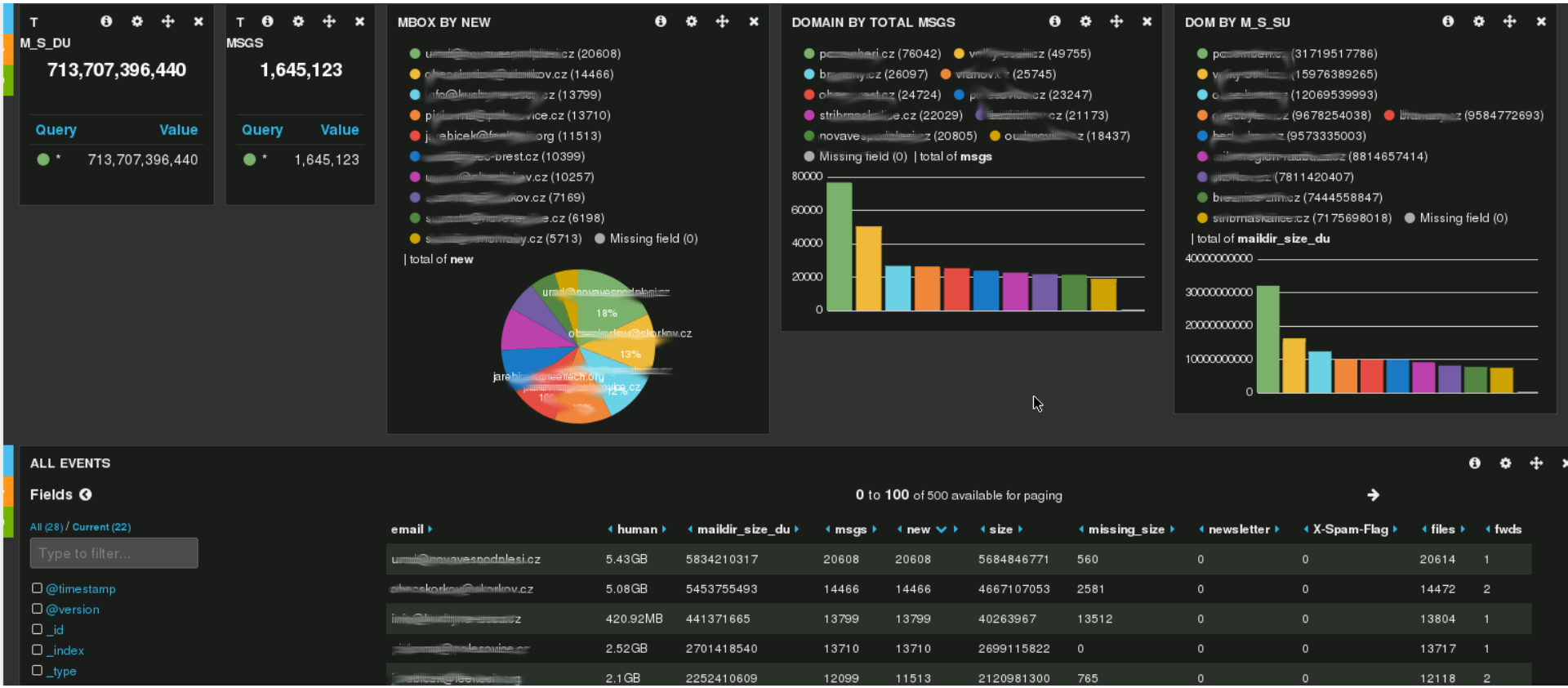
beside groking, logstash can do other things in the pipeline
>> geoip resolution



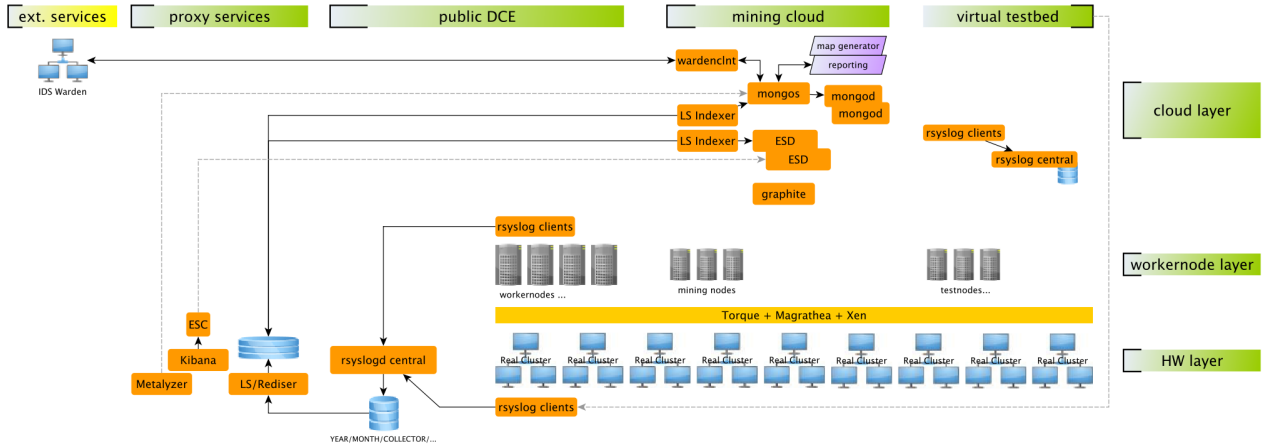
Other applications - maildir screening


```
{
        "files": 29,
        "domain": "xxx",
        "maildir": "/home/postdata/virtual/xxx/yyy",
        "msgs": 10,
        "fw": ["yyy@xxx"],
        "human": "786.25KB",
        "maildir_size_du": 805120,
        "missing_size": 0,
        "fwds": 1,
        "new": 7,
        "X-Spam-Flag": 0,
        "newsletter": 0,
        "calc_time": 0,
        "email": "yyy@xxx",
        "size": 766552
}
```

Other applications - maildir screening



Resume



- It works
  - system scales according current needs
  - custom patches published
  - solution is ready to accept new data
    - with any or almost no structure

- Features
  - collecting -- rsyslog
  - processing -- logstash
  - high interaction interface -- ES, kibana
  - analysis and alerting -- mongomine

Questions ?

now or …

https://wiki.metacentrum.cz/wiki/User:Bodik

http://bit.ly/RQ0LML

mailto:bodik@civ.zcu.cz
mailto:kouril@ics.muni.cz