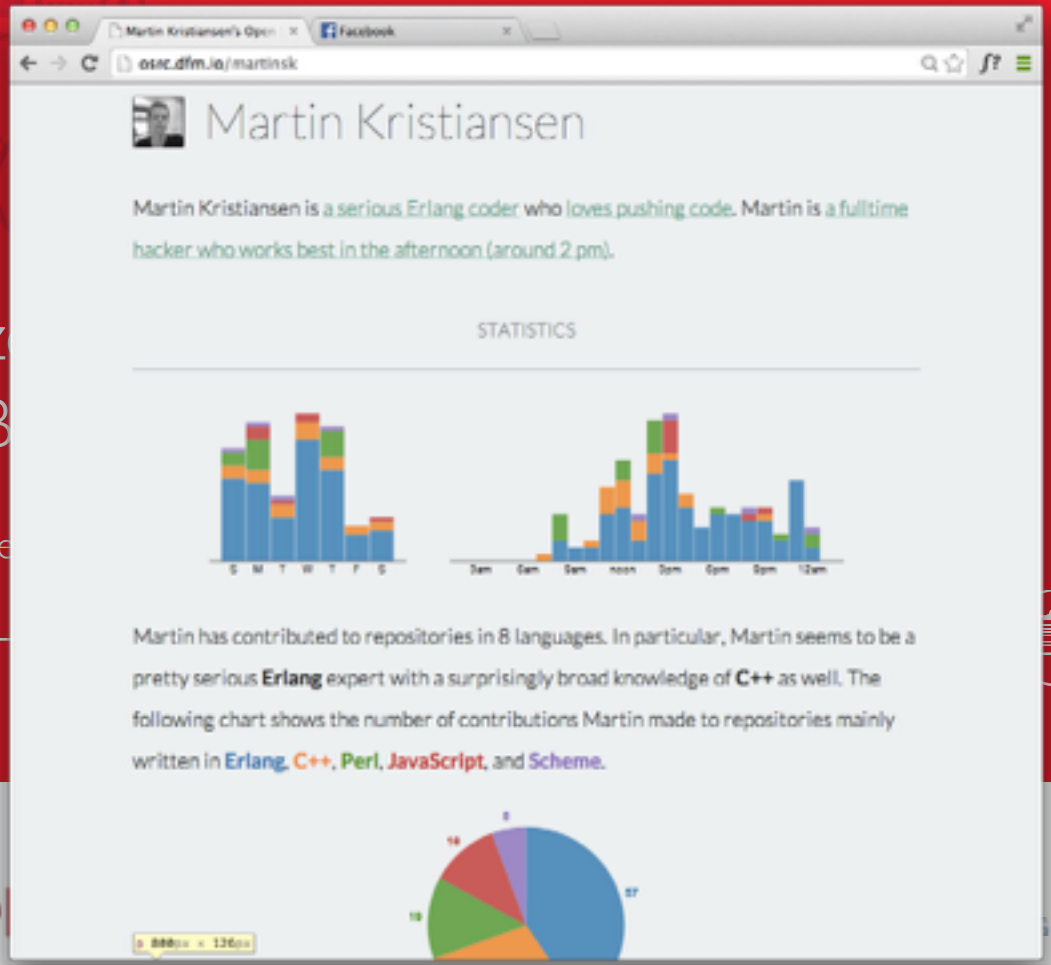
Centraliz
- Event-B

Martin Kristianse



Company Profile

tigertext

TigerText, the recognized leader in **secure texting for enterprises combines mobile technology on an OTT platform to help organizations securely communicate.**

Locations	Los Angeles, Silicon Valley, Boston, Chicago, Phoenix
Business Focus	Secure texting platform and application, serving the needs of regulated organizations in Healthcare, Financial Services, Government, Legal, and more
Usage	More than 3 million downloads and more than 3,000 facilities
Infrastructure	High-availability infrastructure leveraging Amazon's Web Services that utilize Tier IV, SaaS-70 Type II certified data centers
Delivery Scale	Delivering millions of messages per day

The Most Secure Organizations Trust TigerText

TigerText Infrastructure

Secure
HTTPS/XMPP
Connection

Network Provider:
AT&T | Verizon | Sprint

amazon web services™

- ✓ Encryption in Transit & at Rest
- ✓ High Availability
- ✓ Scalability

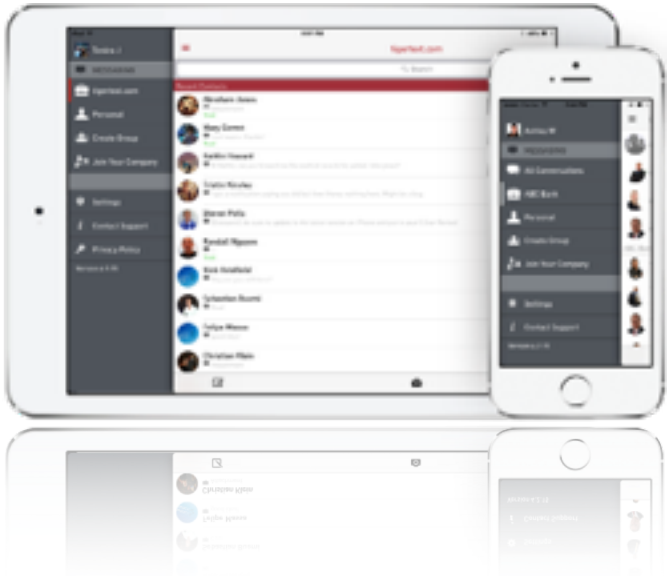
Secure
HTTPS/XMPP
Connection

Network Provider:
T-Mobile | WiFi

We are hiring!

email: msk@tigertext.com

Problem: How many messages are being send, how long did it take?



Standard Solution:

- Setup a database
- On every event input something into the database
- do regular runs on the database to detect changes

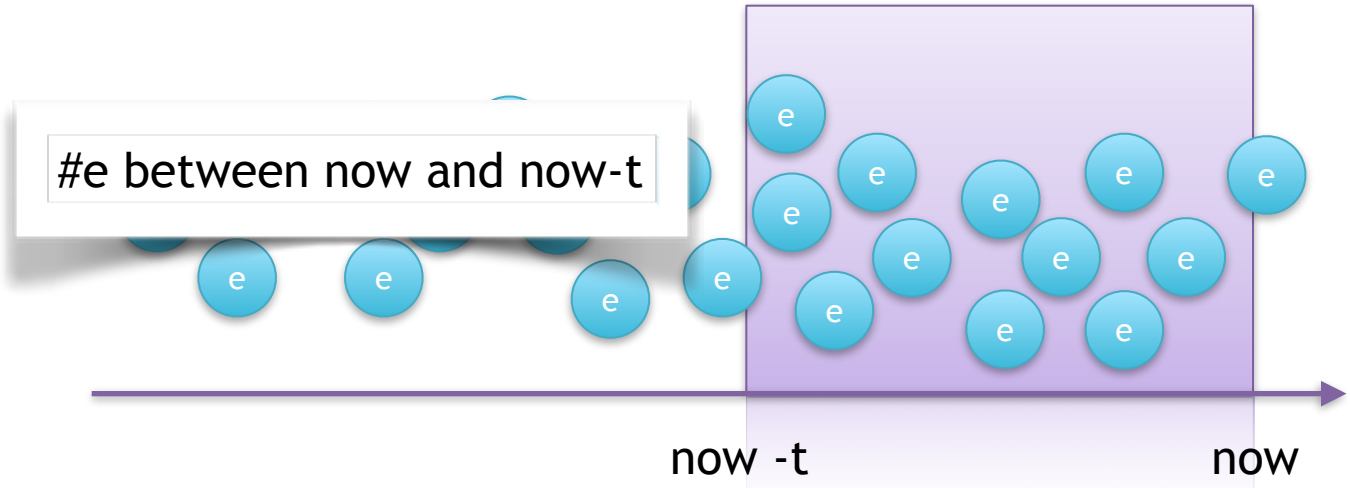
- Setup a database
- Do bucketing on the data in the database
- queries are faster, but bucketing determines resolution

Big-Ohh No!

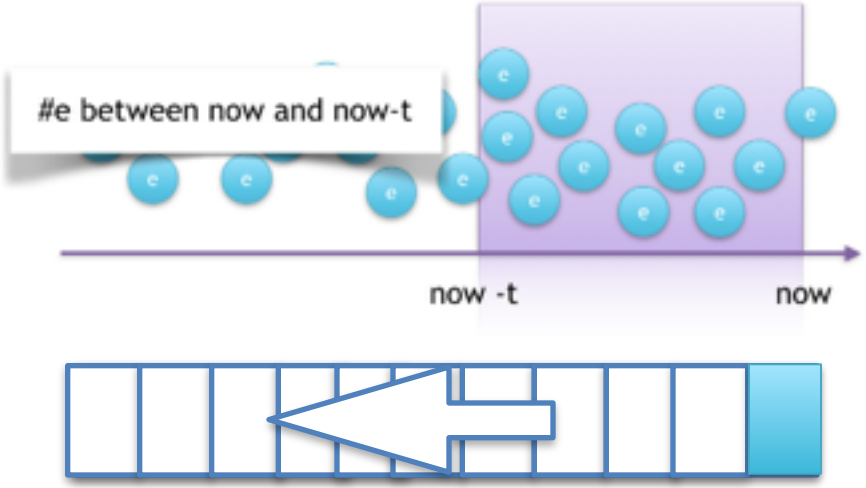
- Setup a database
- On every event input something into the database O(log(N)) for any entry
- do regular runs on the database to detect changes O(log(N)*d) for lookups

- Setup a database
- Do bucketing on the data in the database O(log(B)) for entries
- queries are faster, but bucketing determines resolution O(B*log(B)) for lookups

What if we want to measure something?



What if we want to measure something?



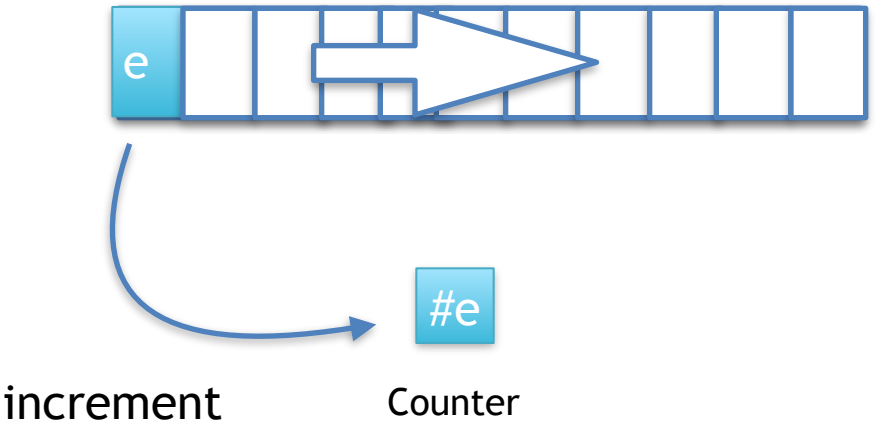
What to do?

- 10M events a day
- 2 months of data
- 560M entries
- Thats really just 2.1Gigs of memory if we are somehow able to store each entry in one 32bit word - no need for fancy software. we will make something our selves.

What to do?!?

Handle_Event

- Assign event an counter index
- Increment counter index
- push index onto queue



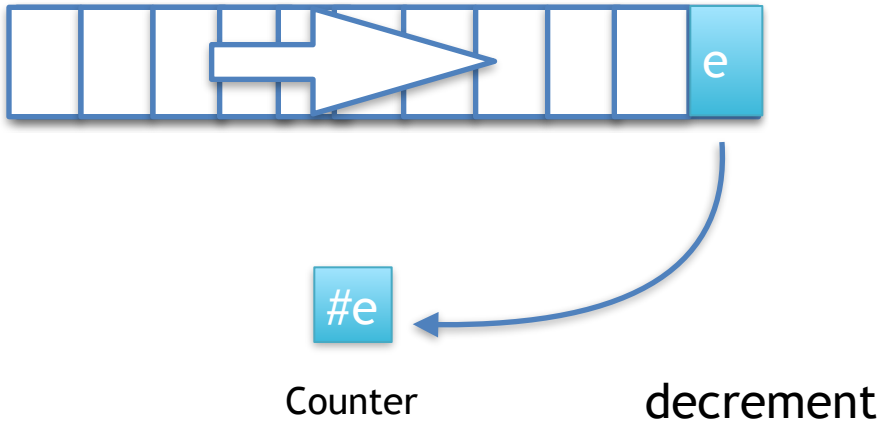
What to do?!?

Handle_Event

- Assign event an counter index
- Increment counter index
- push index onto queue

Handle_update

- Check end of queue for old entries.
- decrement counter



What to do?!?

Handle_Event

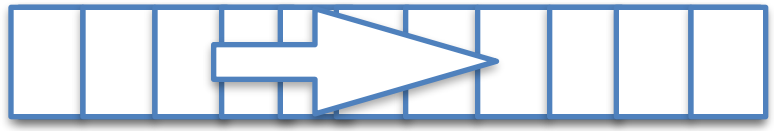
- Assign event an counter index
- Increment counter index
- push index onto queue

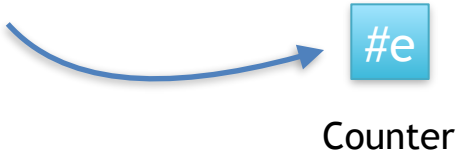
Handle_update

- Check end of queue for old entries.
- decrement counter

Query for event

- Simply fetch the index of the event
- return the counter





Real Layout

Fundamental Components

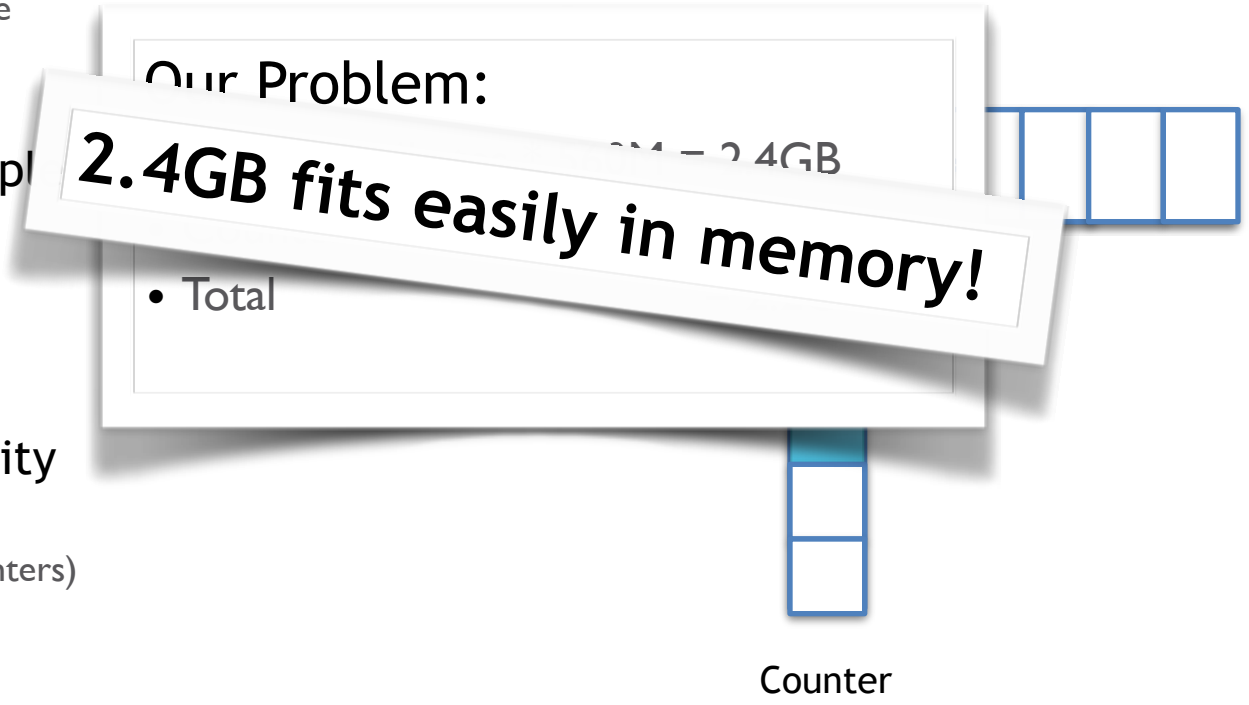
- Simple FIFO Queue
- Set of Counters

Operation Compl

- Inserts O(1)
- Expirations O(1)
- Queries O(1)

Space Complexity

- Queue O(#events)
- Counters O(#counters)

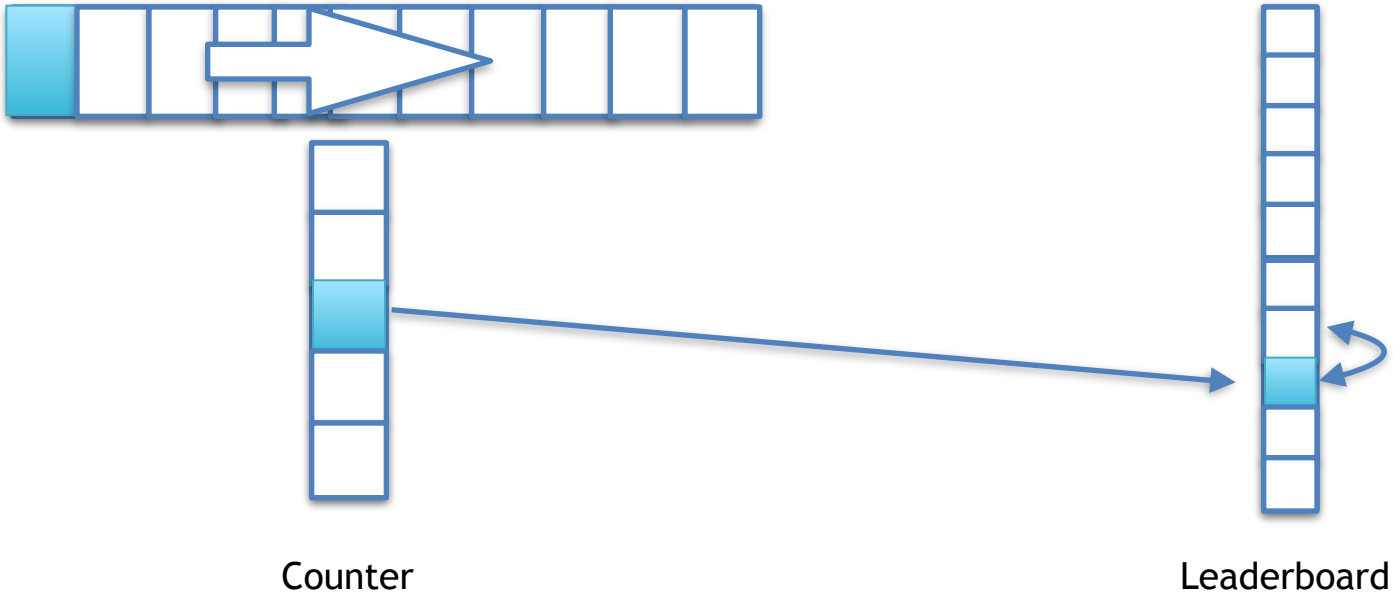


What can we use it for?

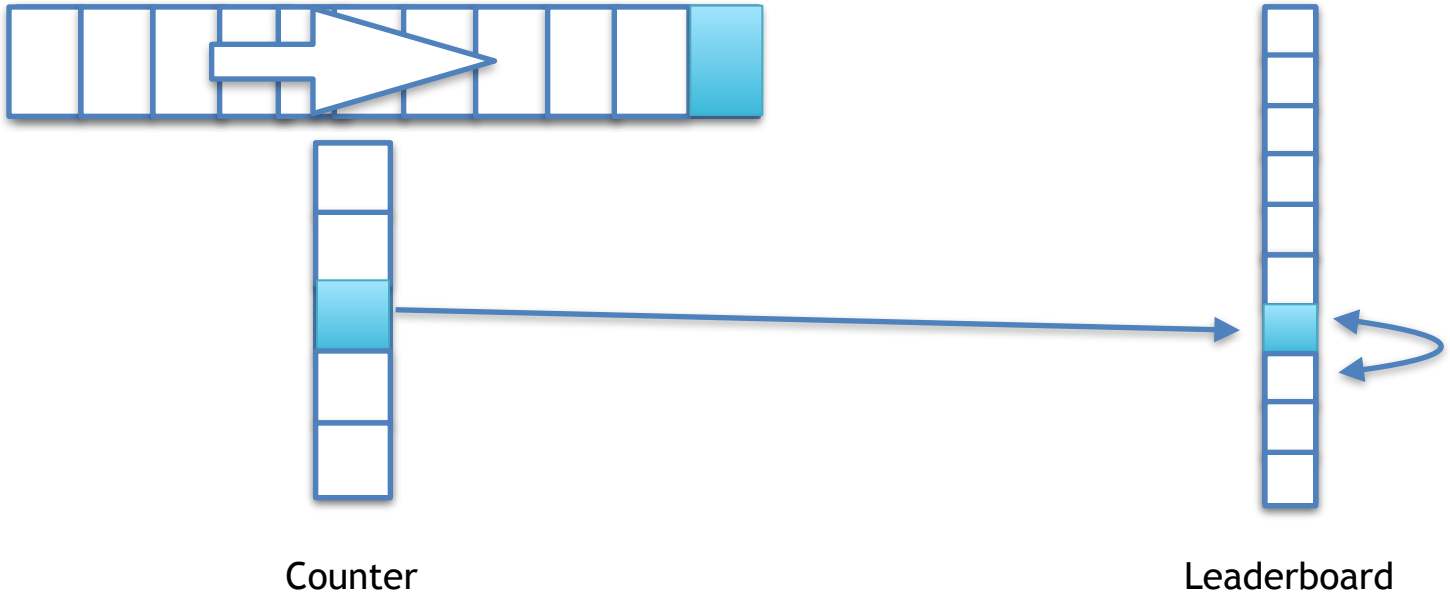
We want leaderboards!

Lists of counters, sorted and paginated

Increment

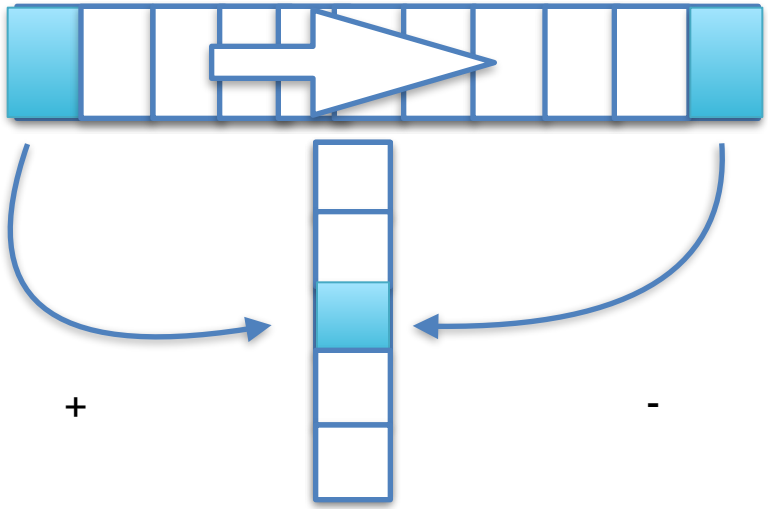


Decrement

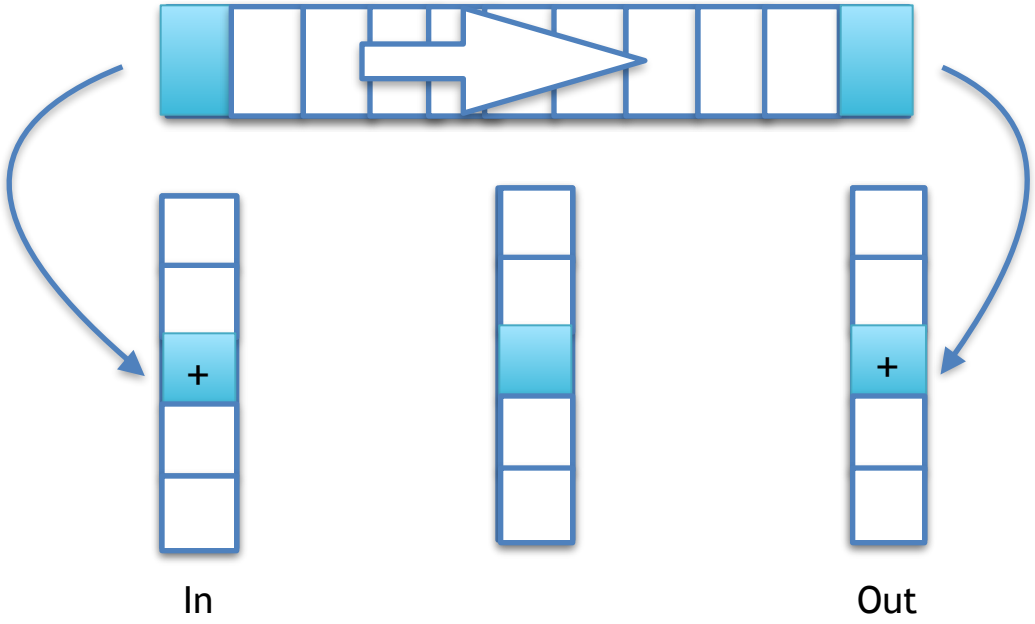


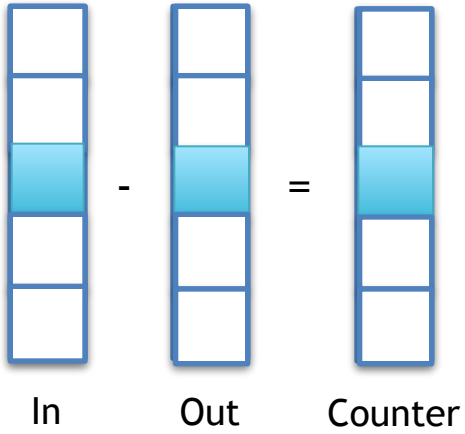
Optimizations ?

Note

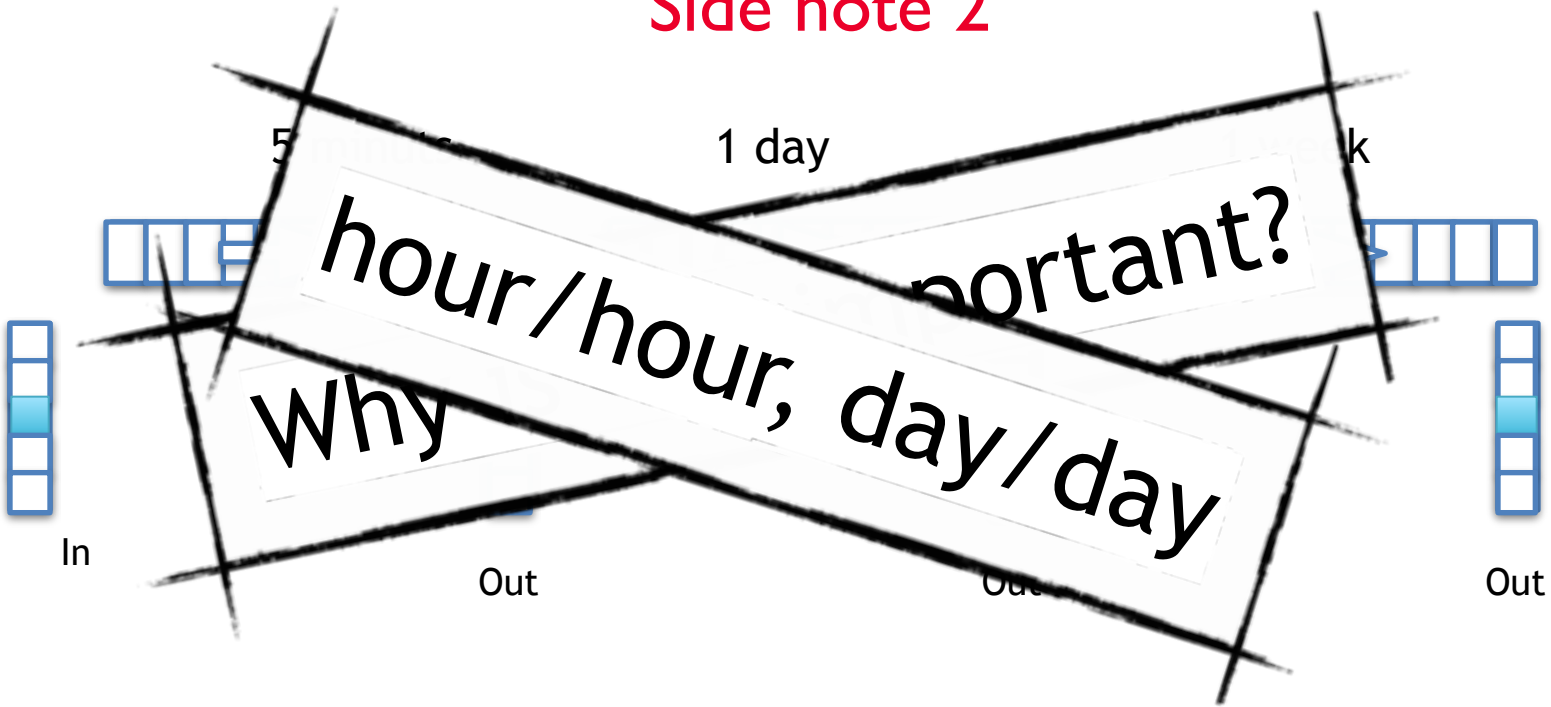


Side note





Side note 2

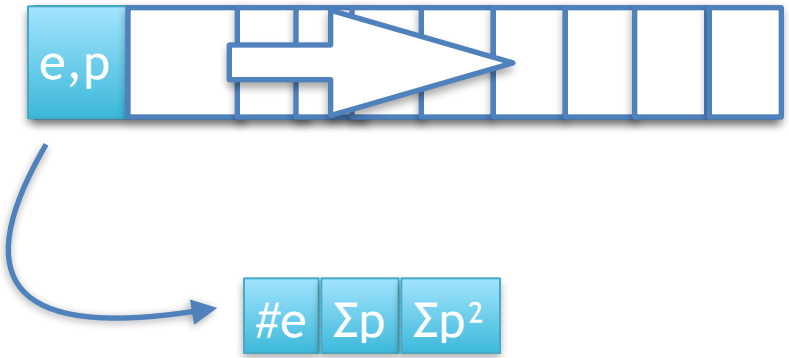


Adding Payloads

What if we want to measure something?

Instead of incoming events we now have incoming measurements

What if we want to measure something?



$$\text{Mean} = \Sigma p \,/\, \#e$$

$$\begin{aligned}\text{Var} &= \Sigma(\text{Mean} - p\_i)^2/\#e \\ &= \Sigma(\text{Mean}^2 + p\_i^2 - 2*\text{Mean}* p\_i)/\#e \\ &= (\#e*\text{Mean}^2 + \Sigma p^2 - 2*\text{Mean}*\Sigma p)\#e\end{aligned}$$

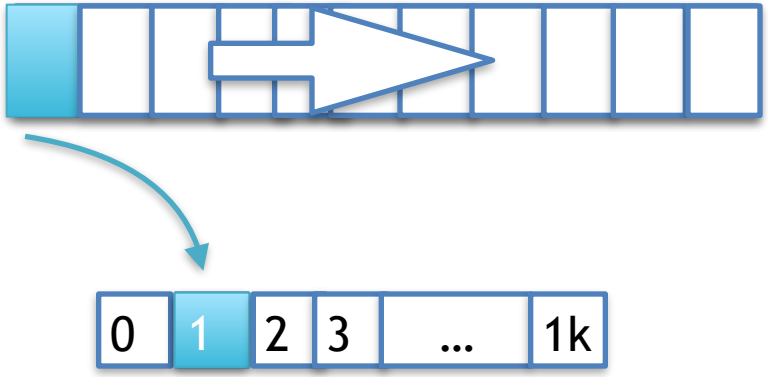
$$\text{Std-dev} = \text{sqrt}(\text{Var})$$

What can we use it for?

We want percentiles

80 % of events were less than X

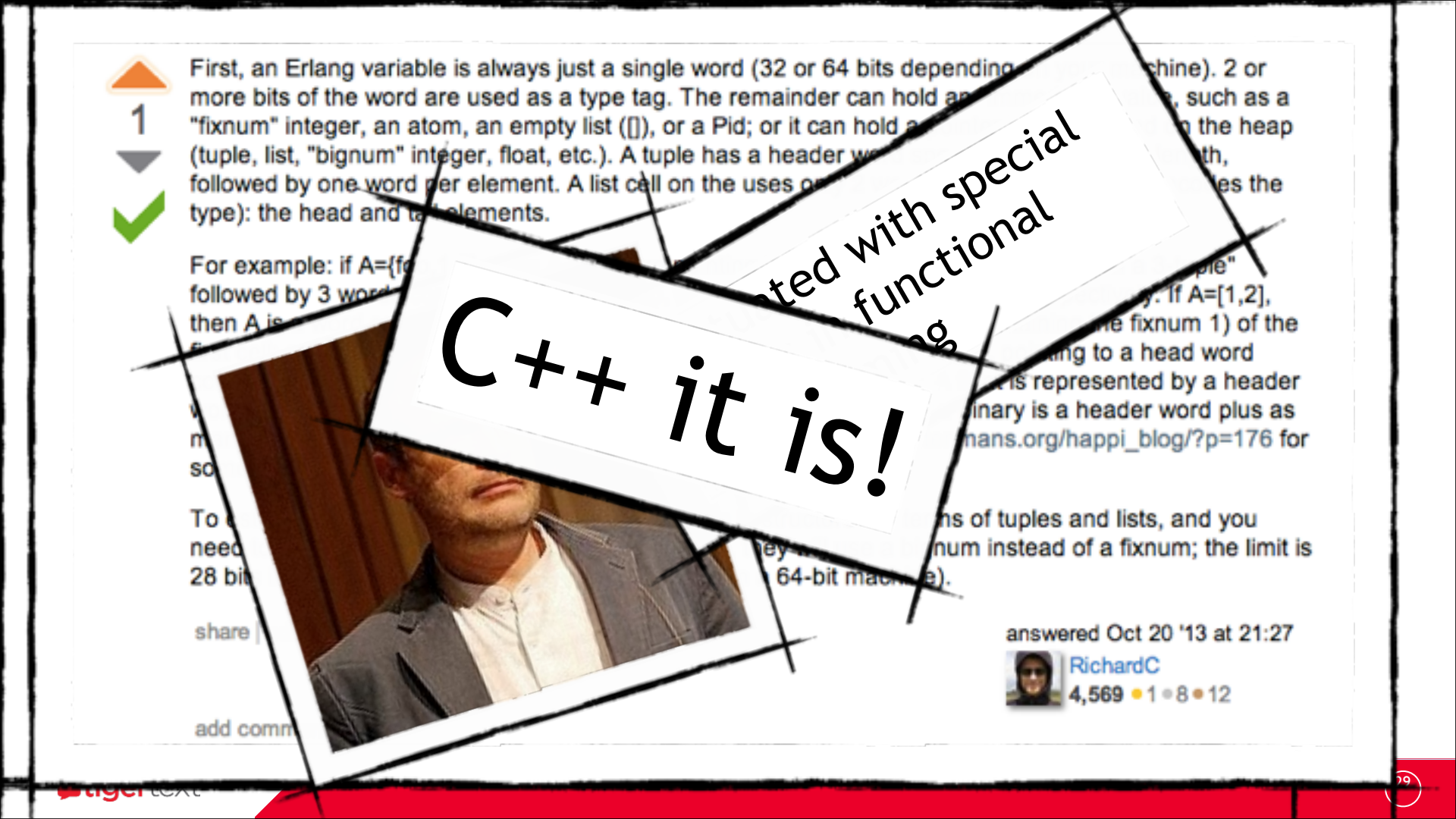
How to solve?



1.01^{x}

The Implementation and requirements

- Fit into our Erlang production environment.
- Be lightweight
- Scale - we are handling large amounts of data, but would like to handle more


1
First, an Erlang variable is always just a single word (32 or 64 bits depending
chine). 2 or
more bits of the word are used as a type tag. The remainder can hold an
, such as a
"fixnum" integer, an atom, an empty list ([]), or a Pid; or it can hold a
the heap
(tuple, list, "bignum" integer, float, etc.). A tuple has a header w
th,
followed by one word per element. A list cell on the uses o
es the
type): the head and ta elements.
For example: if A={fo
ple"
followed by 3 word
. If A=[1,2],
then A is
e fixnum 1) of the
ing to a head word
is represented by a header
inary is a header word plus as
nans.org/happi_blog/?p=176 for
so
To
ns of tuples and lists, and you
need
num instead of a fixnum; the limit is
28 bit
64-bit mac e).
share
answered Oct 20 '13 at 21:27
RichardC
4,569 1 8 12
add comm
ted with special
functional
ng
C++ it is!

Space Requirements

- 4 byte + one bit, per event,
- 4 byte per type of event
- 4 byte per counter in leaderboard

- 10M * 56 = 2.2GB,
- 3 M = 18 MB
- 3 M * 3 = 54 MB

Whats a good comparison?

Redis.io

- C++
- Speed beast.
- Does increment operations

Is this even a good idea?

ops pr call	eTally
1	81842 ops/sec
10	156K ops/sec

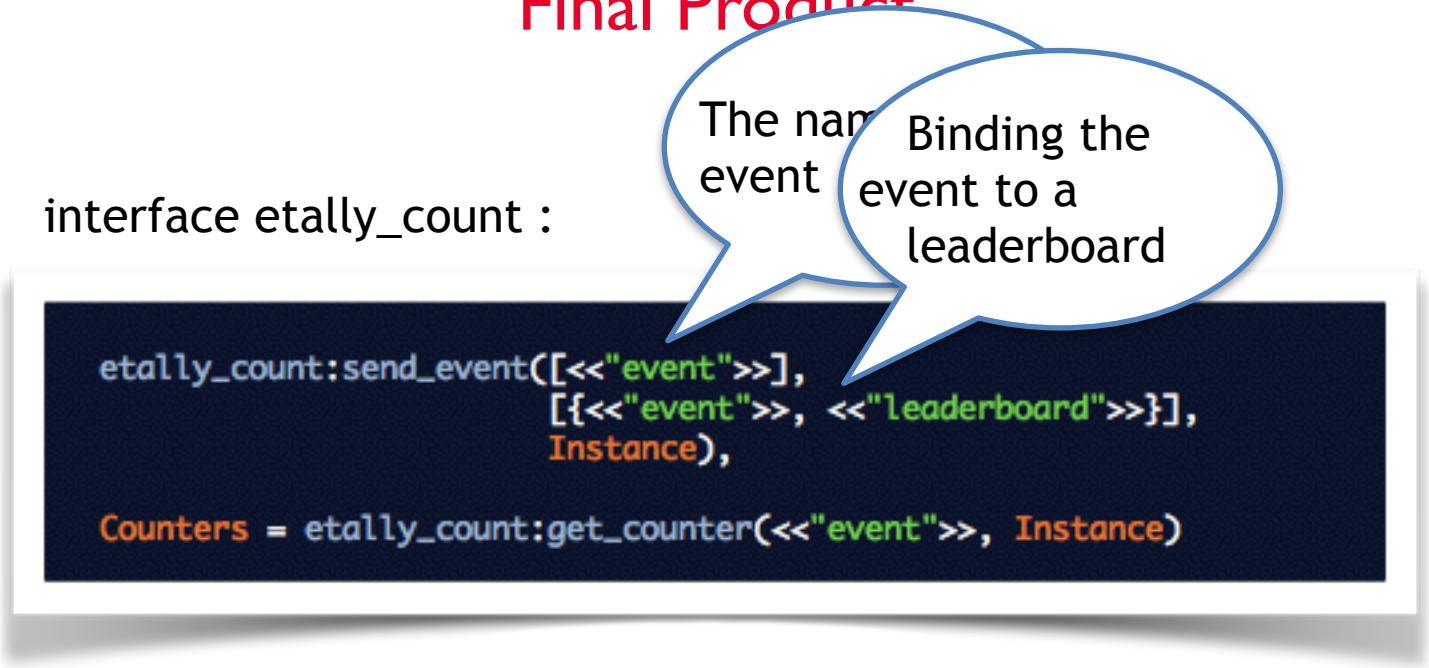
Final Product

Structure :

- etally_core : the codebase for the cnode, written in C++
- etally : the library for calling the cnode, simple fetches as described in the talk

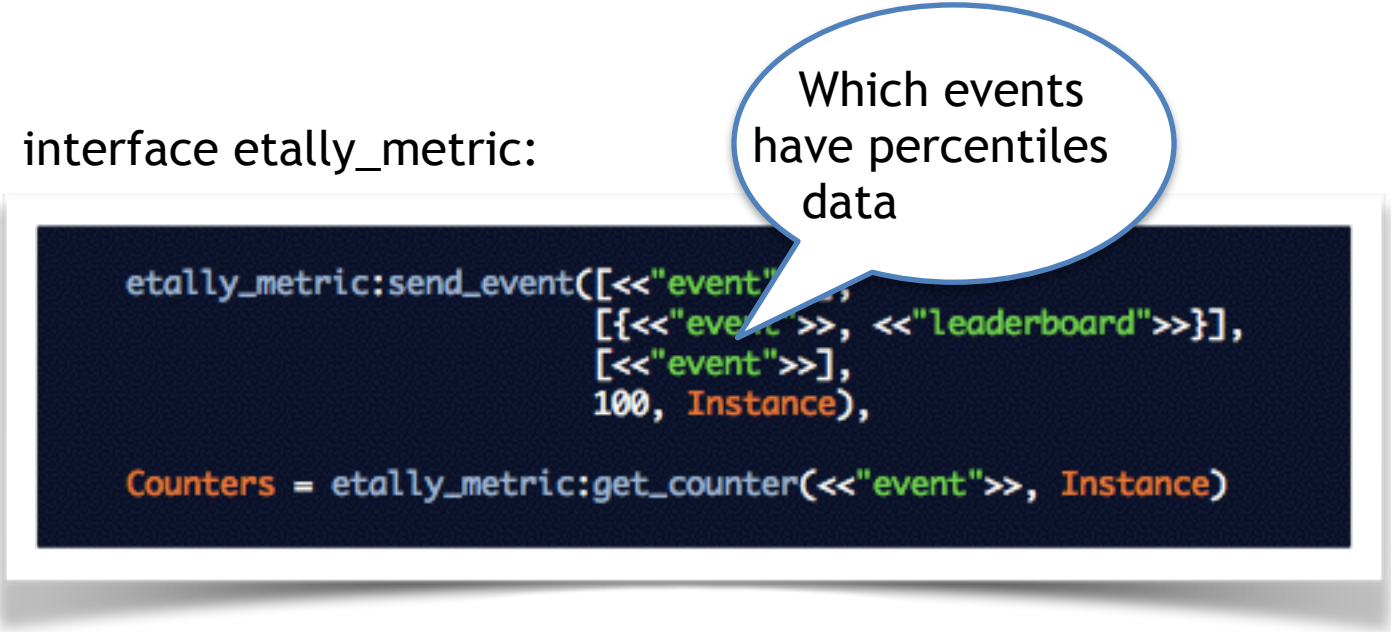
Final Product

interface etally_count :



Final Product

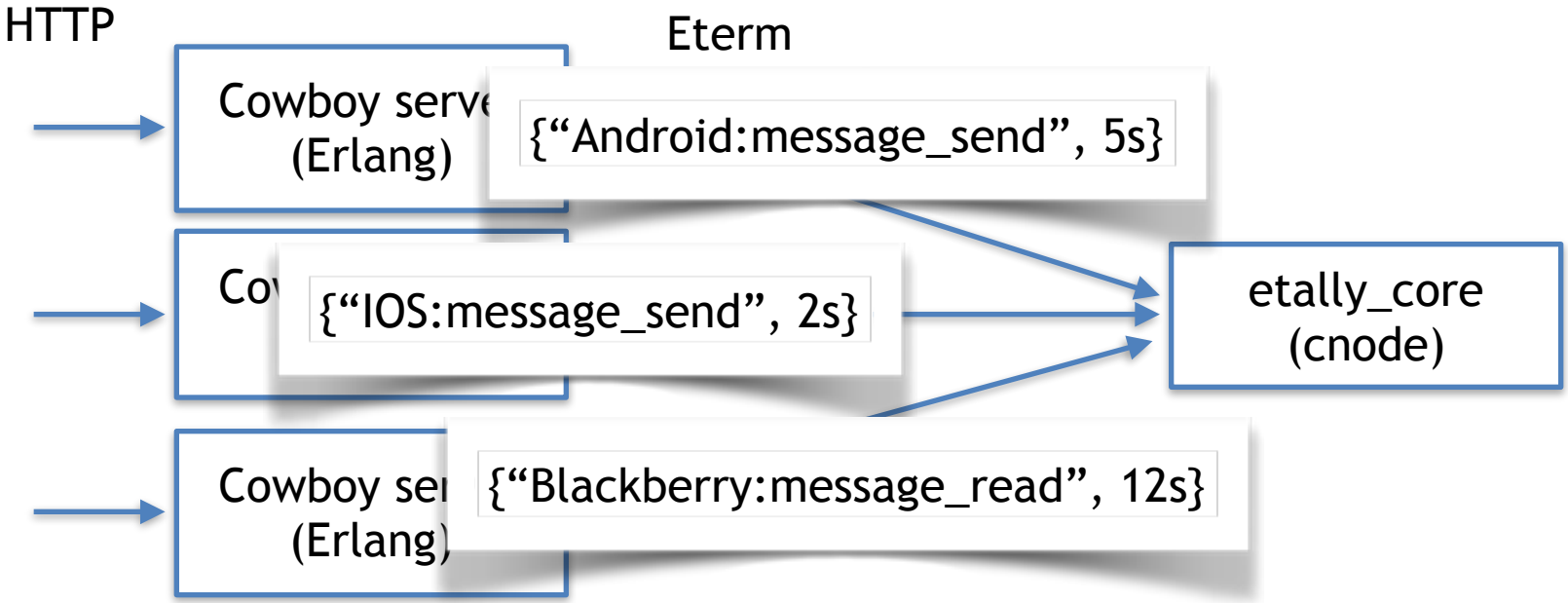
interface etally_metric:



```
etally_metric:get_counter(<<"event">>, Instance) ->

 {[{4838400,56000,5600000,560000000},
   {2419200,56000,5600000,560000000},
   {1209600,56000,5600000,560000000},
   {604800,56000,5600000,560000000},
   {518400,56000,5600000,560000000},
   {432000,56000,5600000,560000000},
   {345600,56000,5600000,560000000},
   {259200,56000,5600000,560000000},
   {172800,56000,5600000,560000000},
   {86400,56000,5600000,560000000},
   {7200,56000,5600000,560000000},
   {3600,56000,5600000,560000000},
   {300,56000,5600000,560000000}],
  [{0.9990000128746033,[{2419200,100},{604800,100},{86400,100},{3600,100}]},
   {0.9900000095367432,[{2419200,100},{604800,100},{86400,100},{3600,100}]},
   {0.949999988079071,[{2419200,100},{604800,100},{86400,100},{3600,100}]},
   {0.8999999761581421,[{2419200,100},{604800,100},{86400,100},{3600,100}]},
   {0.800000011920929,[{2419200,100},{604800,100},{86400,100},{3600,100}]},
   {0.75,[{2419200,100},{604800,100},{86400,100},{3600,100}]},
   {0.5,[{2419200,100},{604800,100},{86400,100},{3600,100}]}]}
```

A use case:


HTTP
Eterm
Cowboy serve
(Erlang)
{“Android:message_send”, 5s}
Co
{“IOS:message_send”, 2s}
etally_core
(cnode)
Cowboy ser
(Erlang)
{“Blackberry:message_read”, 12s}

A use case:

```
etally_metric:send_event([<<"IOS">>, <<"IOS-Version">>],
                         [{<<"IOS">>, <<"clients-leaderboard">>},
                          {<<"IOS-version">>, <<"IOS-clients-leaderboard">>}],
                         [<<"IOS">>, <<"IOS-version">>],
                         100, Instance),
```

Trend Dashboard

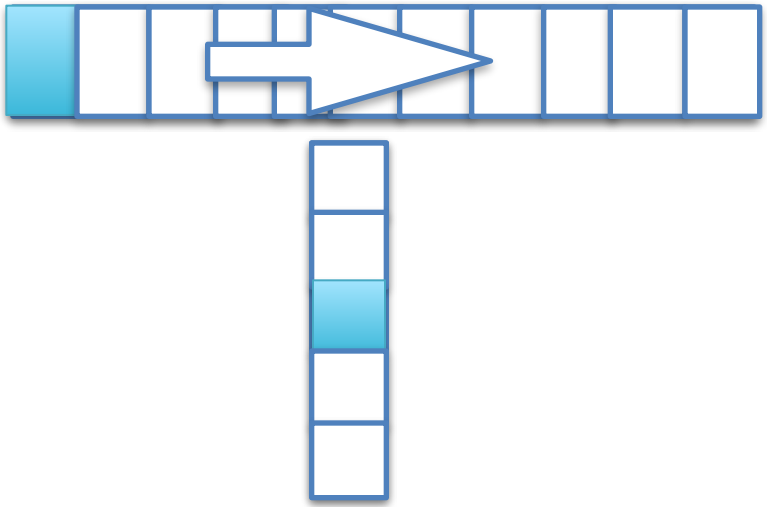
Devices | Versions

	Token		Day	Week	Month
0	Apple iOS_4.5.9 Apple iOS_4.5.9	Info	-5% ↓ 5s 14s	136.82K -66% ↓ 5s 16s	1.3M – ↑ 4s 17s
1	Apple iOS_4.5.55 Apple iOS_4.5.55	Info	17% ↑ 4s 17s	325.52K 377% ↑ 5s 18s	394.86K – ↑ 5s 19s
2	Apple iOS_3.1.412 Apple iOS_3.1.412	Info	K 3% ↑ 22s 32s	76.64K 2% ↑ 21s 33s	260.33K – ↑ 21s 33s
3	Apple iOS_4.2.31 Apple iOS_4.2.31	Info	16% ↑ 3s 17s	20.57K -0% ↓ 4s 16s	91.03K – ↑ 4s 15s
4	Apple iOS_3.1.411 Apple iOS_3.1.411	Info	12% ↑ 18s 30s	6.4K 3% ↑ 19s 32s	22.97K – ↑ 20s 34s
5	Apple iOS_4.1.131 Apple iOS_4.1.131	Info	0 3% ↑ 4s 14s	4.8K -22% ↓ 4s 8s	20.59K – ↑ 3s 10s

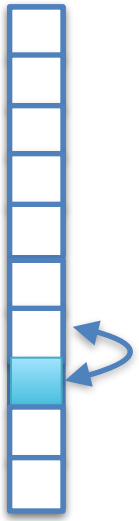
← Previous

Next →

Future work on scalability

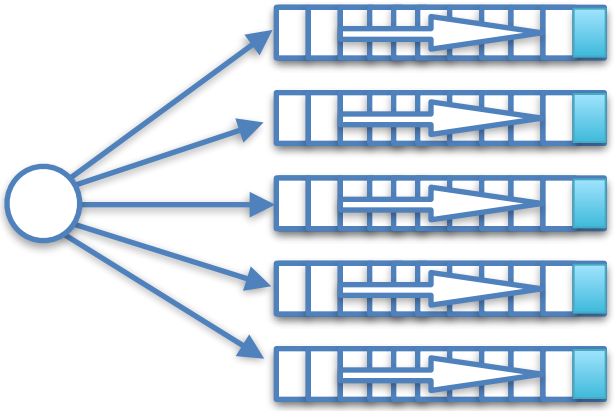


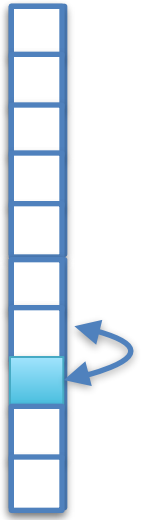
Counter



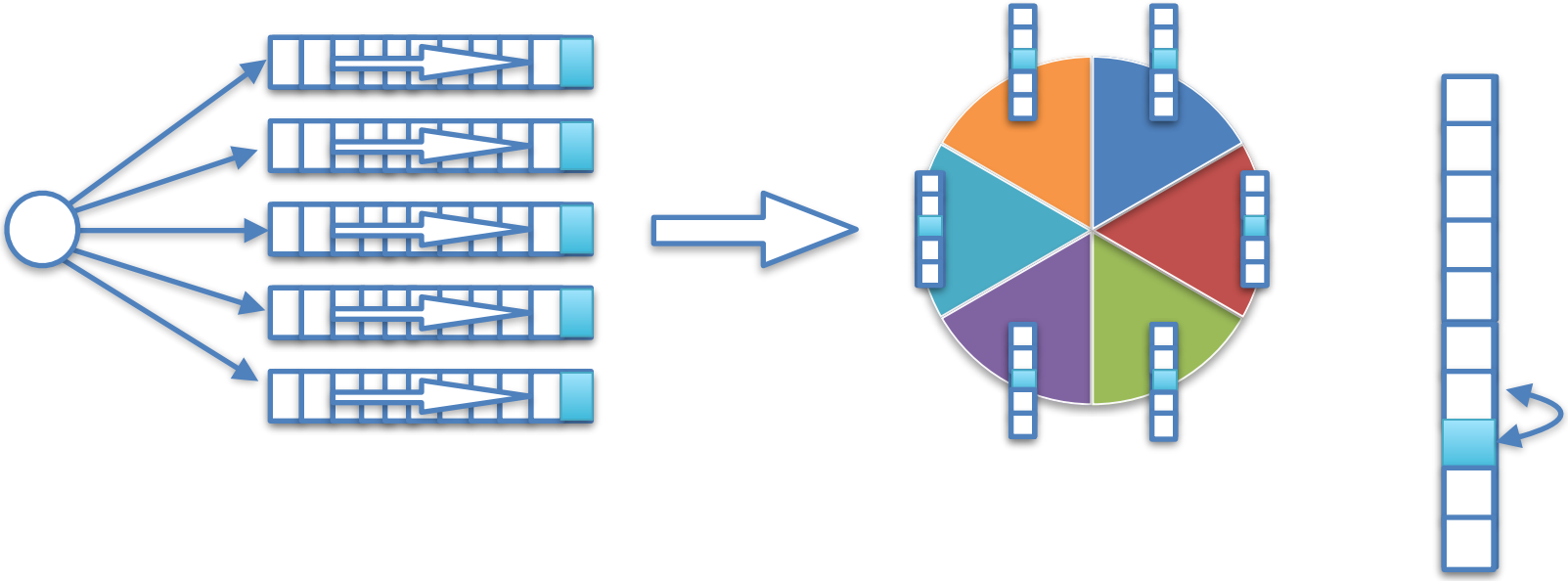
Leaderboard

Future work on scalability

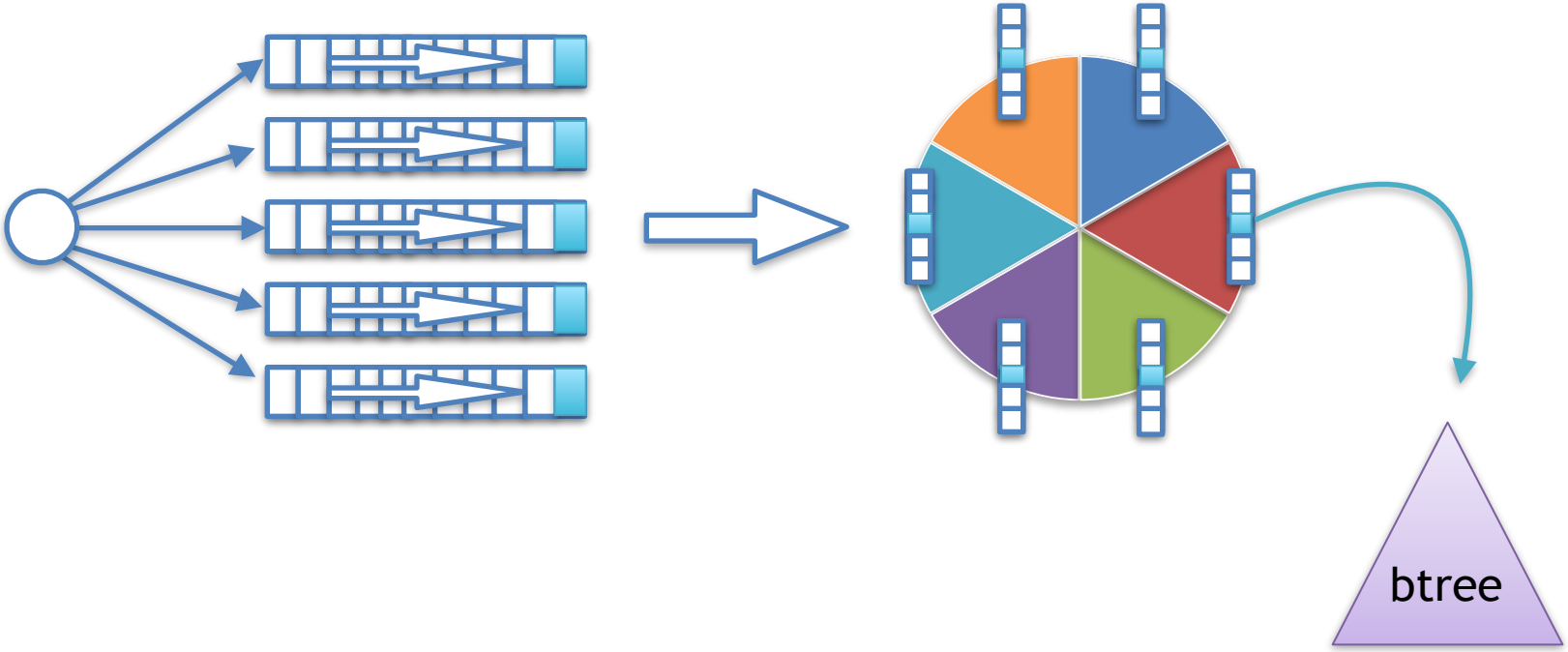




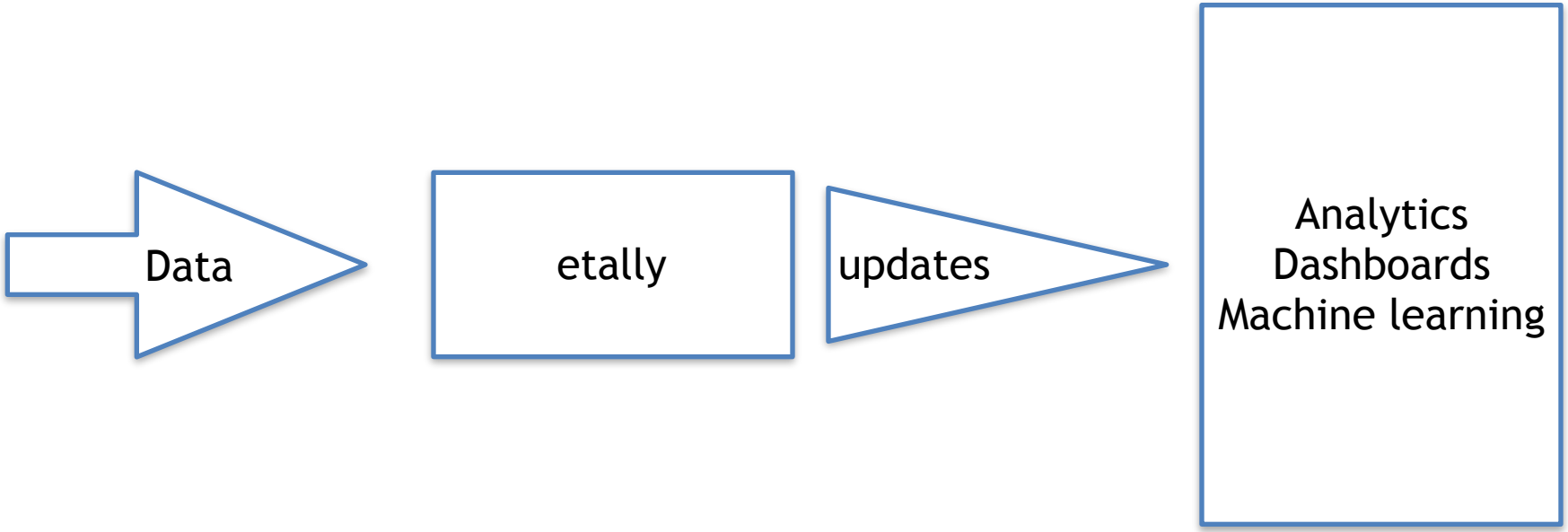
Future work on scalability

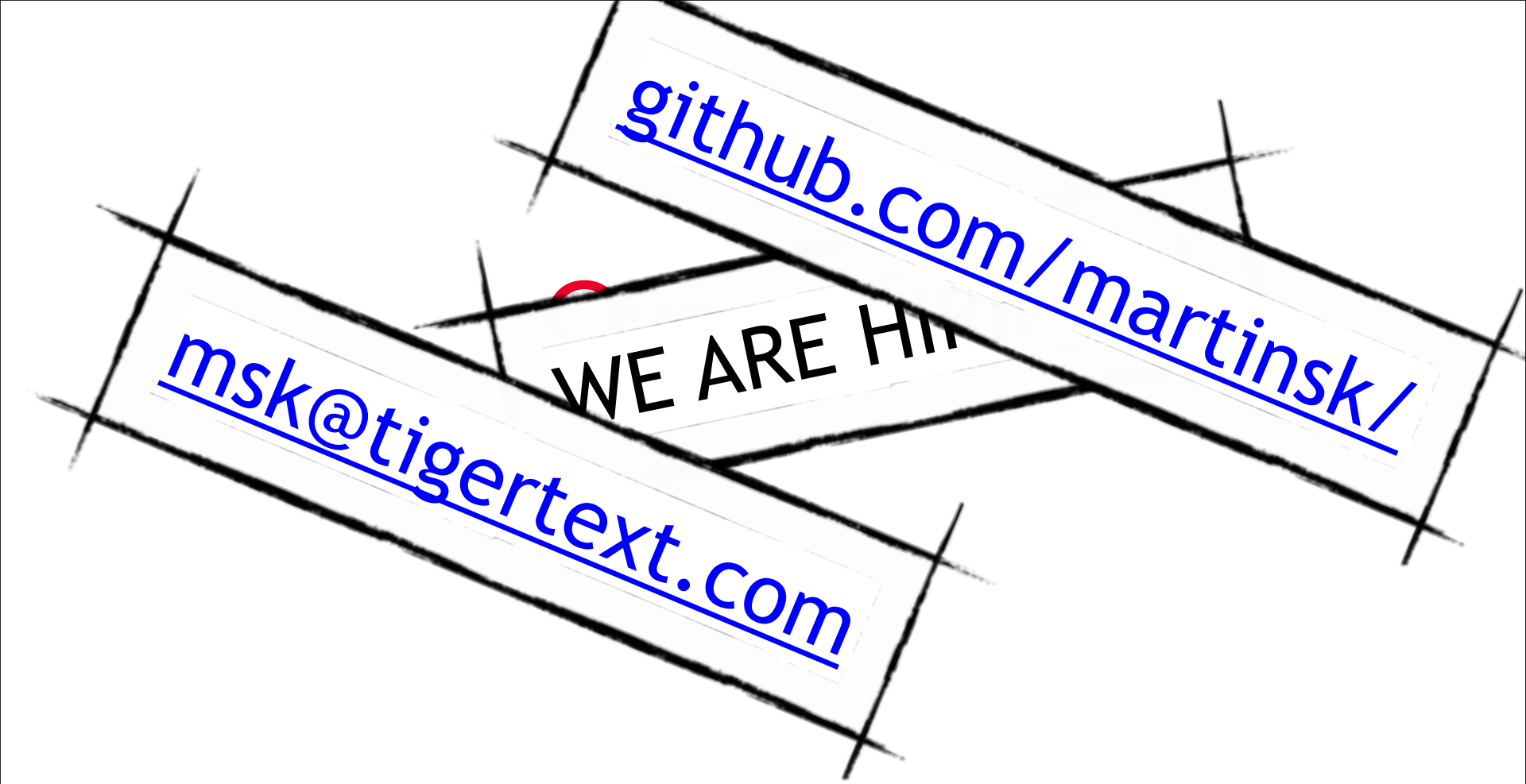


Future work on scalability



Building on top of eTally




github.com/martinsk/
WE ARE HI
msk@tigertext.com