MZ
MachineZone

About the Speaker

- Fredrik Linder
- 15+ years as a dev (Erlang/C++ mostly)
- MachineZone, Klarna, Ericsson, …
- M.Sc at University of Stockholm

MACHINE ZONE

Machine Zone

2015-06-11

MachineZone

We’re a Tech Company

We’re a Tech Company that Builds Games

We build top grossing games

- Original Gangstaz
- iMob
- iKnight
- JetFighters
- ...
- and our most recent success:

MachineZone

Game of War

World's largest
single-world interactive game

MachineZone

Game of War

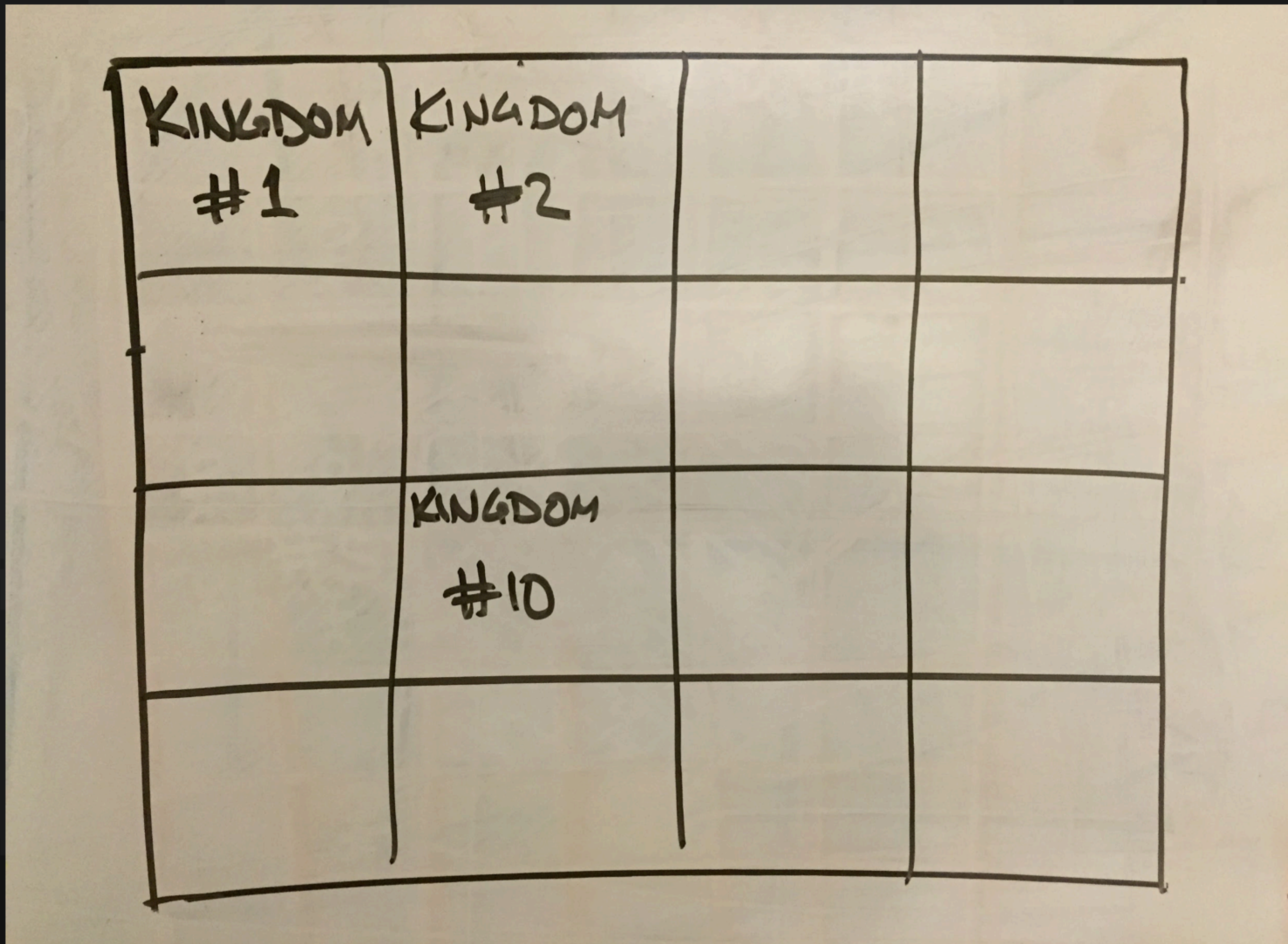
MachineZone

Game of War

- Build your city
- Build your army
- Level up your hero
- Equip hero with gear
- Fight monsters
- Attack your enemies
- Gather nearby resources
- Research new technologies
- ...

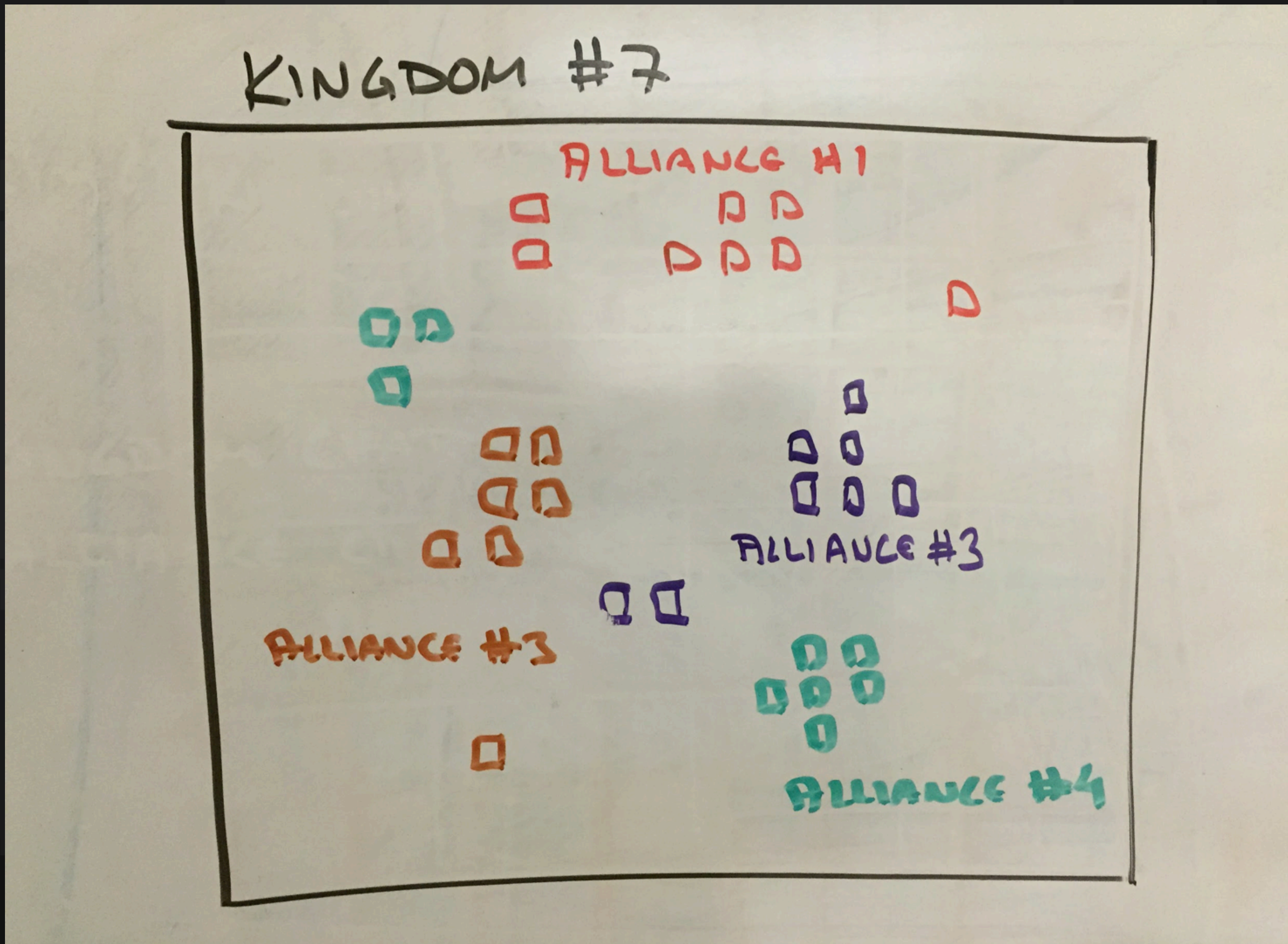
MachineZone

GoW needs a map



MACHINE ZONE

GoW needs a map



MachineZone

Game of War

- Join an alliance
- Start a war
- Defend your alliance
- Discuss plans
- Get/give advice on how to play
- Meet people
- ...

MACHINEZONE

GoW is a social game

GoW needs a chat with translations

- Translation of chat messages
- Language detection
- Text tokenization
- Chat speak: lol, mdr etc

MachineZone

Game vs Tech

Game side

- Game design
- Game logic
- User interaction
- User retention
- ...

Tech side

- Map service
- Chat service
- Translation service
- ...

MachineZone

- Map service
- Chat service

MachineZone

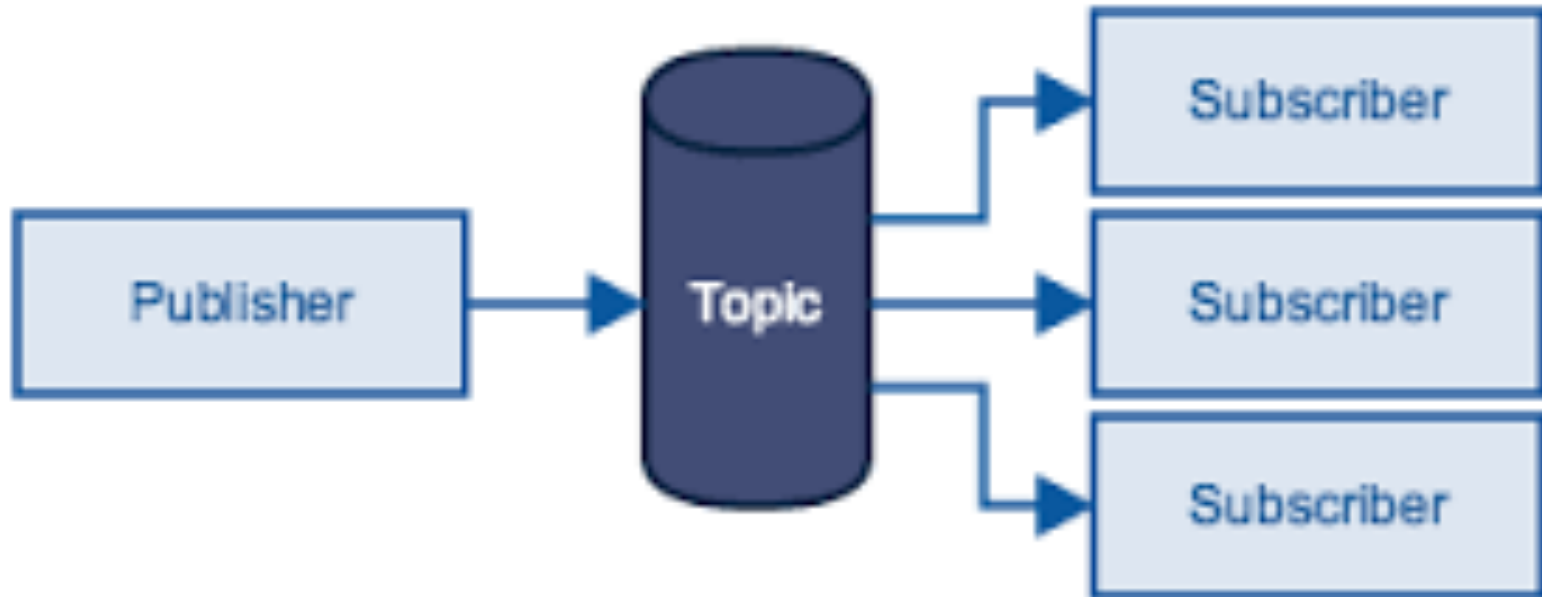
PubSub

- Map service
- Chat service

- … are both pubsub services

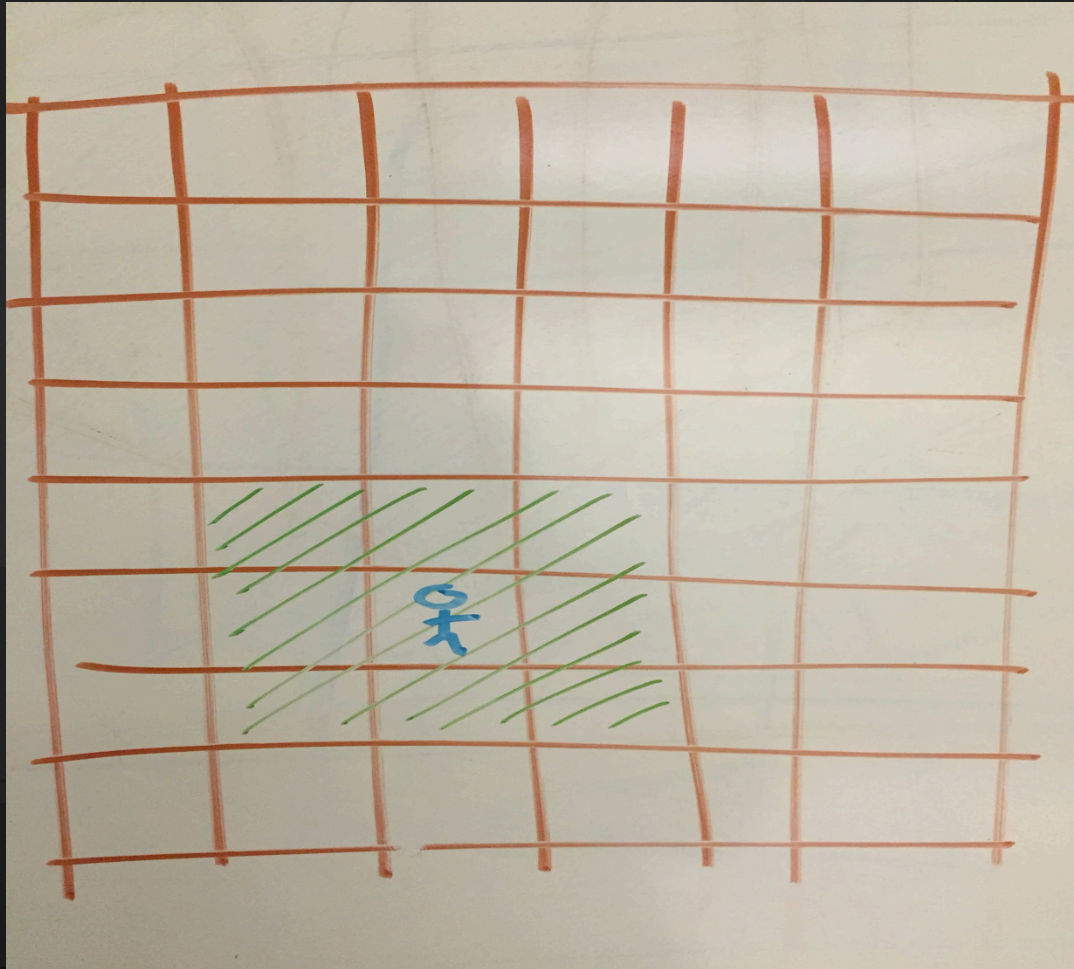
MachineZone

PubSub



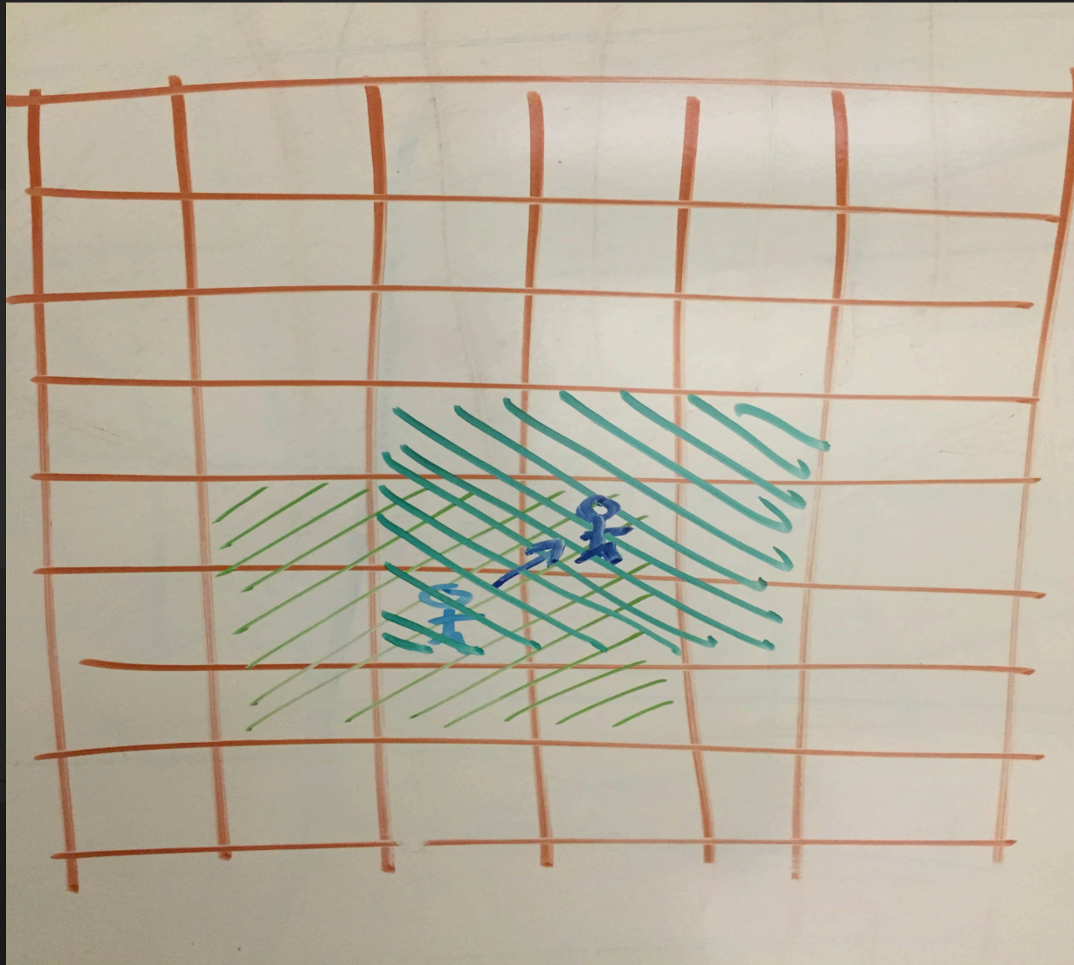
MACHINEZONE

Map



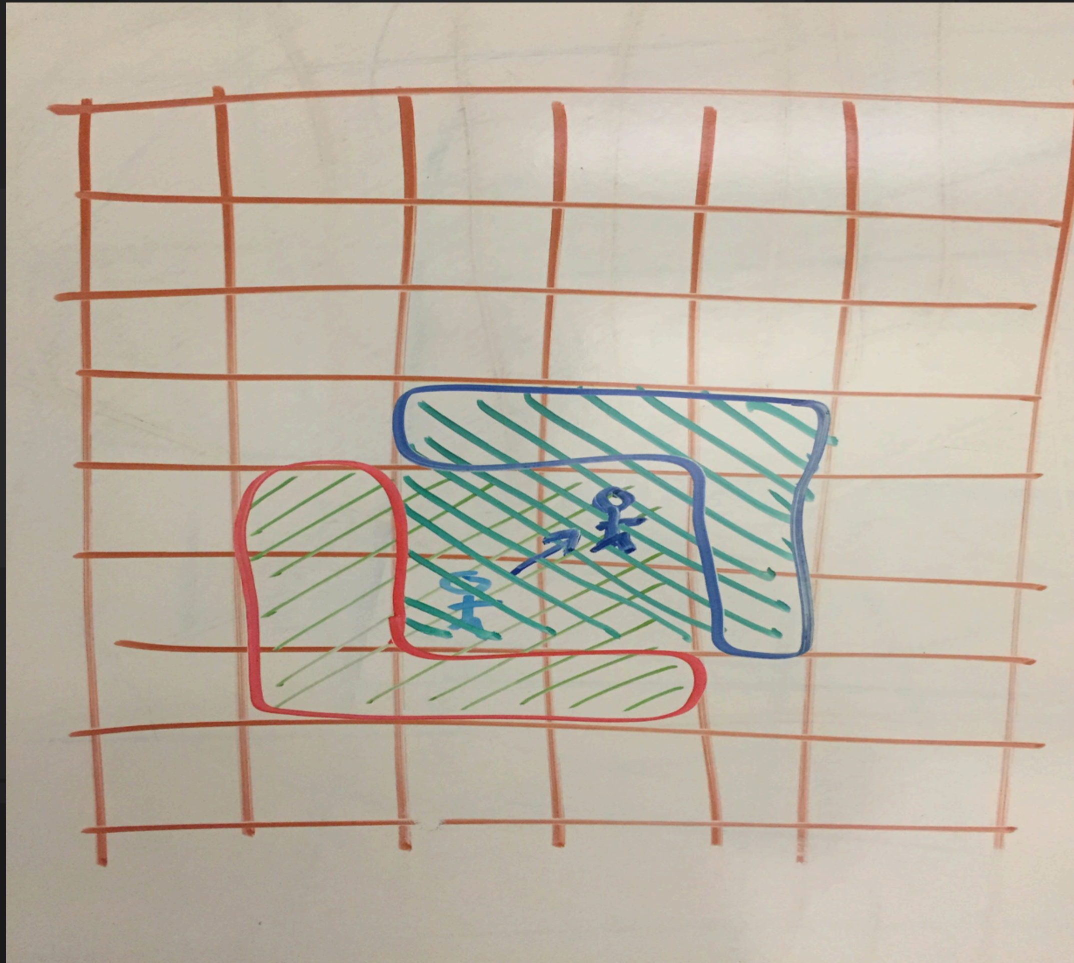
MachineZone

Map



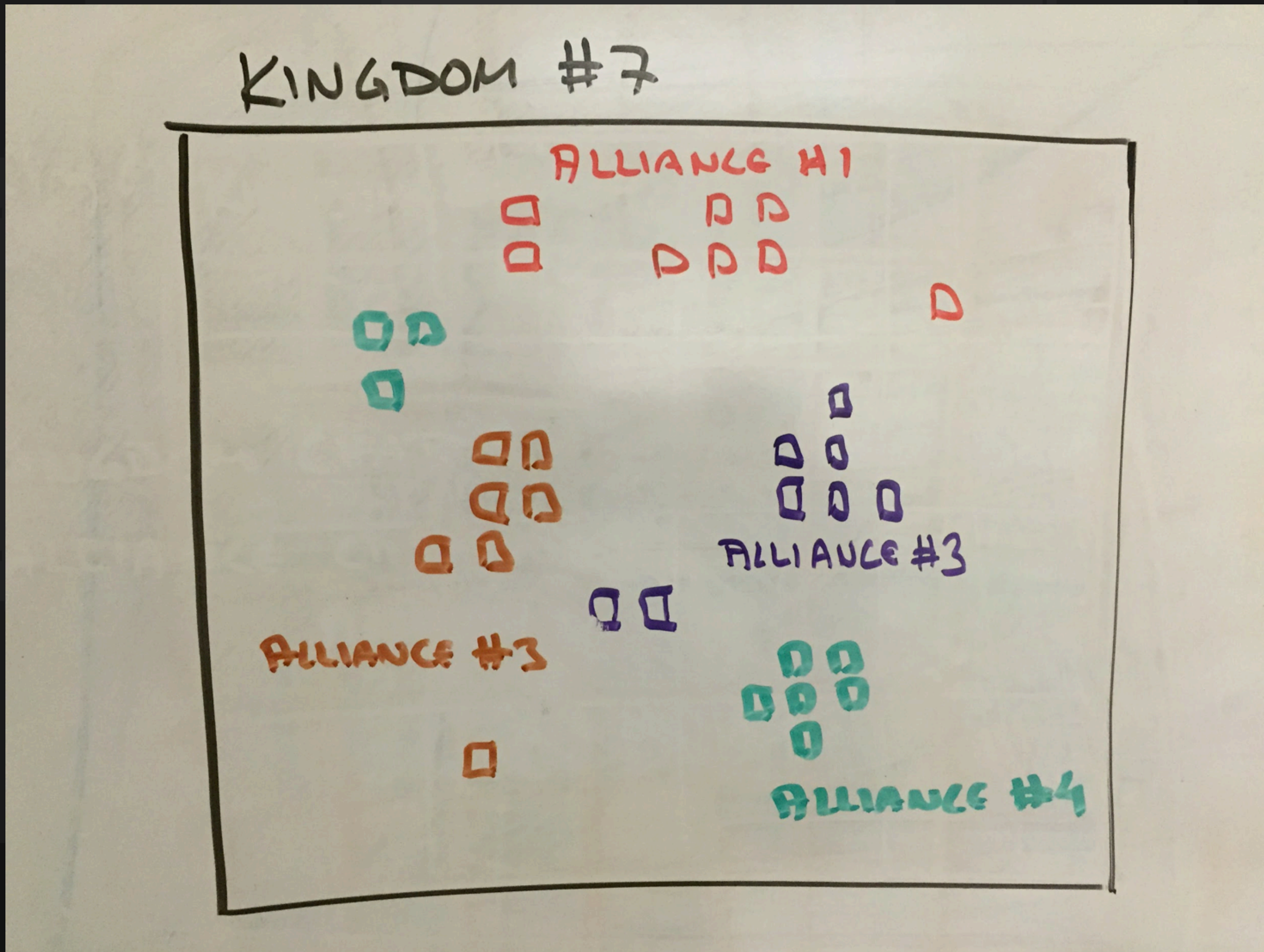
MachineZone

Map



MachineZone

Chat



MachineZone

Current GoW pubsub

- More mps than Twitter
- Sub-second delivery latency

MachineZone

Challenge

- *We want (a lot) more features that depends on a fast and reliable pubsub*
  /Game Team

MachineZone

Challenge

- *We want (a lot) more features that depends on a fast and reliable pubsub*
  /Game Team

- How can we make it 10x / 100x / 1000x?

MACHINE ZONE

Performance

- Throughput
- Latency

MACHINEZONE

Performance

- Throughput
  - From Little's law:
    throughput = # msgs / queuing delay
- Latency
  - latency = processing time + queuing delay

MachineZone

Performance

- Throughput
  - From Little's law:
    throughput = # msgs / queuing delay
  - ***Maximize # msgs***
  - ***Minimize queuing delay***
- Latency
  - latency = processing time + queuing delay
  - ***Minimize processing time***

MachineZone

Maximize # msgs

- Do things in parallel
  - Bound by # cores
  - Bound by ordering guarantees
- Batch msgs
  - Route 1+ batched msgs ≈ route 1 msg

MACHINEZONE

Minimize queuing delay

- TCP
  - Ideally: read data faster than it is being written
  - Big enough `{buffer, pos_integer()}`
  - Fast enough parsing (NIF) and dispatching
- Inbox
  - Ideally: empty inbox when being scheduled out
  - < 2000 reductions per msg
  - Minimize # active processes (up to # cores)
  - Look out for large inboxes
- Avoid synchronous (blocking) calls

MZ
MachineZone

Minimize processing time

- Use the right algorithms
- Use the right data structure
- Use the right language constructs
- Avoid generating garbage

MACHINE ZONE

New PubSub

Current

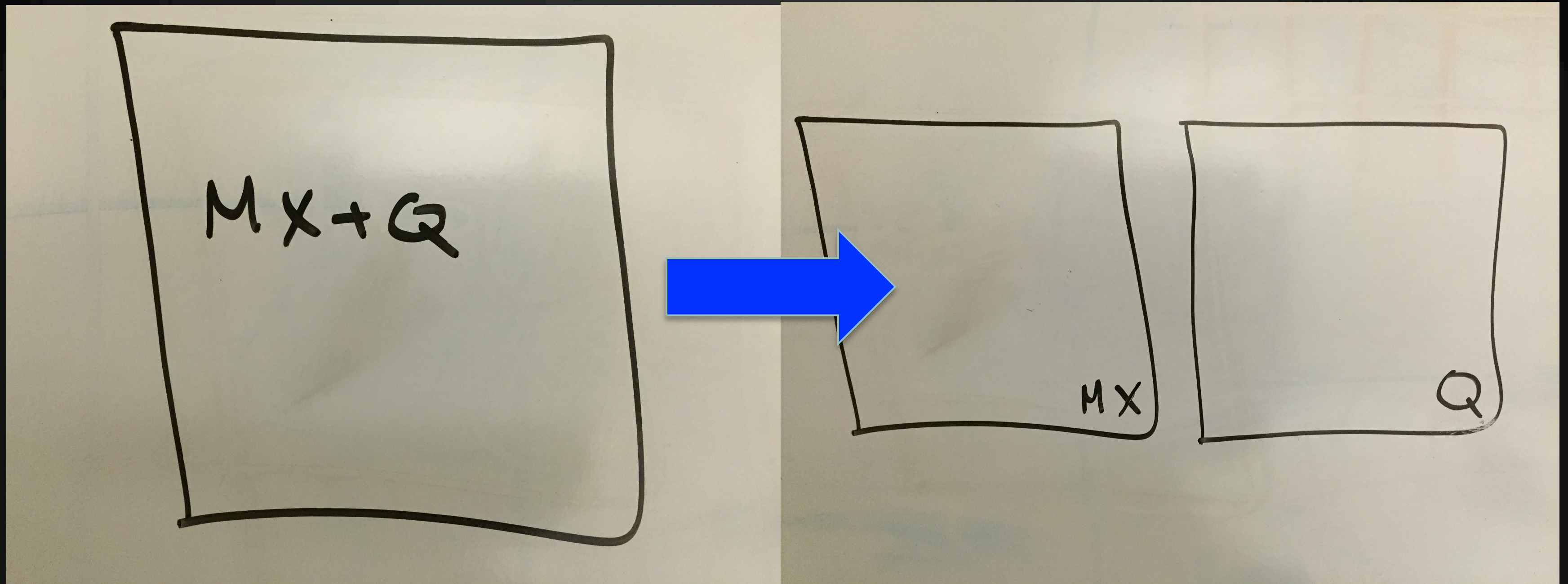
- Ejabberd-based
- Single node type
- Fully connected mesh
- XMPP

New

- In-house
- Multiple node types
- N:M-like connected mesh
- Json

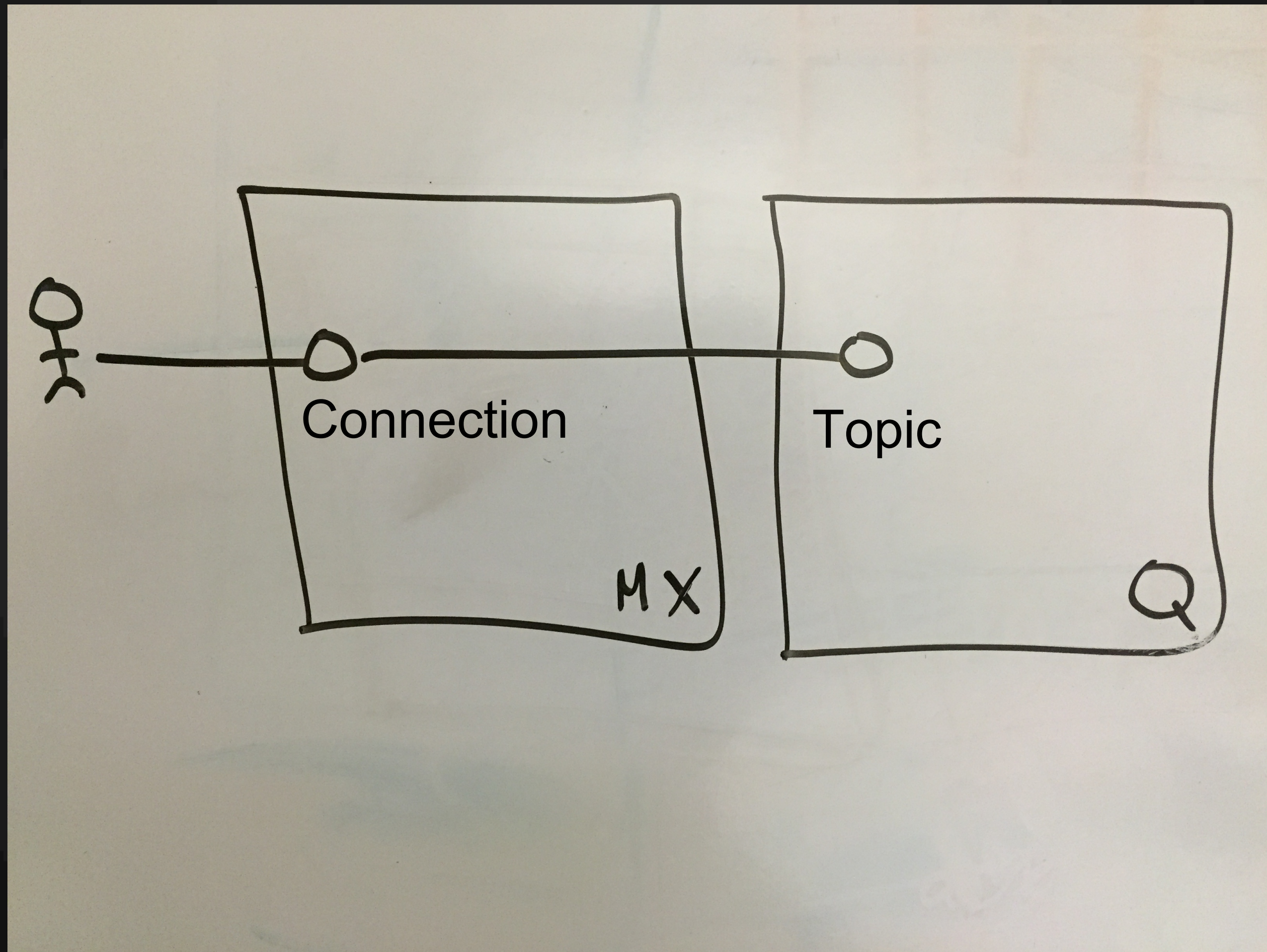
MACHINE ZONE

Split nodes



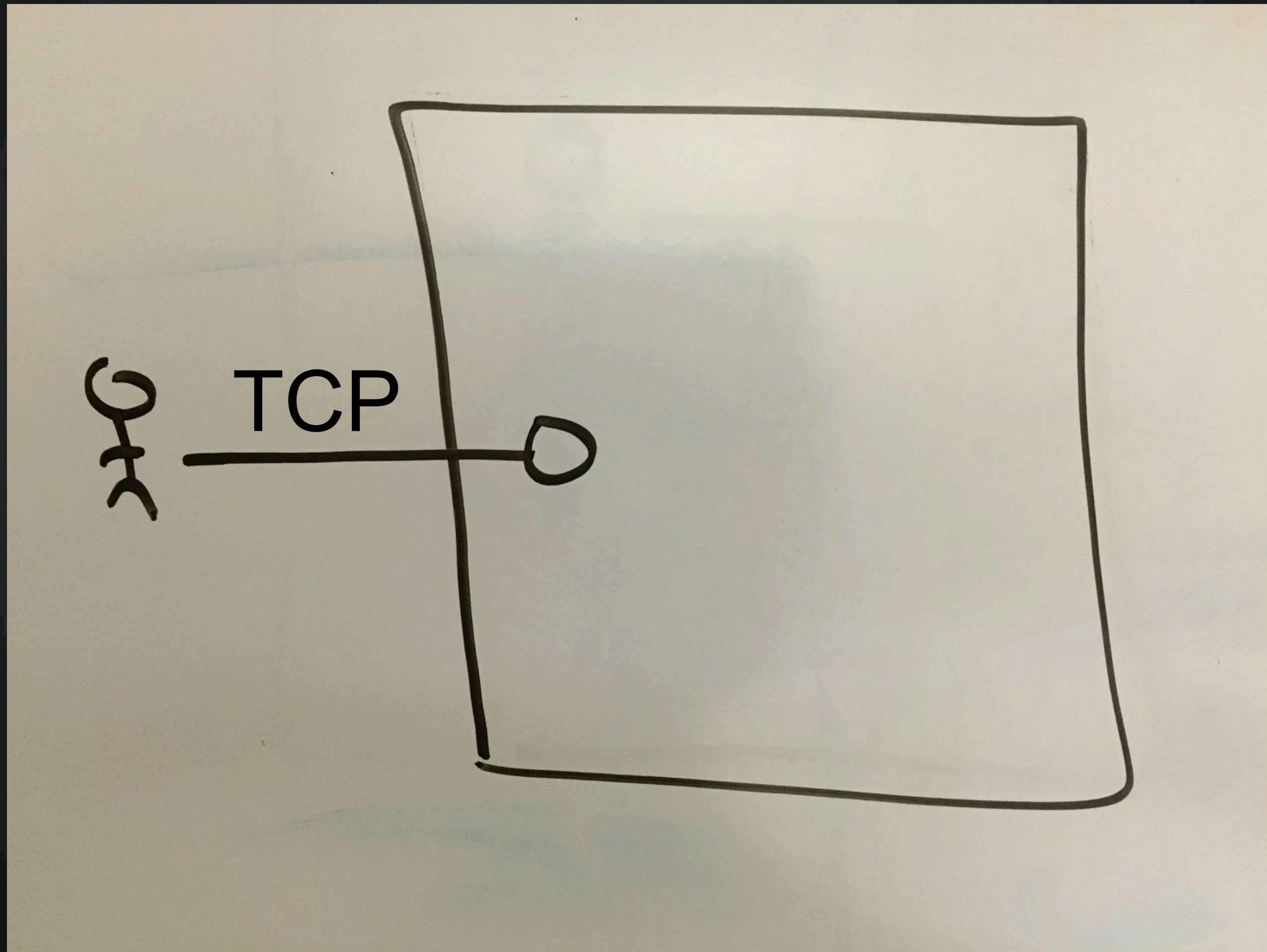
MZ
MachineZone

Data path



MachineZone

Client – Mx



MACHINE ZONE

Active or passive mode?

- `{active, false}`
- `{active, once}`
- `{active, pos_integer()}`

MachineZone

Active or passive mode?

- `{active, false}`
  - Polling from socket
  - Cannot do anything else while waiting for data
- `{active, once}`
- `{active, pos_integer()}`

MachineZone

Active or passive mode?

- `{active, false}`
  - Polling from socket
  - Cannot do anything else while waiting for data
- `{active, once}`
  - Faster than `{active, false}`
  - Bonus: offers clean way to apply backpressure
- `{active, pos_integer()}`

MachineZone

Active or passive mode?

- `{active, false}`
  - Polling from socket
  - Cannot do anything else while waiting for data
- `{active, once}`
  - Faster than `{active, false}`
  - Bonus: offers clean way to apply backpressure
- `{active, pos_integer()}`
  - Surprise, surprise – slower than {active, once}

MZ
MachineZone

Active or passive mode?

- Still not good enough throughput…

MachineZone

Maximize # msgs

- `{buffer, Size :: pos_integer()}`

MachineZone

Maximize # msgs

- `{buffer, Size :: pos_integer()}`
- Bigger `buffer` => longer queuing delay

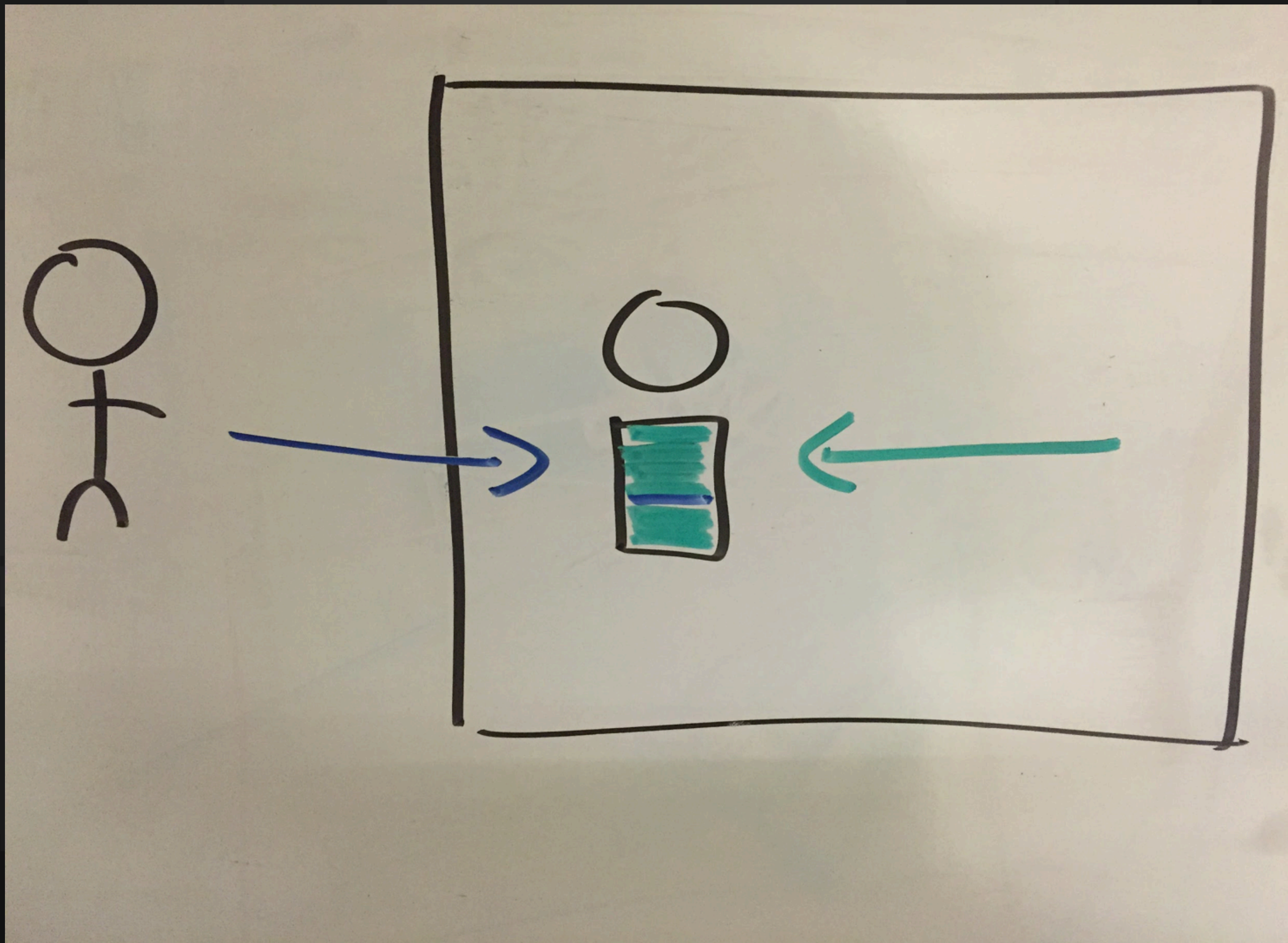
MachineZone

A closer look

- Connection process
  - receives inbound traffic from client
  - parses input
  - forwards requests
  - receives outbound traffic from topic
  - sends outbound traffic to client

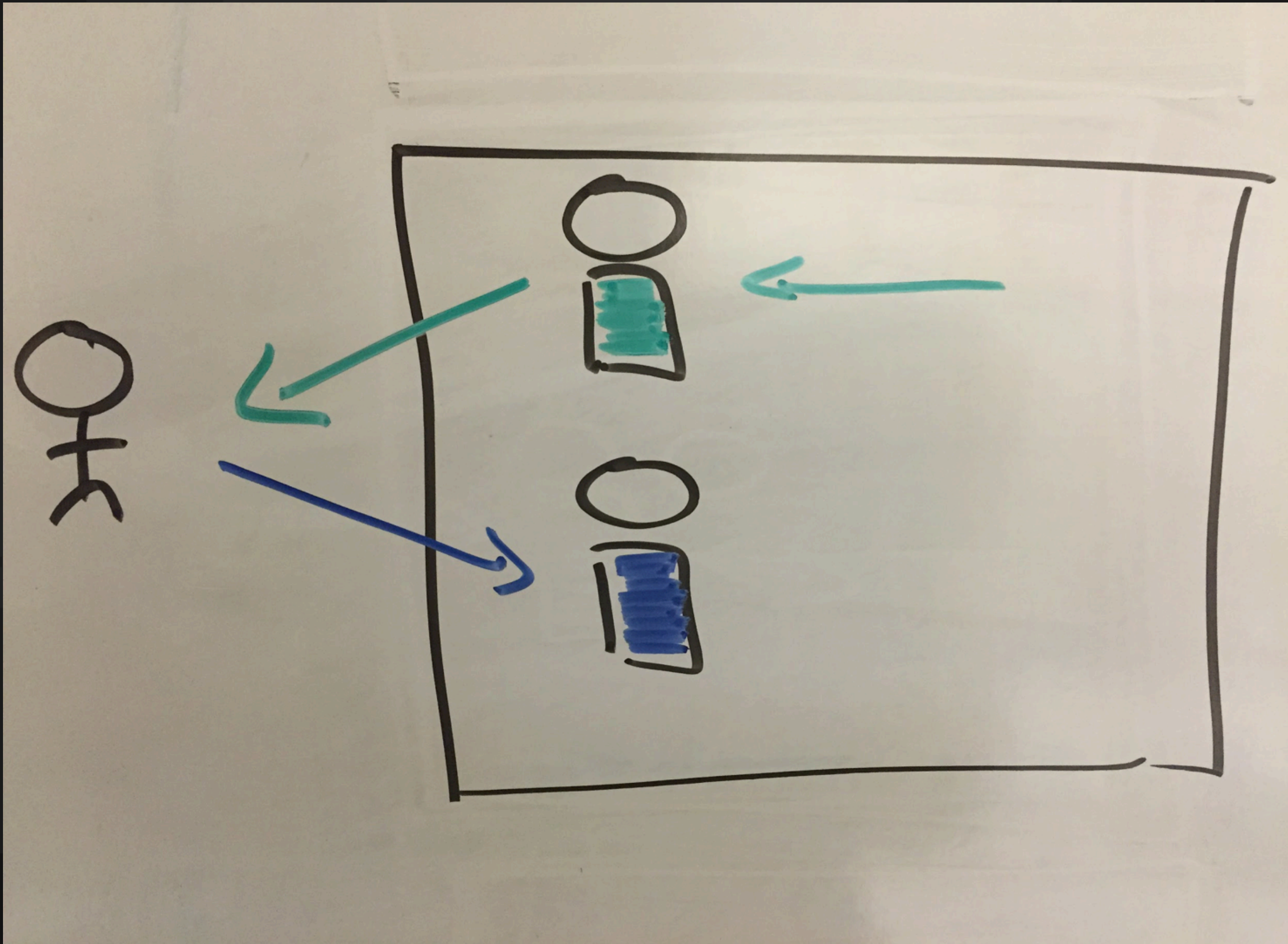
MACHINEZONE

A closer look



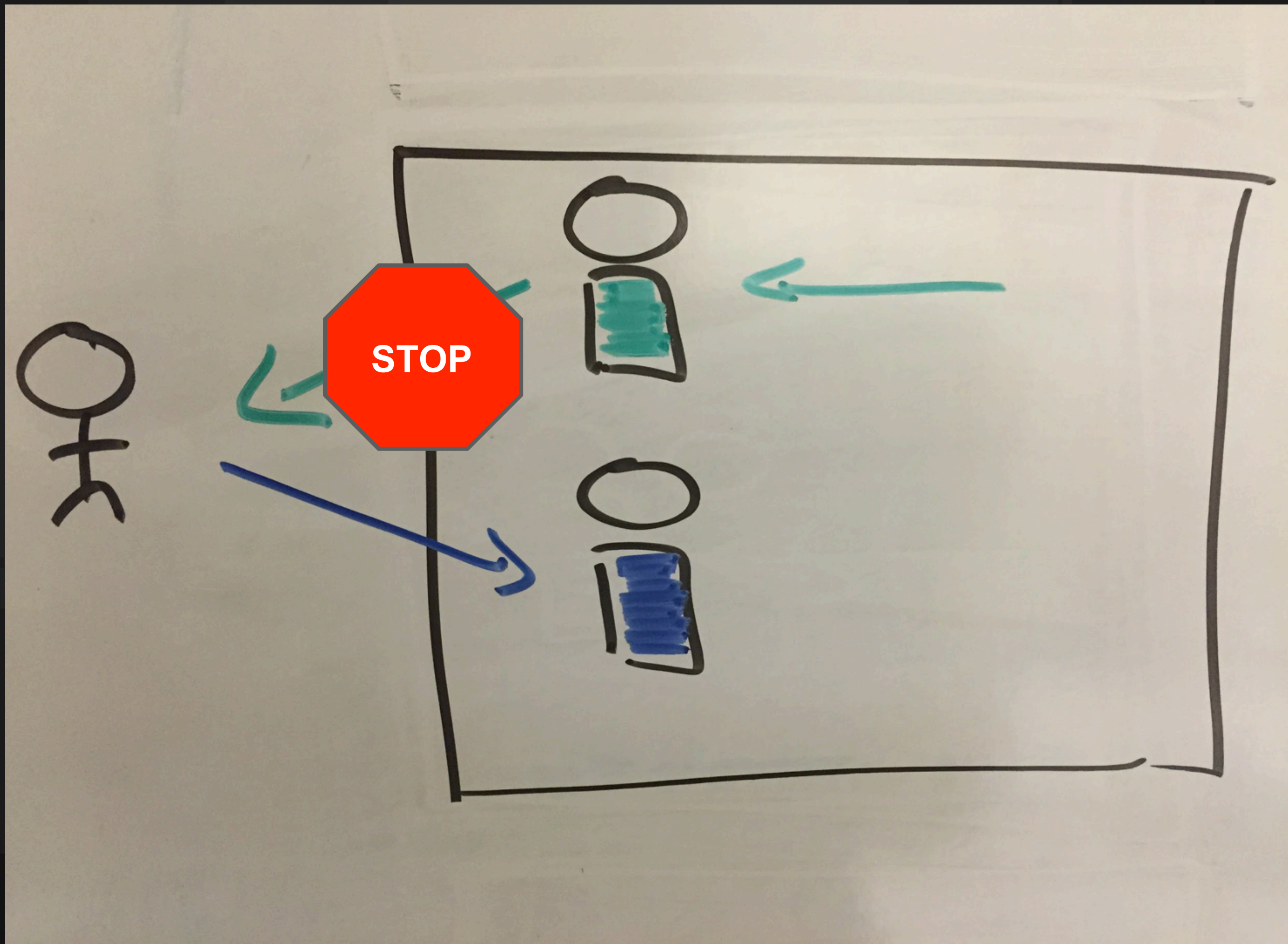
MachineZone

Minimize queuing delay

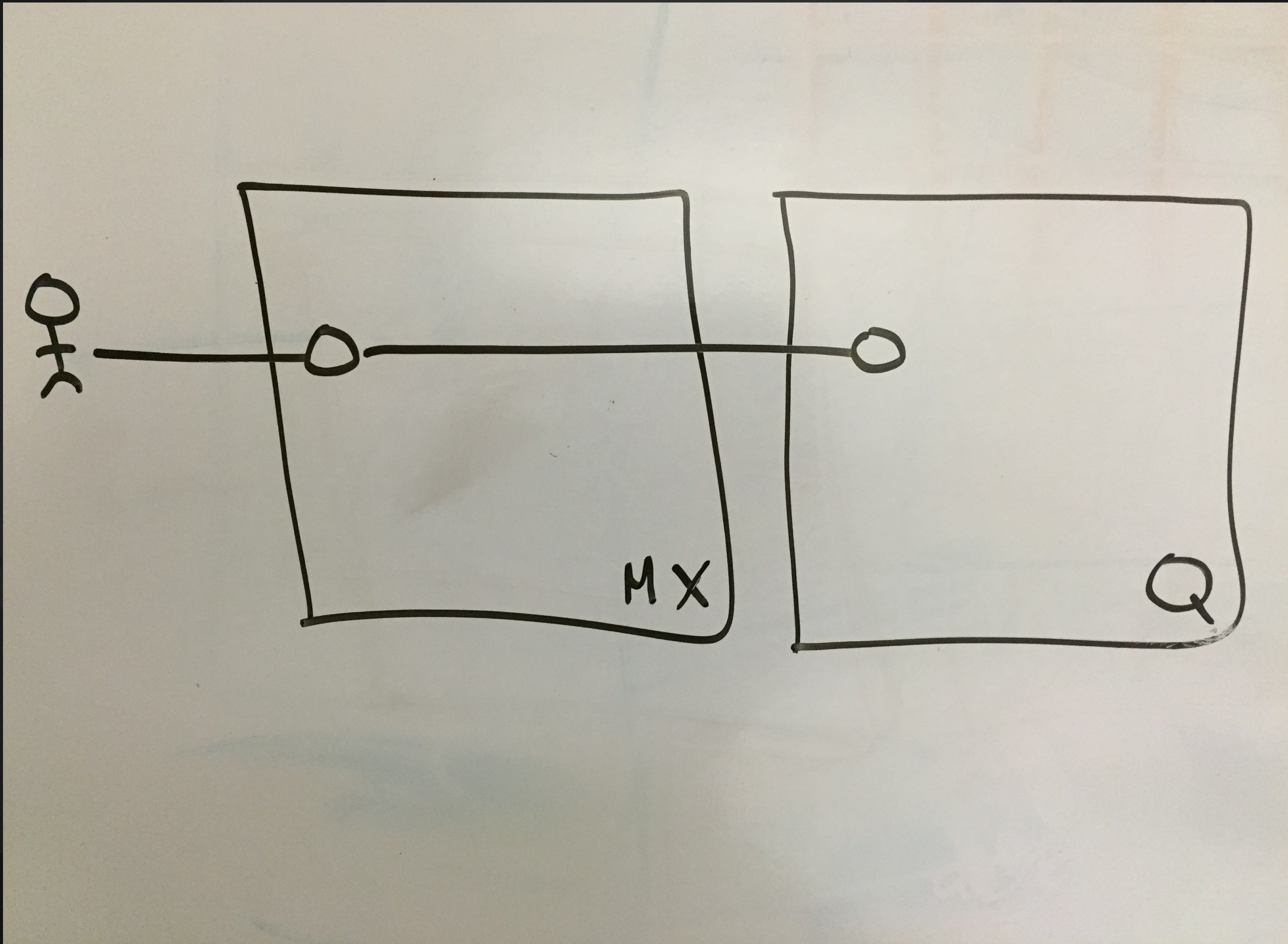


MachineZone

Radio shadow

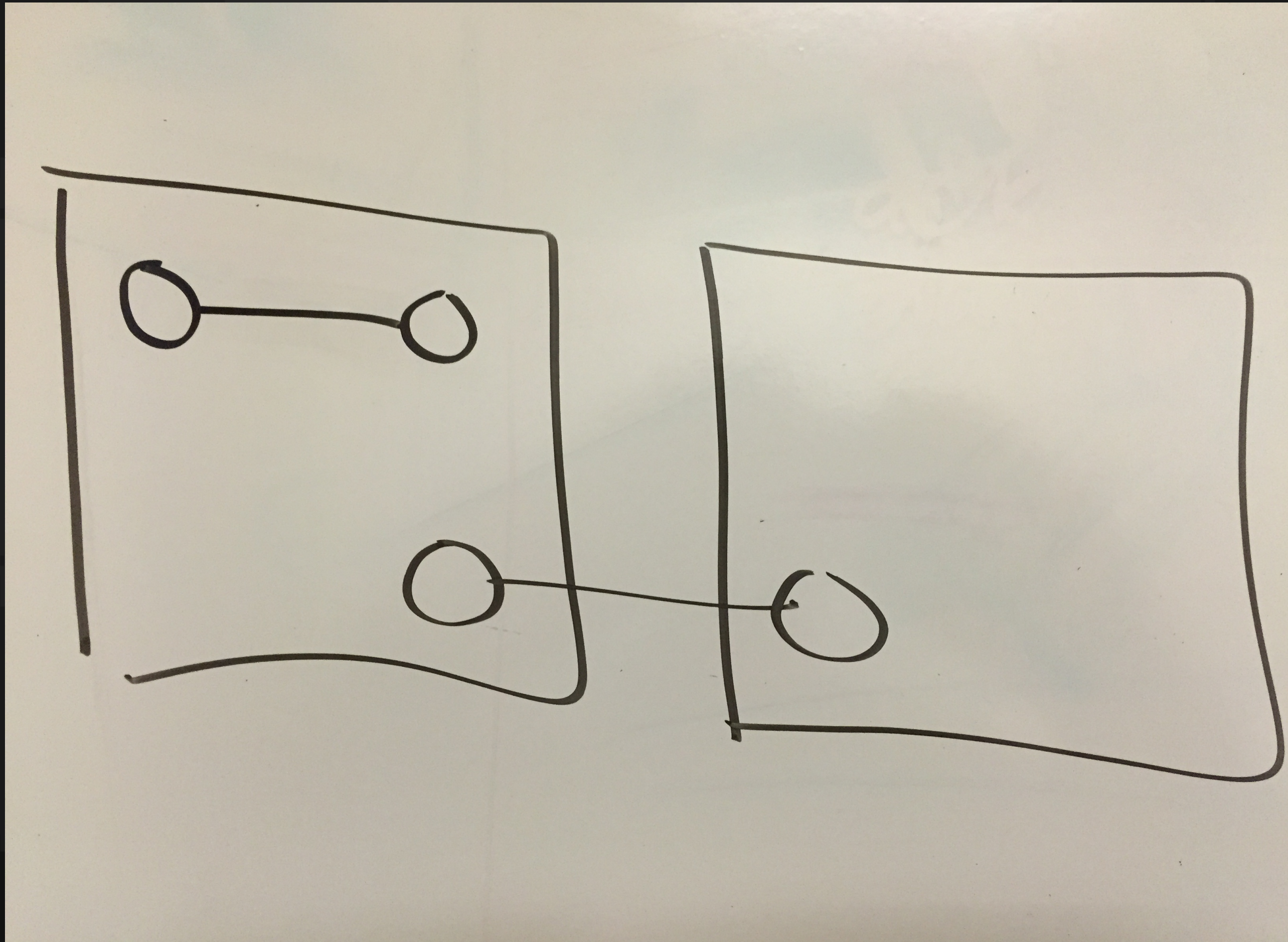


Mx – Q



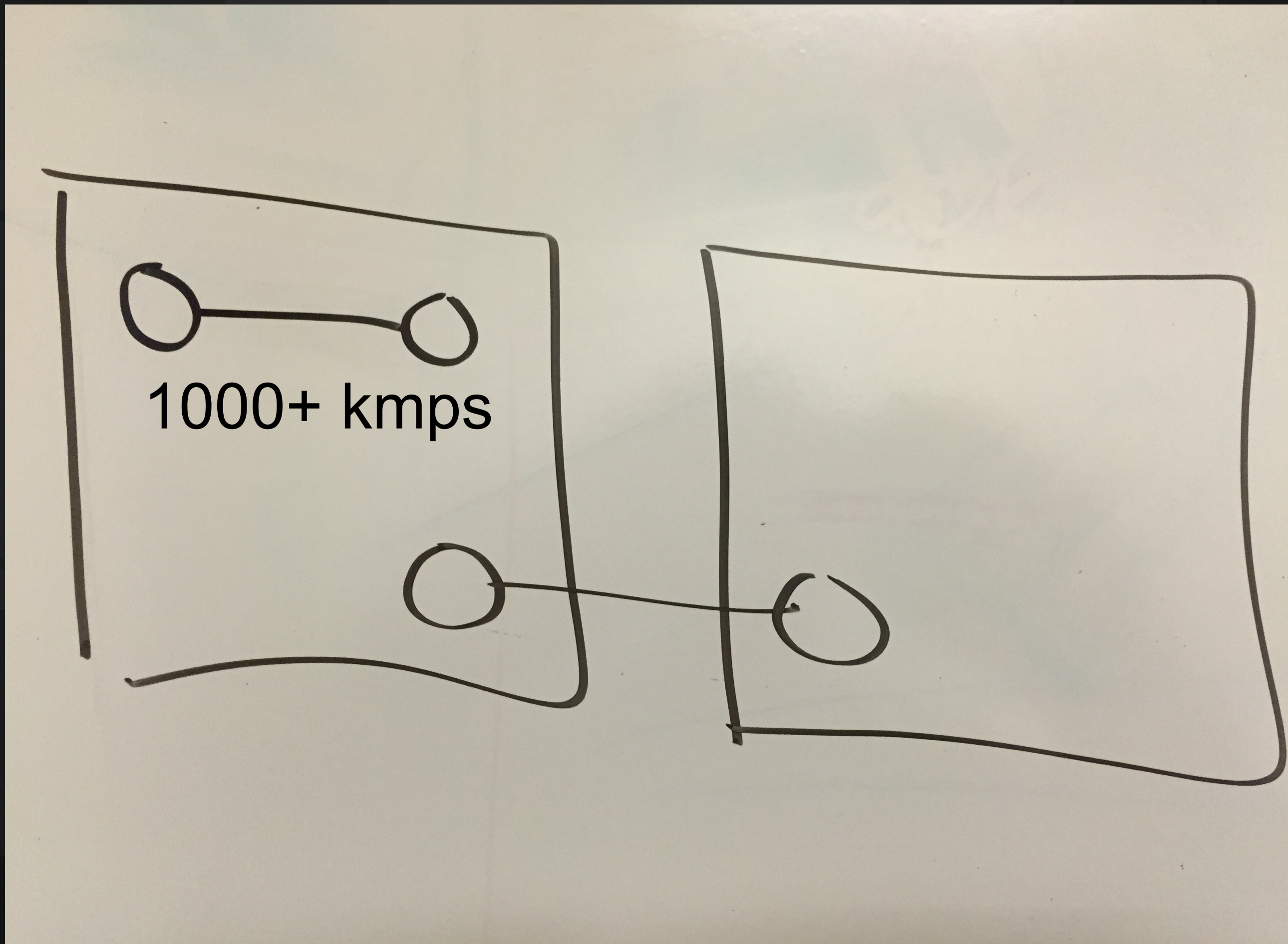
MachineZone

Message passing

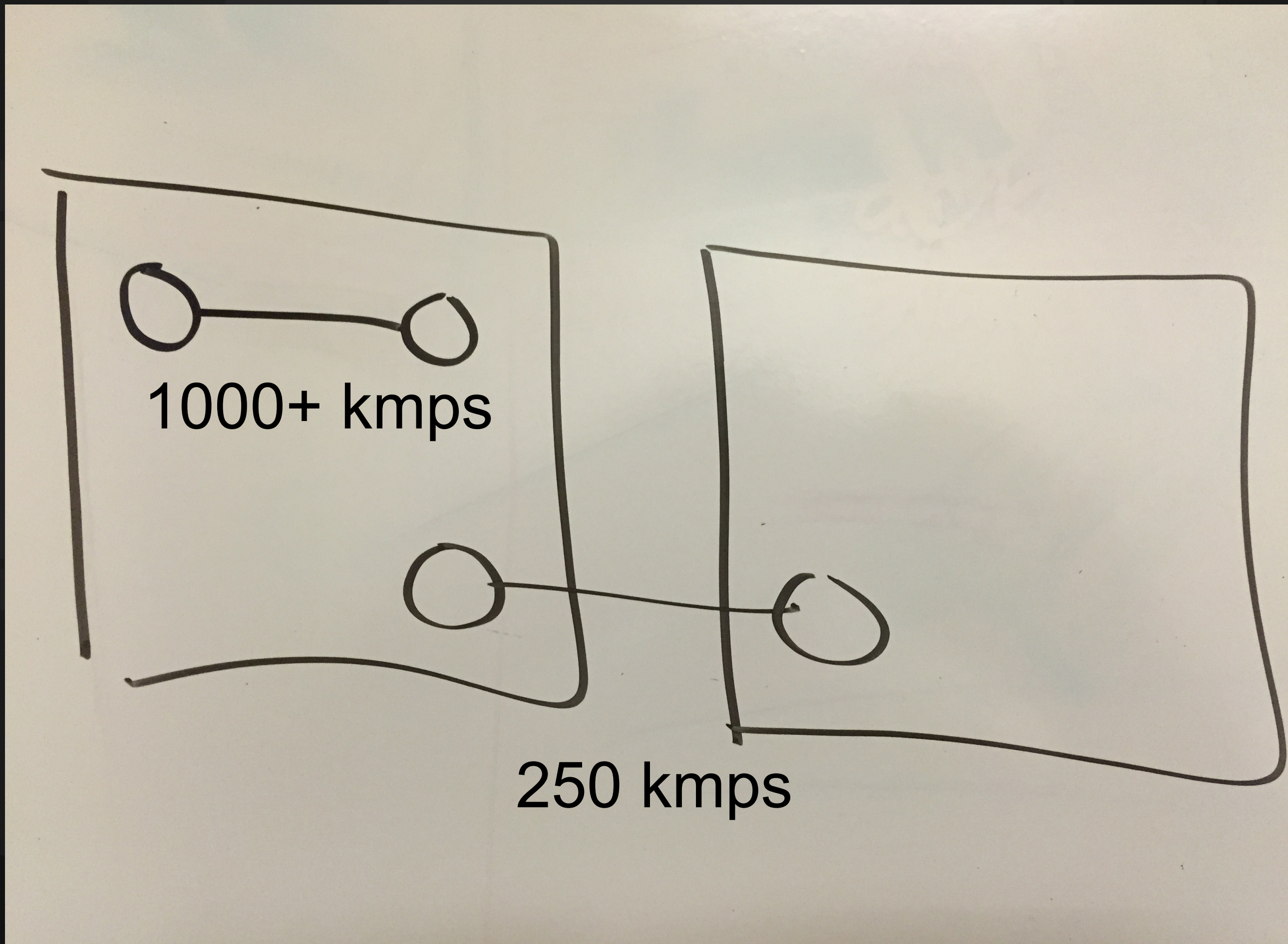


MachineZone

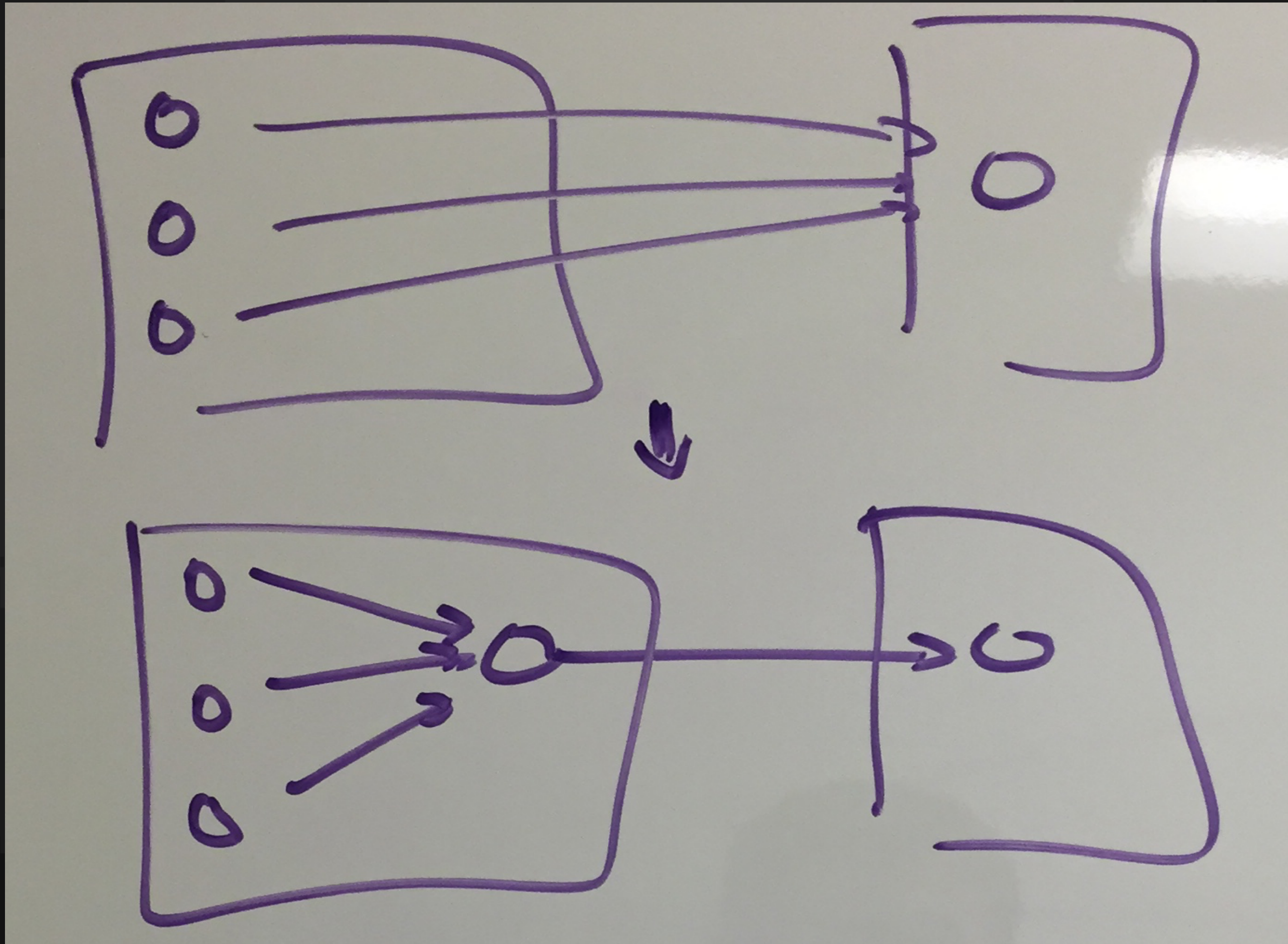
Message passing



Message passing

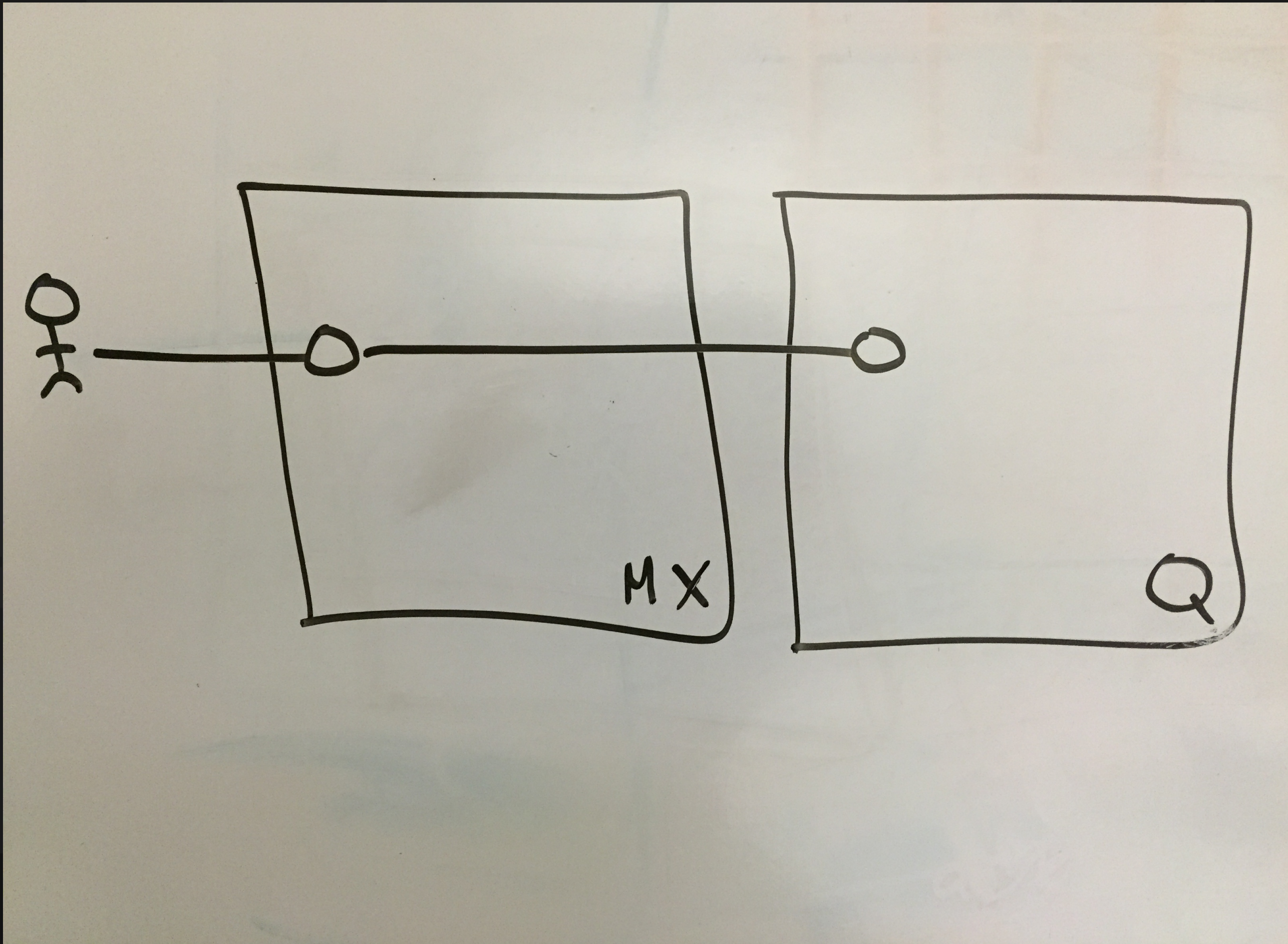


Maximize # msgs



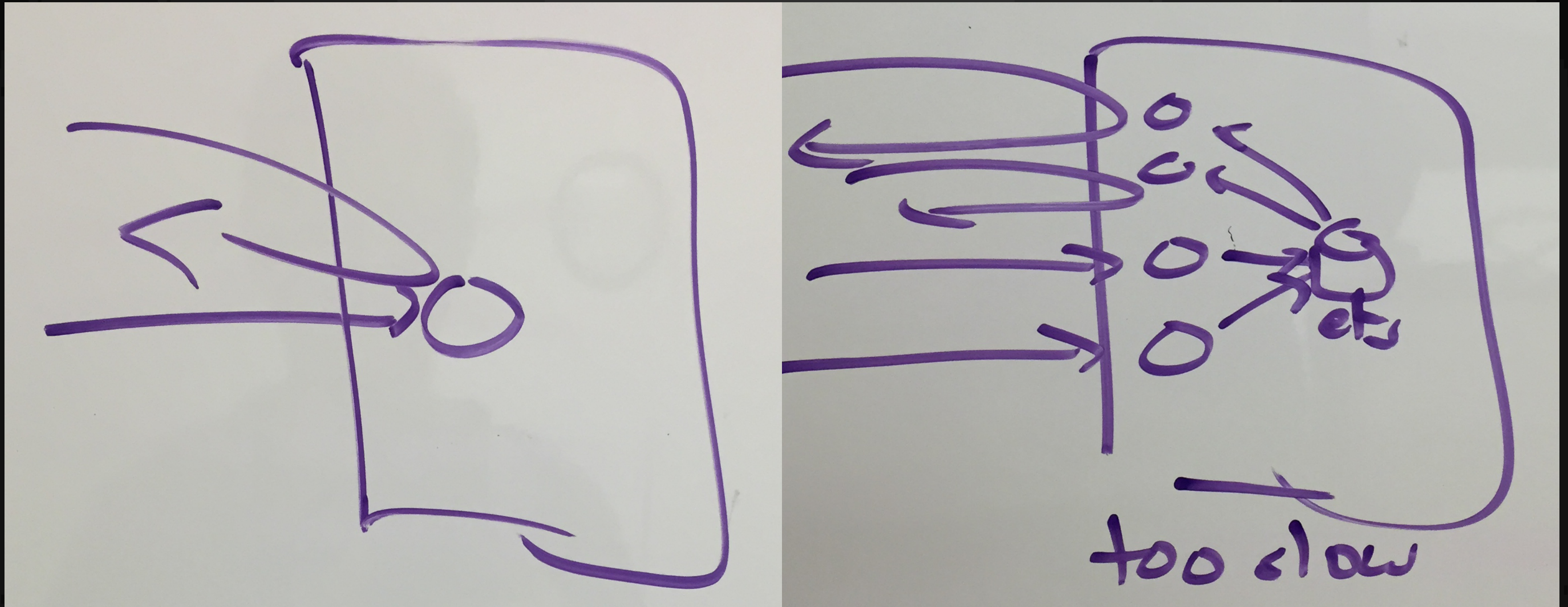
MachineZone

Q



MachineZone

Maximize # msgs



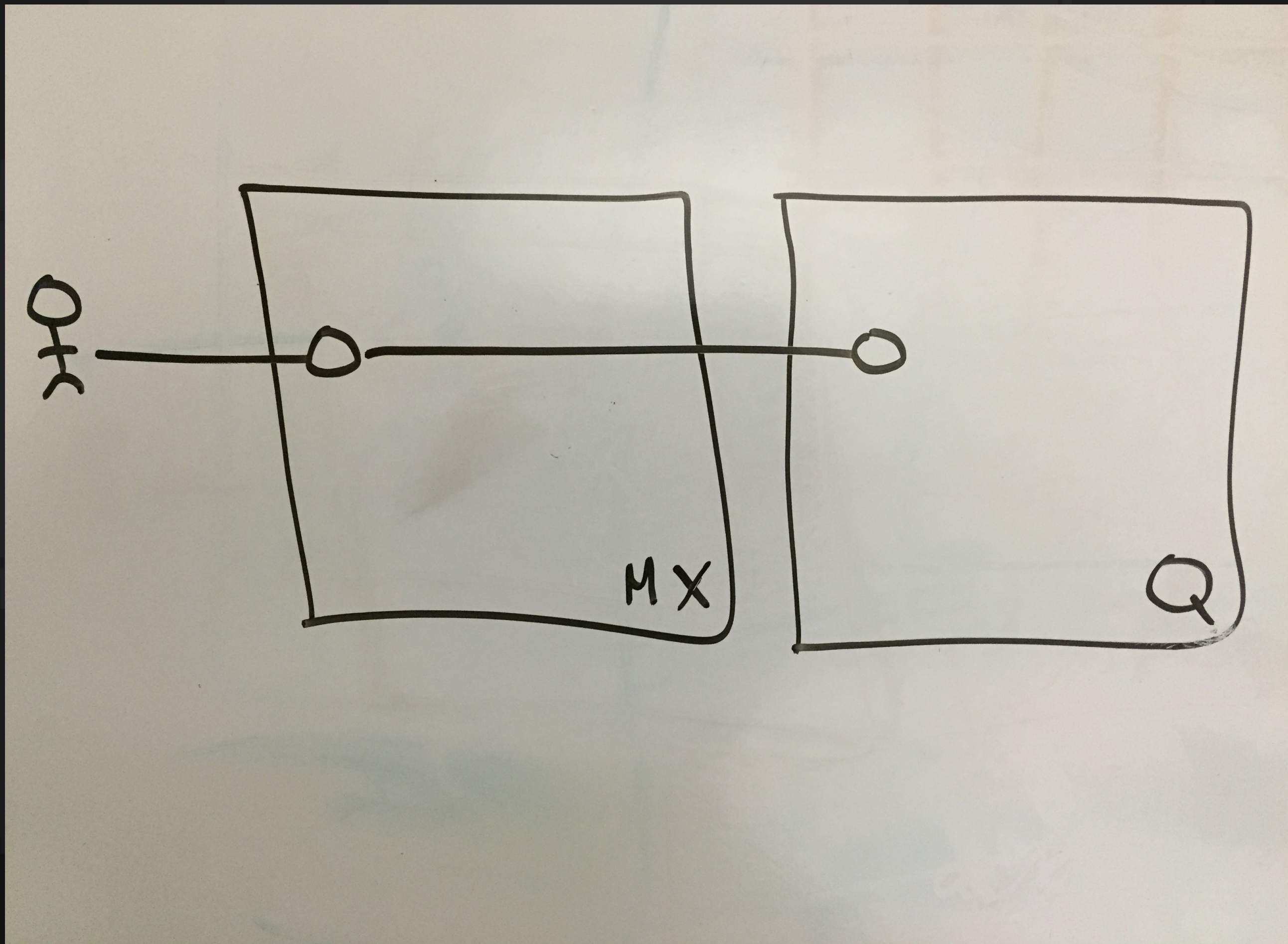
MACHINE ZONE

Maximize # msgs

- A non-blocking, order preserving multi-write-multi-read queue would do (NIF)

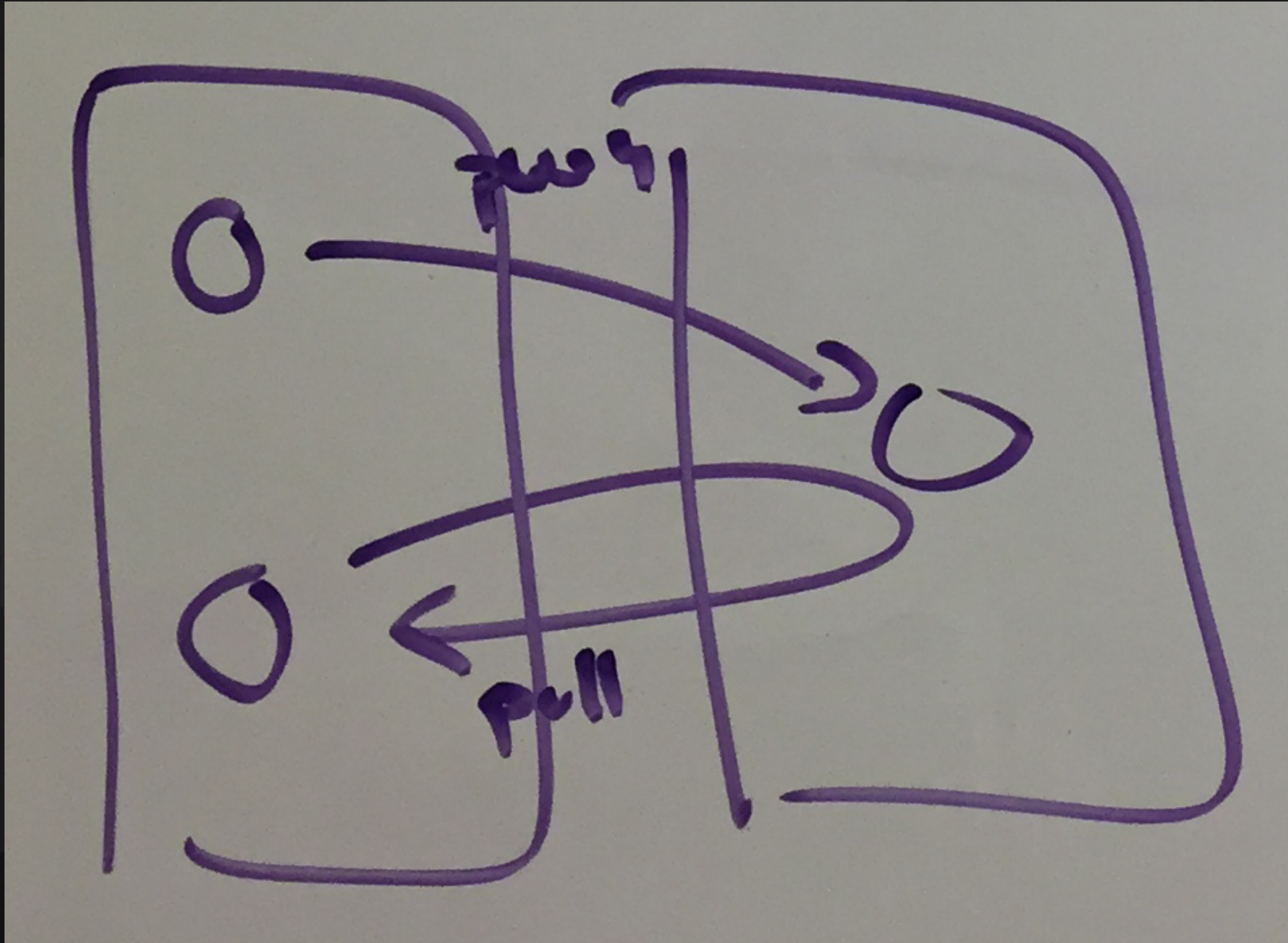
MachineZone

Q – Mx



MACHINEZONE

Pull



MachineZone

Pull

- Push to and pull from the same process
- Cannot send between pulls

MachineZone

Pull

- Push to and pull from the same process
  - Competes over the same inbox:
    minimize queuing delay
  - Competes for the same processing capacity:
    maximize # msgs
- Cannot send between pulls

MachineZone

Pull

- Push to and pull from the same process
  - Competes over the same inbox:
    minimize queuing delay
  - Competes for the same processing capacity:
    maximize # msgs
- Cannot send between pulls
  - Sending blocked between requests:
    minimize queuing delay
  - Data sent at irregular intervals:
    minimize queuing delay
  - Each pull is from a different offset:
    minimize processing time

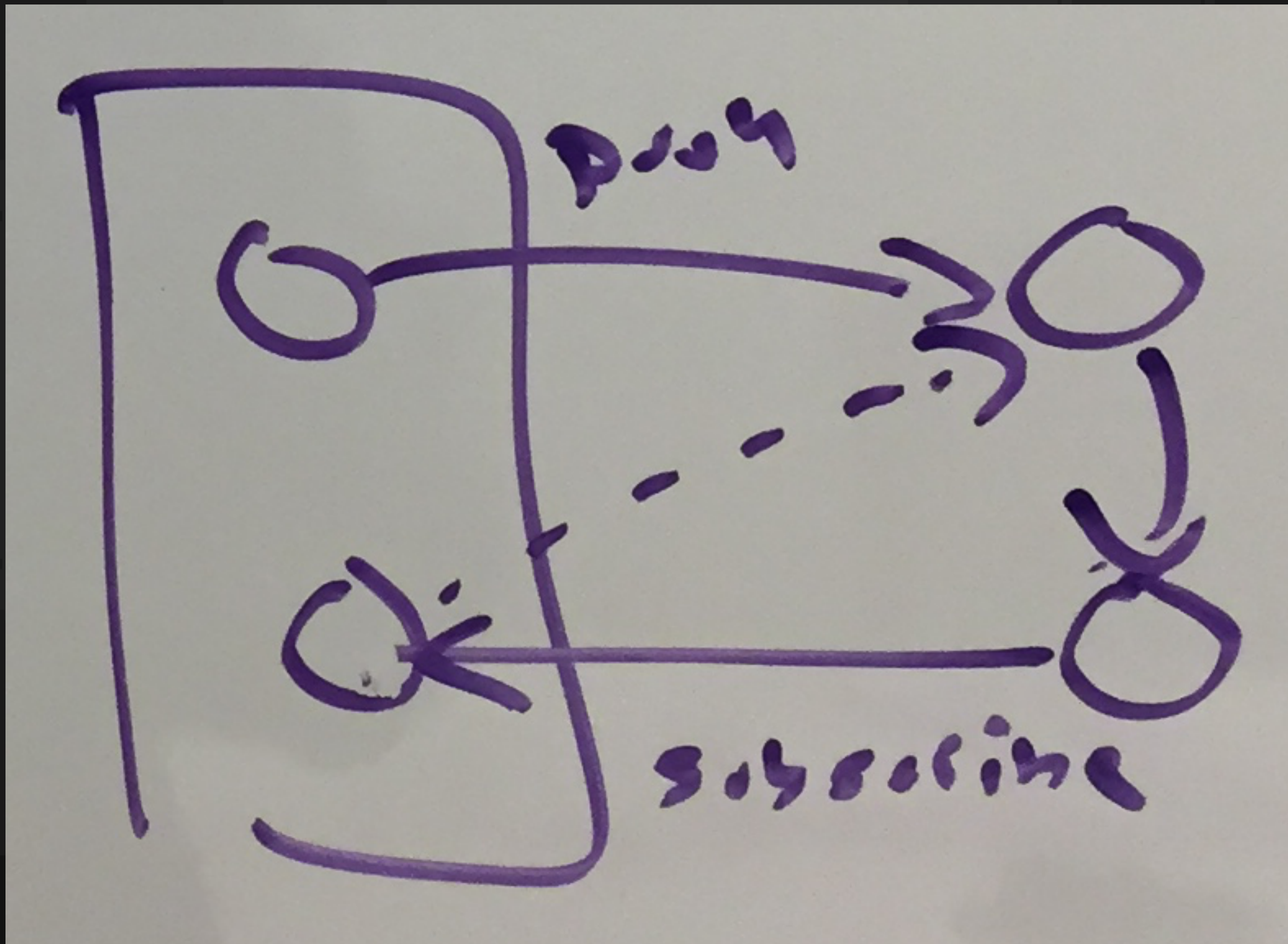
MachineZone

Solution

- Separate into writer and reader processes
- Pull => subscribe protocol

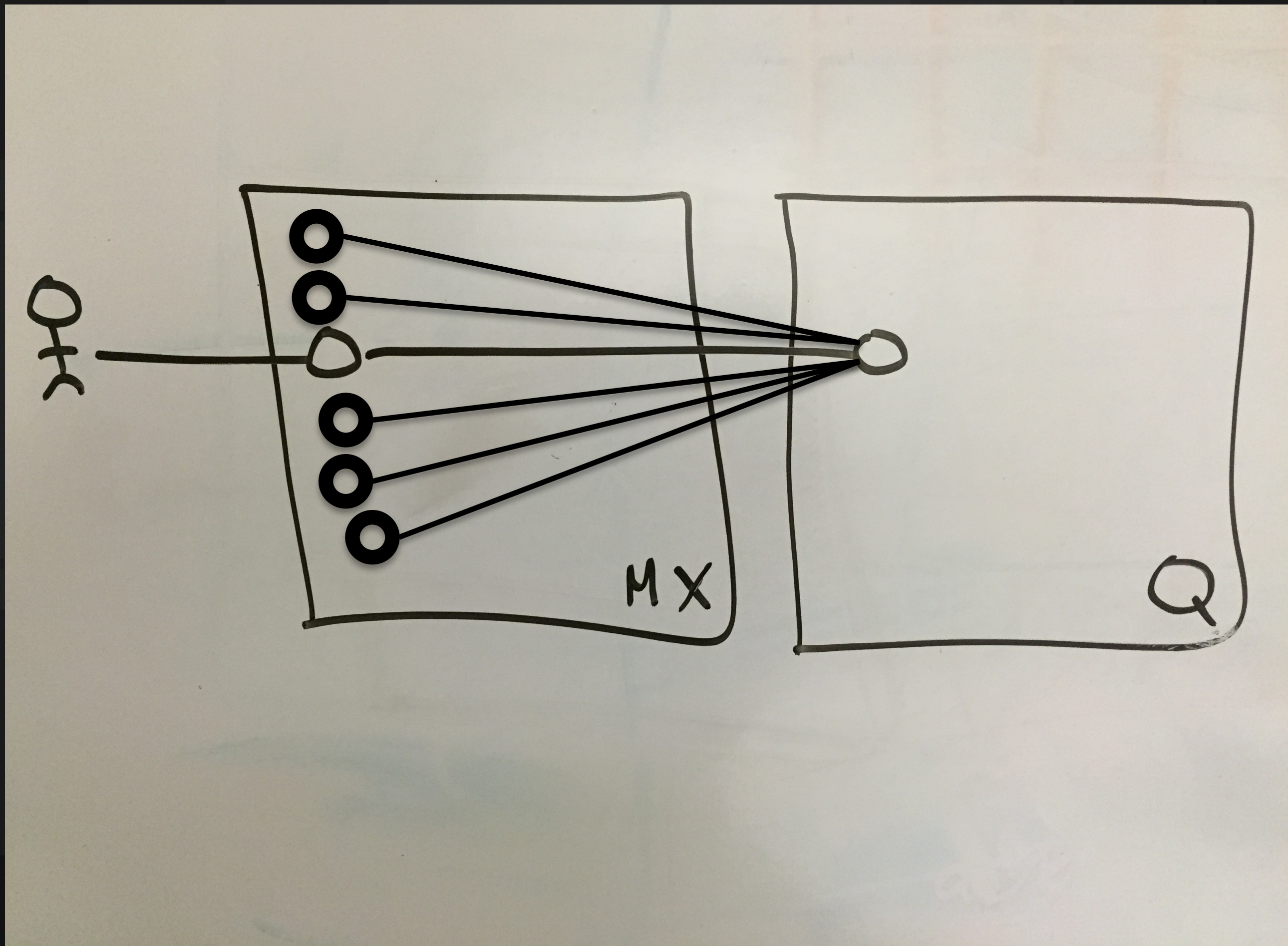
MachineZone

Solution



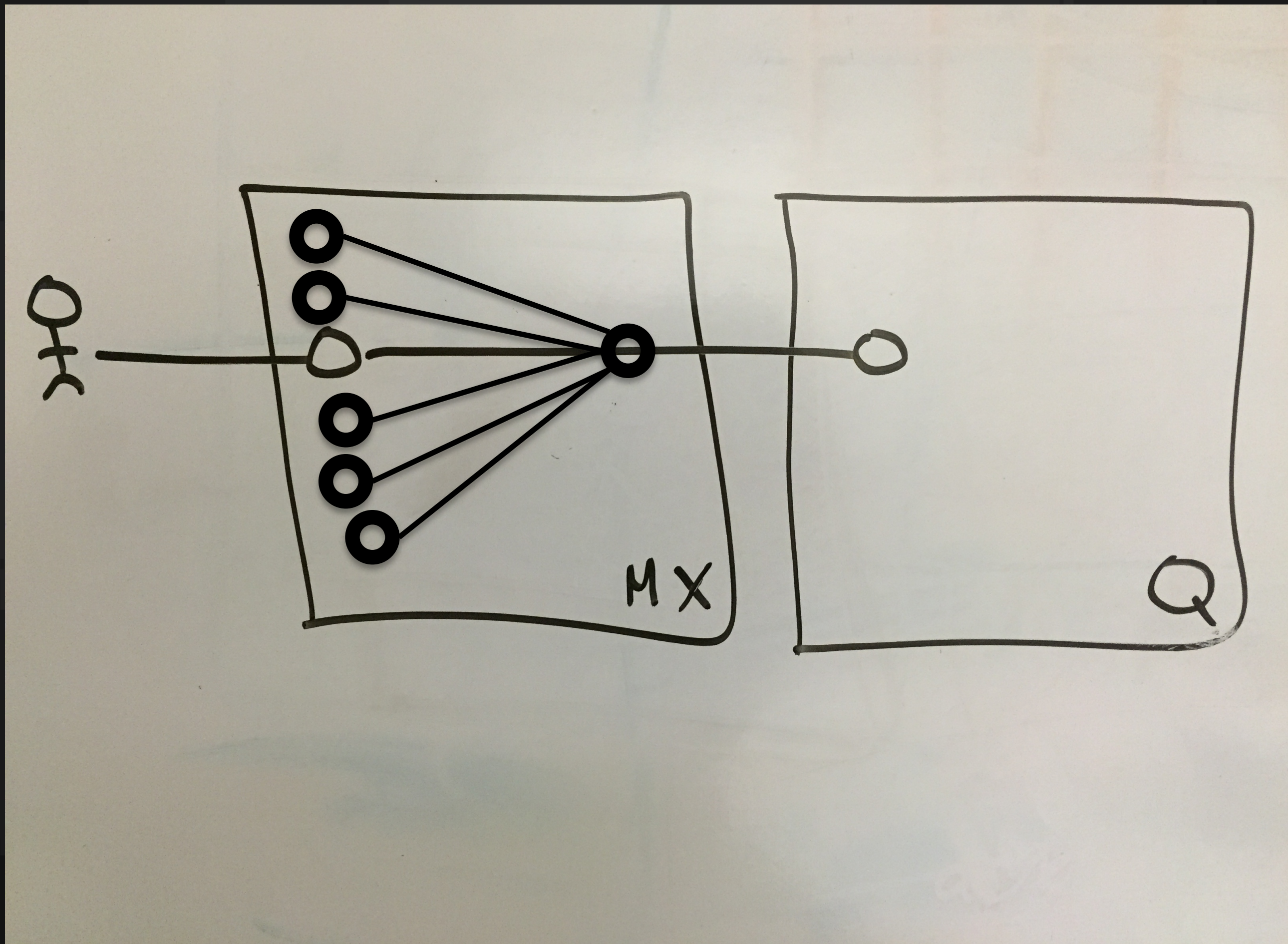
MachineZone

Limited bandwidth



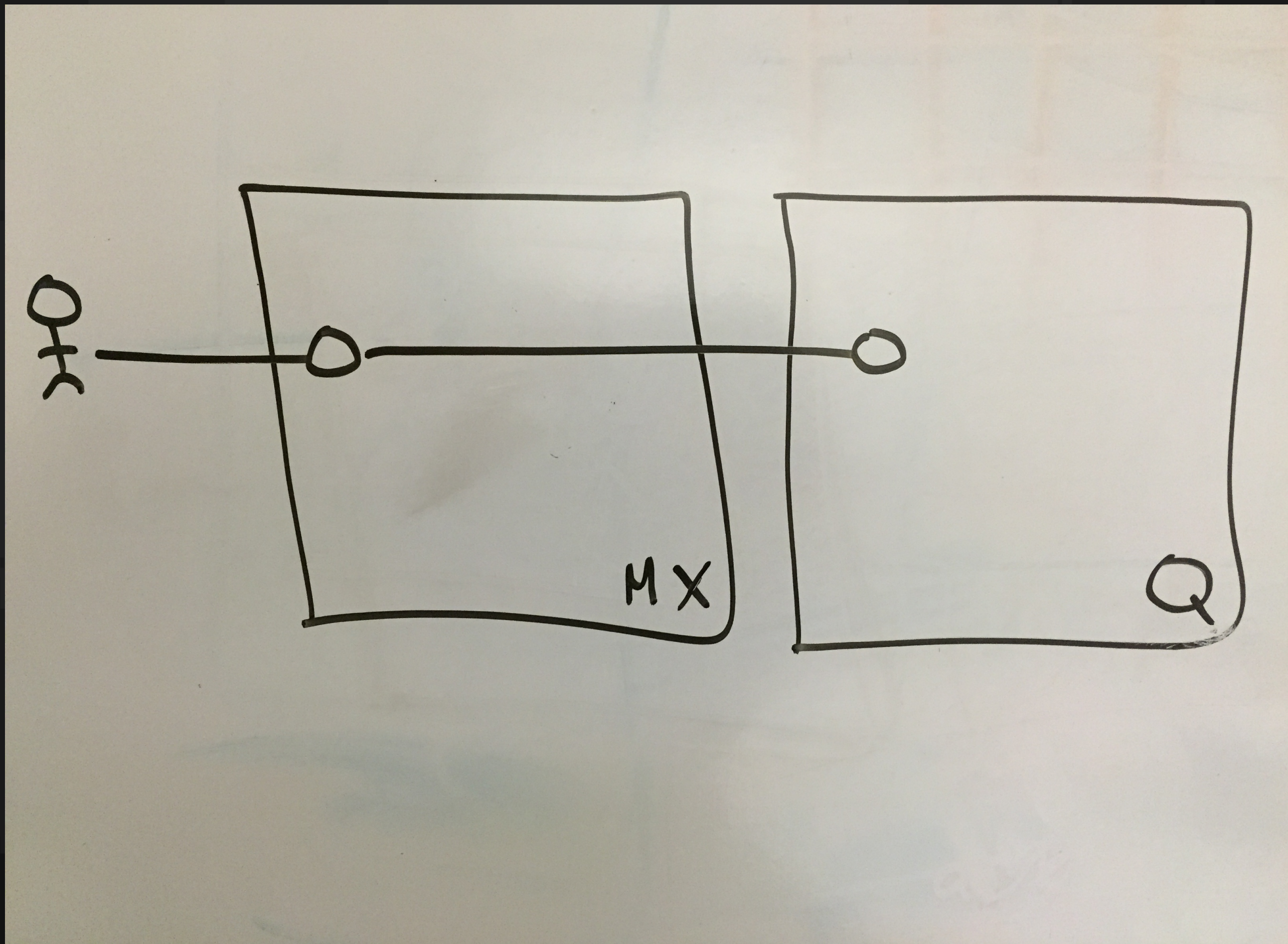
MachineZone

Maximize # msgs



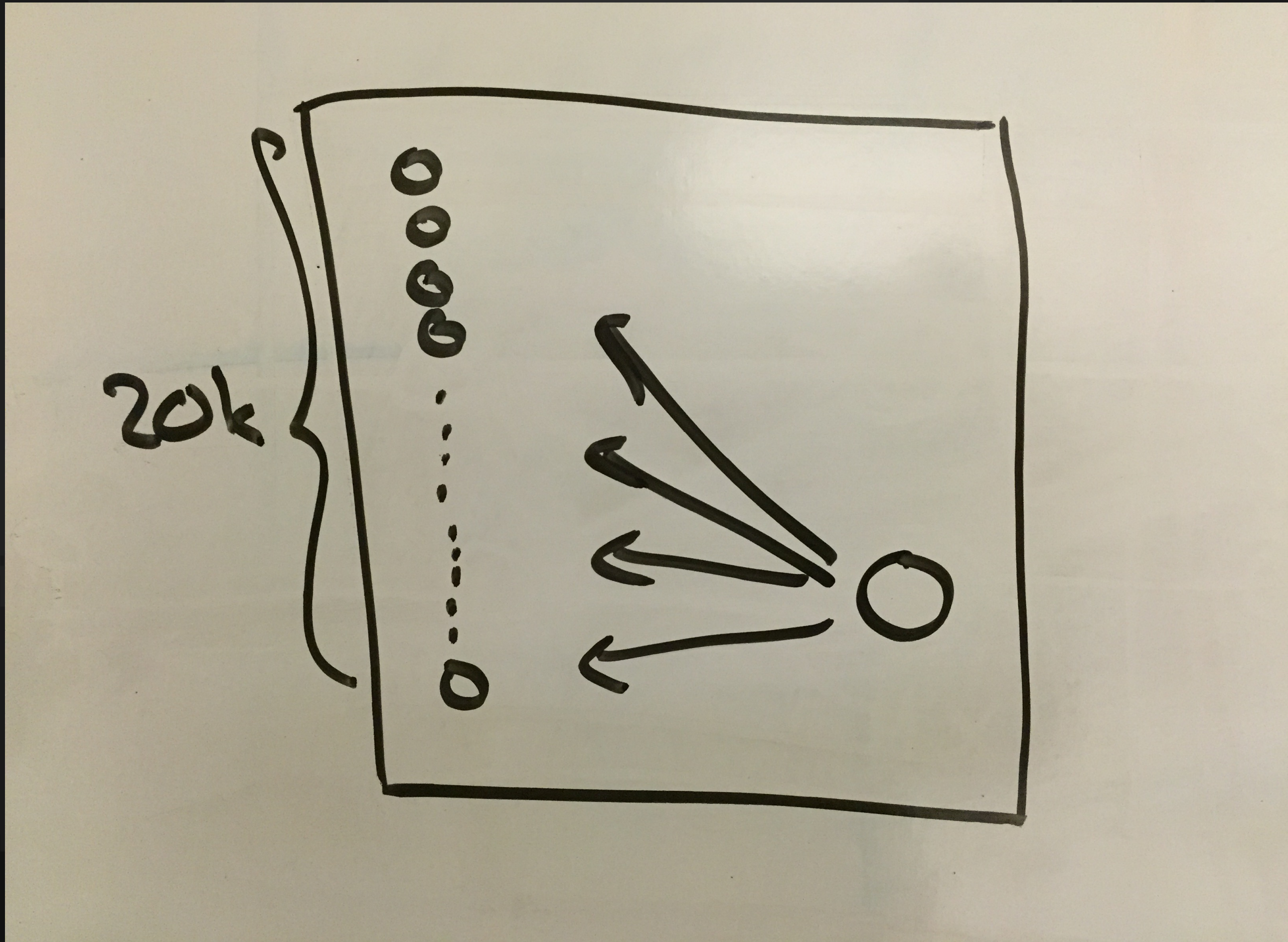
MachineZone

Mx



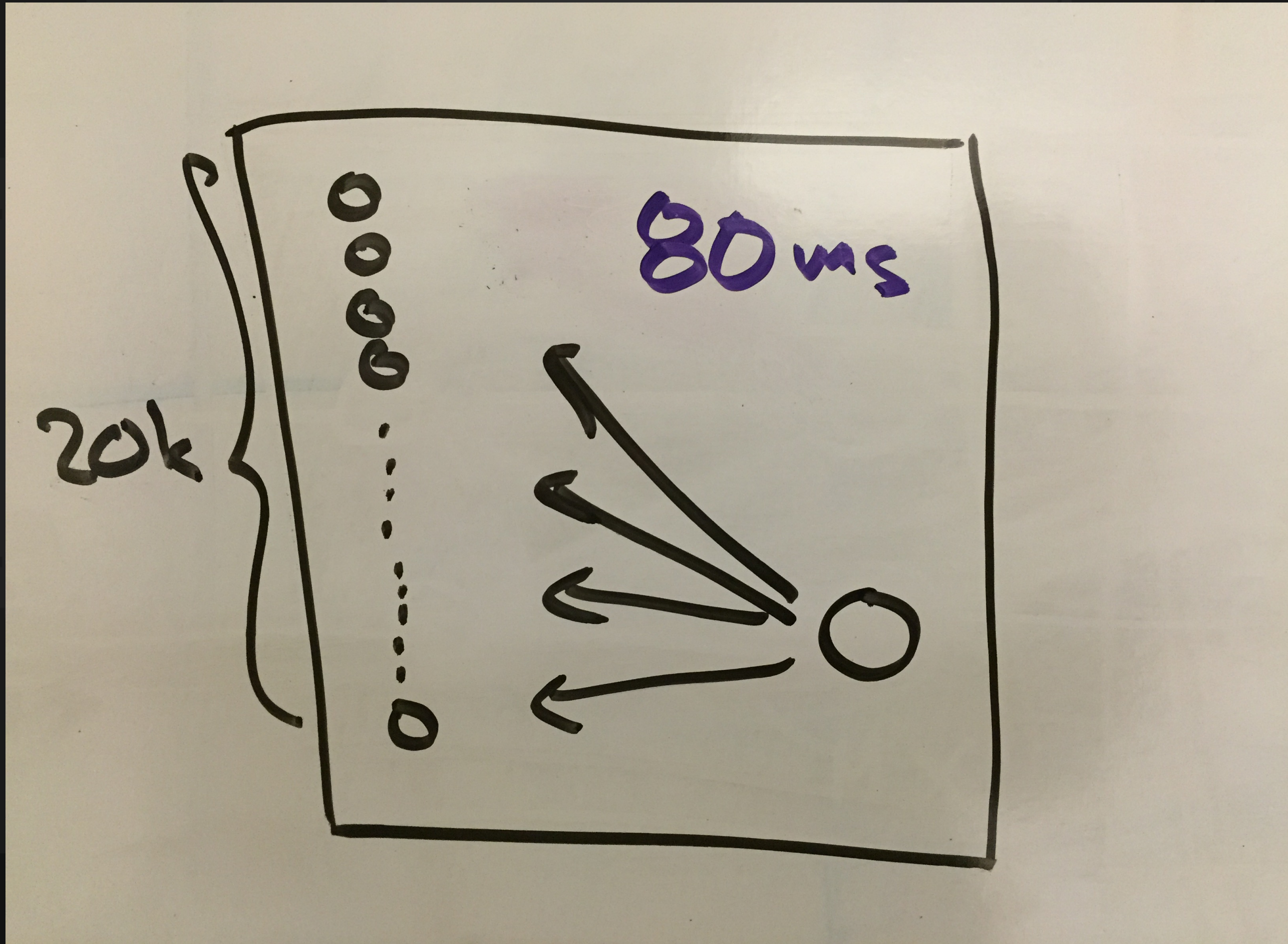
MachineZone

Send to many processes



MACHINE ZONE

Send to many processes



MACHINE ZONE

Minimize queuing delay

- NIF
  - Single writer
  - Multiple readers pulling as long as data is available
  - Notify idle readers only

MachineZone

Minimize processing time

- Use the right algorithms
- Use the right data structure
- Use the right language constructs
- Avoid generating garbage

MachineZone

Key-value store

- There are plenty of key-value stores in Erlang/OTP
  - lists: [{key(), value()}]
  - proplists
  - dict
  - orddict
  - gb_trees
  - ets — set, ordered_set, bag
  - process dictionary
  - record — for a (small) finite set of keys

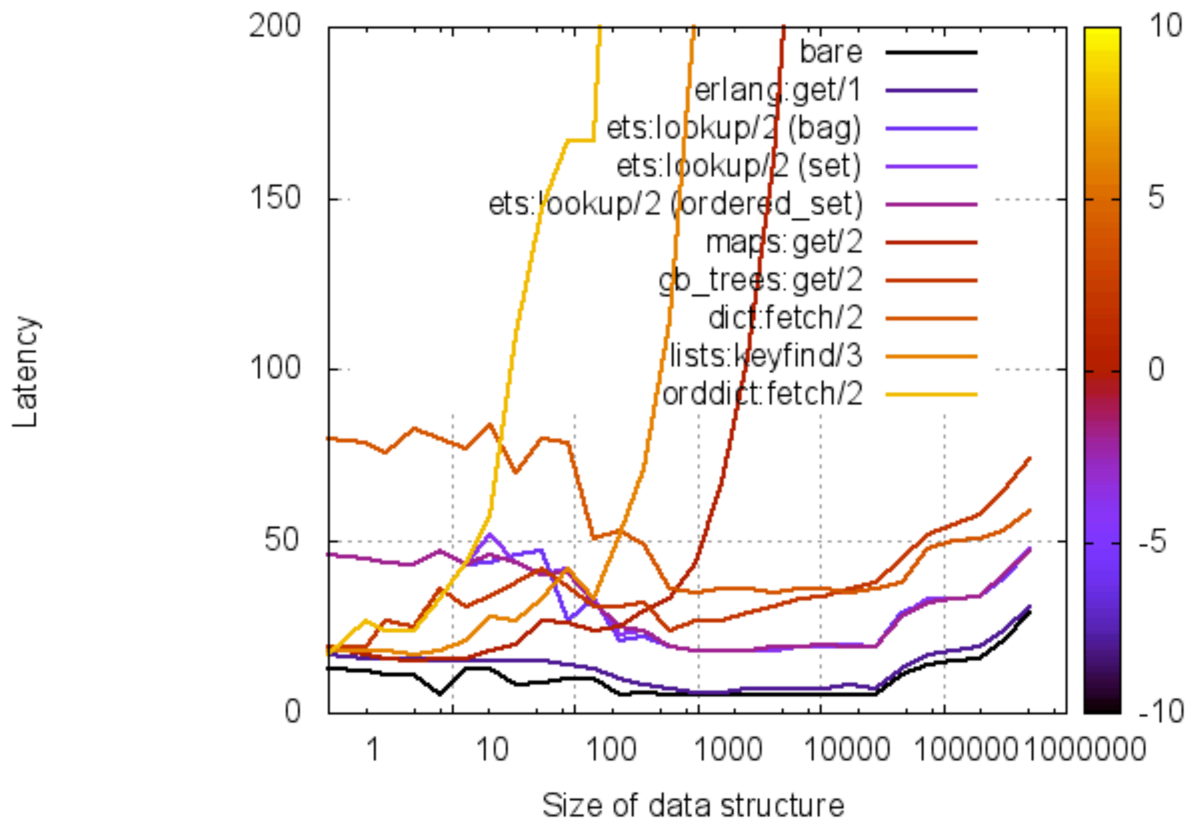
MachineZone

Test code snippet

```
measure(N, DeckKey, Txt, Fun) when N > 0 ->
    Timecap = timecap(),
    case ets:lookup(too_slow, Txt) of
        [] ->
            K = ets:lookup_element(deck, DeckKey, 2),
            erlang:garbage_collect(),
            {Latency, _} = timer:tc(fun() -> repeat(100, Fun, K) end),
            if
                Latency >= Timecap ->
                    ets:insert(too_slow, {Txt, []}),
                    lists:duplicate(N, Timecap);
                true ->
                    N2 = N - 1,
                    DeckKey2 = ets:next(deck, DeckKey),
                    [Latency | measure(N2, DeckKey2, Txt, Fun)]
            end;
        [_] ->
            lists:duplicate(N, Timecap)
    end;
measure(_, _DockKey, _Txt, _Fun) ->
    [].
```

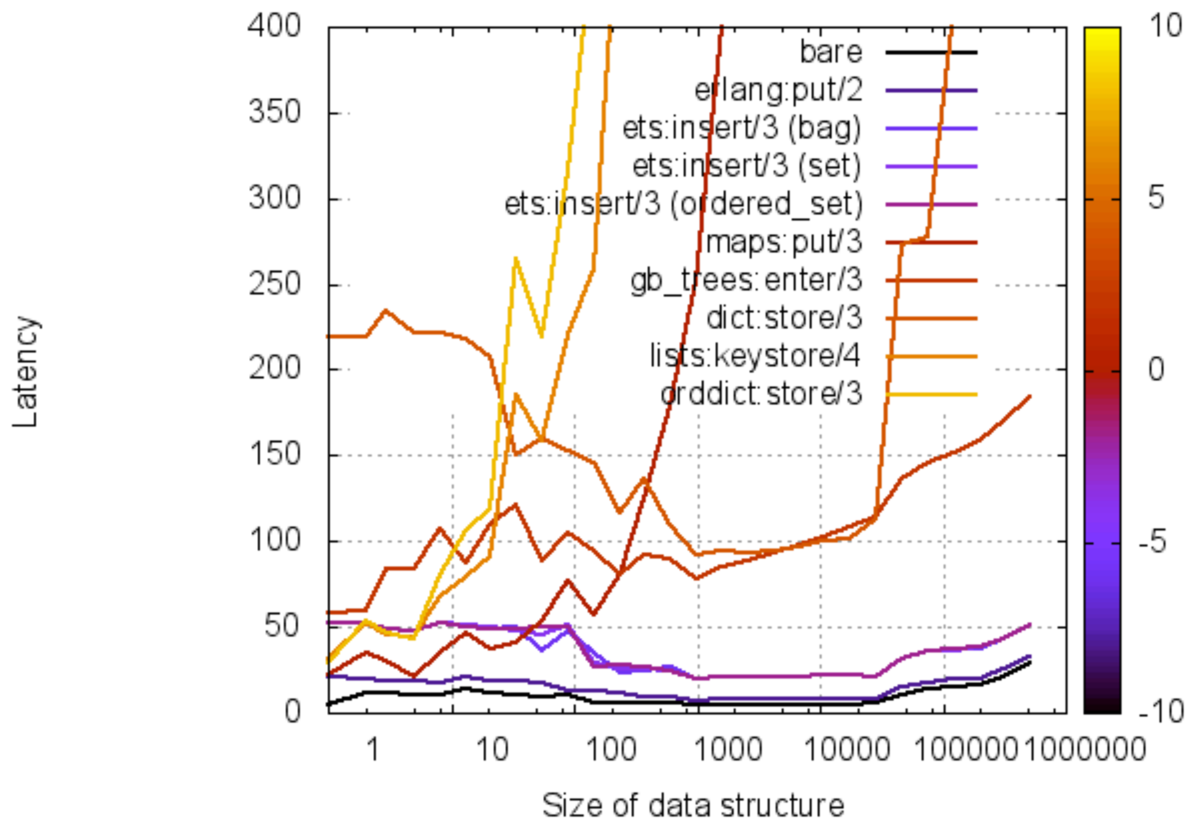
MachineZone

Getters



MachineZone

Setters



MachineZone

Erlang efficiency guide

- http://www.erlang.org/doc/efficiency_guide/users_guide.html

MachineZone

List comprehensions better than lists:{foldl,foldr,foreach,map}

```
foo() ->
    lists:foldl(
        fun({rec, x, _}, Acc)) ->
                 R = local_foo(x),
                 [R | Acc] %% garbage
        end, ...),
    ok.
===
foo() ->
    [local_foo(Rec#rec.x) || Rec <- List],
    ok. %% <- this line
```

MachineZone

Measurement

```
-module(x).
-compile(export_all).

run_all(N) ->
    D = data(N),
    lists:unzip(
        [run(fun foldl/1, D),
         run(fun foreach/1, D),
         run(fun lc/1, D)]).

run(F, D) ->
    erlang:garbage_collect(),
    erlang:yield(),
    B = reductions(),
    {T, _} = timer:tc(fun() -> F(D) end),
    A = reductions(),
    {T, A - B}.

reductions() ->
    {_,R}=process_info(self(),reductions),
    R.
```

```
data(N) ->
    lists:duplicate(N, value).


foldl(Data) ->
    _ = lists:foldl(
            fun(E, A) -> [E | A] end,
            [],
            Data),
    ok.

foreach(Data) ->
    _ = lists:foreach(
            fun(E) -> E end,
            Data),
    ok.

lc(Data) ->
    _ = [E || E <- Data],
    ok.
```

MachineZone

Measurement

```
Erlang/OTP 17 [erts-6.2] [source] [64-bit] [smp:8:8] [async-threads:10]
[kernel-poll:false]

Eshell V6.2  (abort with ^G)
1> x:repeat(10,10000).
{[{[525,417,72],[20011,20011,10011]},
  {[389,320,74],[20011,20011,10011]},
  {[387,321,72],[20011,20011,10011]},
  {[386,329,54],[20011,20011,10011]},
  {[341,287,54],[20011,20011,10011]},
  {[340,282,54],[20011,20011,10011]},
  {[340,253,49],[20011,20011,10011]},
  {[313,252,48],[20011,20011,10011]},
  {[304,253,49],[20011,20011,10011]},
  {[276,229,44],[20011,20011,10011]}],
 [{foldl,   360.1, 5.926365170598046 5},
  {foreach, 294.3, 4.602492462006751},
  {lc,       57.0, 0.9948273555495675}]}
```

MachineZone

Reductions

- Affects processing time (reduction ≈ speed)
- Affects scheduling

MachineZone

Avoid garbage

```
do(L) ->
    [do_something(E) || E <- L],
    ok.

avoid(L) ->
    L2 = [do_something(E) || E <- L],
    L2. %% L2 may be garbage
```

MachineZone

Measurement

```
-module(x).
-compile(export_all).

run_all(N) ->
    D = data(N),
    lists:unzip(
        [run(fun inline/1, D),
         run(fun avoid/1, D)]).

run(F, D) ->
    erlang:garbage_collect(),
    erlang:yield(),
    B = reductions(),
    {T, _} = timer:tc(fun() -> F(D) end),
    A = reductions(),
    {T, A - B}.

reductions() ->
    {_,R}=process_info(self(),reductions),
    R.
```

```
data(N) ->
    lists:duplicate(N, value).


inline(Data) ->
    _ = [do_something(E) || E <- L],
    ok.

avoid(Data) ->
    [do_something(E) || E <- L].
```

MachineZone

Measurement

```
Erlang/OTP 17 [erts-6.2] [source] [64-bit] [smp:8:8] [async-threads:10]
[kernel-poll:false]

Eshell V6.2  (abort with ^G)
1> x:repeat(10,100000).
{[{[419,2489],[100011,100011]},
  {[392,1894],[100011,100011]},
  {[428,1889],[100011,100011]},
  {[389,1022],[100011,100011]},
  {[392,1057],[100011,100011]},
  {[389,1037],[100011,100011]},
  {[404,1127],[100011,100011]},
  {[441,985],[100011,100011]},
  {[390,1075],[100011,100011]},
  {[390,1023],[100011,100011]}],
 [{do,     403.4,  0.4083703936806842},
  {avoid, 1359.8, 90.38233336363601}]}
```

MachineZone

Prefer “inline”

```
do(L) ->
    [begin
         X = do_something(E),
         do_something_else(X)
     end
     || E <- L],
    ok.

avoid(L) ->
    L2 = avoid2(L),
    avoid3(L2), %% L2 may be garbage
    ok.
avoid2(L) ->
    [do_something(E) || E <- L].
avoid3(L) ->
    [do_something_else(E) || E <- L].
```

MachineZone

Measurement

```
-module(x).
-compile(export_all).

run_all(N) ->
    D = data(N),
    lists:unzip(
        [run(fun inline/1, D),
         run(fun avoid/1, D)]).

run(F, D) ->
    erlang:garbage_collect(),
    erlang:yield(),
    B = reductions(),
    {T, _} = timer:tc(fun() -> F(D) end),
    A = reductions(),
    {T, A - B}.

reductions() ->
    {_,R}=process_info(self(),reductions),
    R.
```

```
data(N) ->
    lists:duplicate(N, value).

do(L) ->
    [begin
         X = do_something(E),
         do_something_else(X)
     end
     || E <- L],
    ok.

avoid(L) ->
    L2 = avoid2(L),
    avoid3(L2),
    ok.
avoid2(L) ->
    [do_something(E) || E <- L].
avoid3(L) ->
    [do_something_else(E) || E <- L].
```

MACHINE ZONE

Measurement

```
Erlang/OTP 17 [erts-6.2] [source] [64-bit] [smp:8:8] [async-threads:10]
[kernel-poll:false]

Eshell V6.2  (abort with ^G)
1> x:repeat(10,100000).
{[{[1341,5848],[300011,408563]},
  {[1103,3617],[300011,408564]},
  {[1106,4263],[300011,408565]},
  {[1104,3656],[300011,408567]},
  {[1104,3644],[300011,408568]},
  {[1254,3625],[300011,408569]},
  {[1177,3534],[300011,408570]},
  {[1295,4657],[300011,408571]},
  {[1227,3768],[300011,408572]},
  {[1187,3815],[300011,408573]}],
 [{do,    1189.8,  2.891326370742263},
  {avoid, 4042.7, 58.14763189270434}]}
```

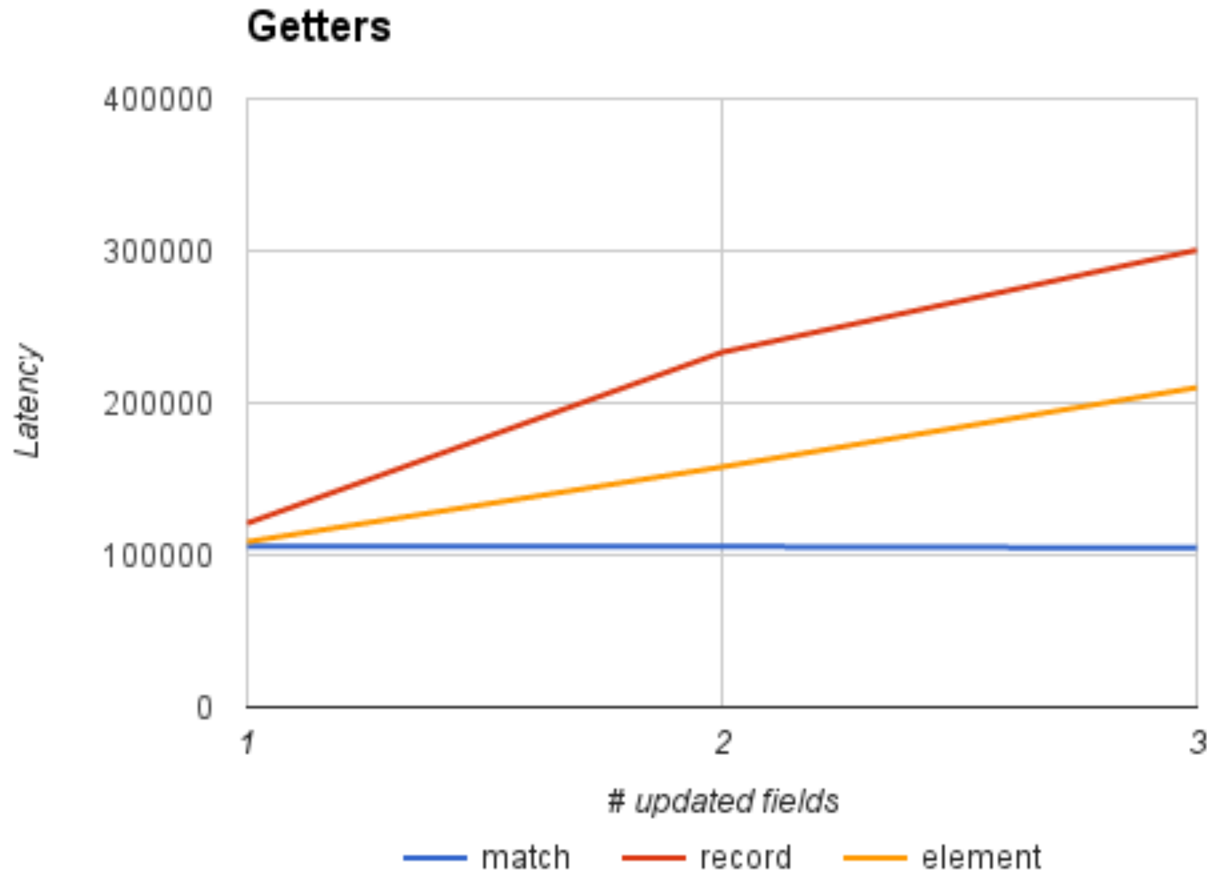
MachineZone

Match, record and element/2

- `#x{v = V} = X`
- `X#x.v`
- `element(#x.v, X)`

MachineZone

Match, record and element/2

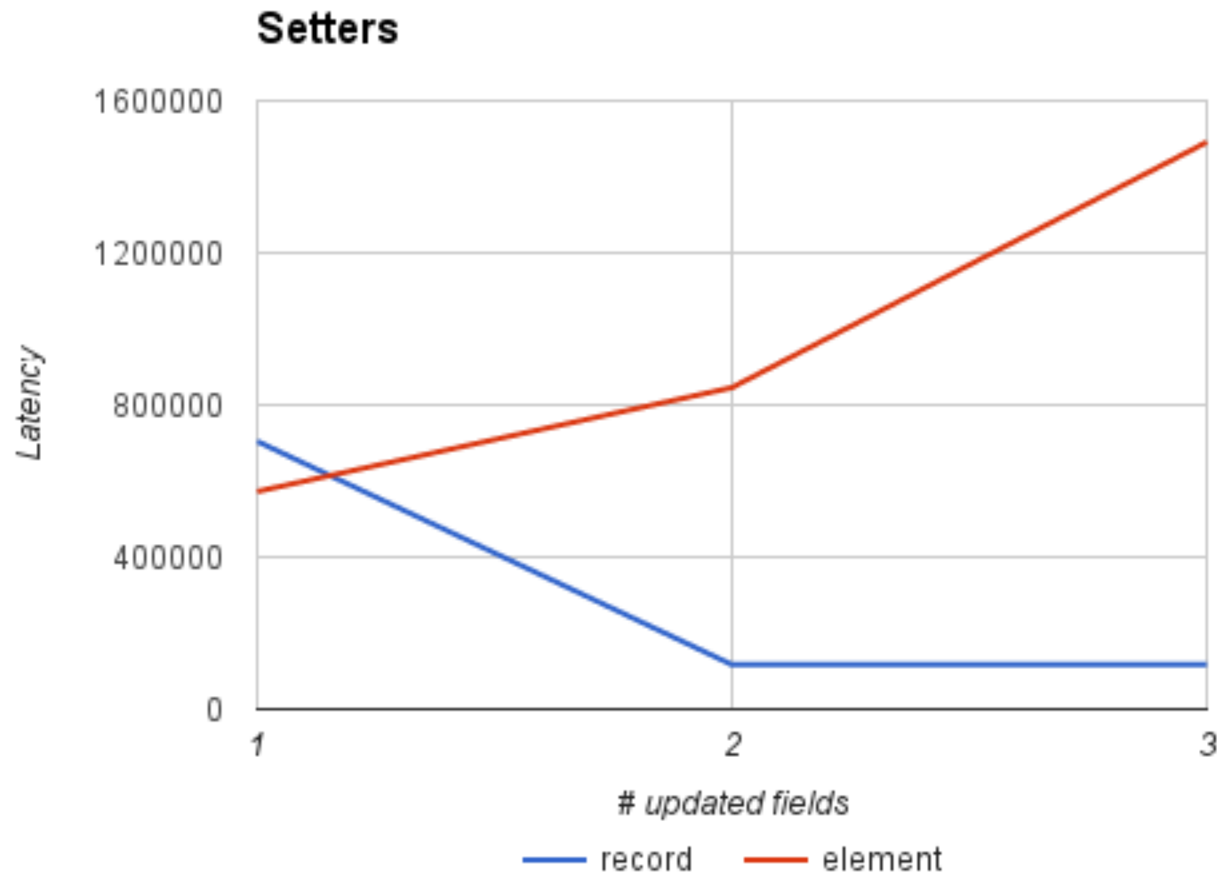


Record and setelement/3

- `X#x{v = V}`
- `setelement(#x.v, X, V)`

MACHINE ZONE

Record and setelement/3



MACHINEZONE

State record vs process dictionary

- Proc dict slightly faster
  - … for single key access

MachineZone

NIF “evil” or not?

MACHINE ZONE

NIF

- It's a tool
  - We use it for things Erlang is bad at
- Caveats
  - 1 ms max call time (dirty schedulers)
  - Cannot `meck` nif modules
  - No support for C/C++ unit tests (rebar etc)
  - Unique concept – ErlNifEnv (owns ERL_NIF_TERM)
  - Unique concept – NIF Resource

MACHINE ZONE

NIF resources

- Perfect for sharing data
  - Use when message passing / `ets` is too slow
- Perfect for fast data structures
  - Use when `ets` is too slow

MachineZone

NIF terms

- All terms belong to a context
  - Called `ErlNifEnv`
- Uses
  - Given in each call, used for return values
  - Send env – for term sent to a process
  - Separate env – for values that should survive a call

MachineZone

MZ
MachineZone