Scaling messaging systems

Michał Piotrowski
michal.piotrowski@erlang-solutions.com
@michalwski

FOSDEM 2018

Who am I

- Happy husband to wonderful and inspiring wife
- Happy father to:
  - 3.5 years old, full of life daughter
- Senior Software Developer at Erlang Solutions
  - 6 years
  - Involved in many XMPP projects for customers and MongooseIM product development
- I’m running
  - Finished 2 half-marathons
  - Preparing for marathon in April

Agenda

1. MongooseIM - what it is
2. Scalability considerations
3. Starting small
4. When one node is not for you?
5. What one cluster is not enough?

1.

MongooseIM

MongooseIM platform

- XMPP server written in Erlang
  - Newest version 2.1.1: https://www.erlang-solutions.com/blog/mongooseim-2-1-1-more-than-a-patch.html
- MongoosePush
  - Standalone service in Elixir for push notifications
  - https://github.com/esl/MongoosePush
- MongooseICE
  - Standalone stun/turn server in Elixir
  - https://github.com/esl/MongooseICE
- Contributions to client libraries
  - Smack
  - XMPPFramework

2.

Scalability considerations

> “
>
> Every MongooseIM installation is unique. There is no single scalability pattern which would work for everyone in the same way.

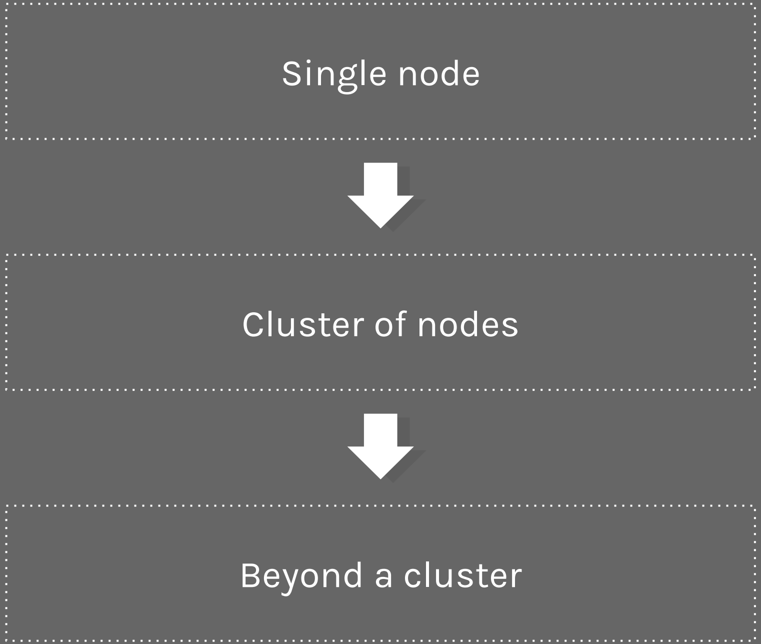
Scalability depends on many variables

- Machine power (CPU, memory, IO)
- Type of connected users / devices
  - Mobile applications
  - Web clients
- Used features
  - One-to-one chat
  - Group chat
  - Presence
  - Message archiving
- Database usage

Scalability limitations

- Memory
    - How many online users / devices connected to a single node we can have?
- CPU
    - How “chatty” are the users / devices ?
- Database
    - Is every message archived?

MongooseIM scalability steps



Single node

Cluster of nodes

Beyond a cluster

3.

Starting small

Single node

- Erlang VM allows to utilise all available CPUs / cores on the machine
- Connected devices are modeled around processes
  - 2-3 depending if we terminate encryption on MongooseIM or some where before
- Adding more resources increases the capacity

Single node - advantages

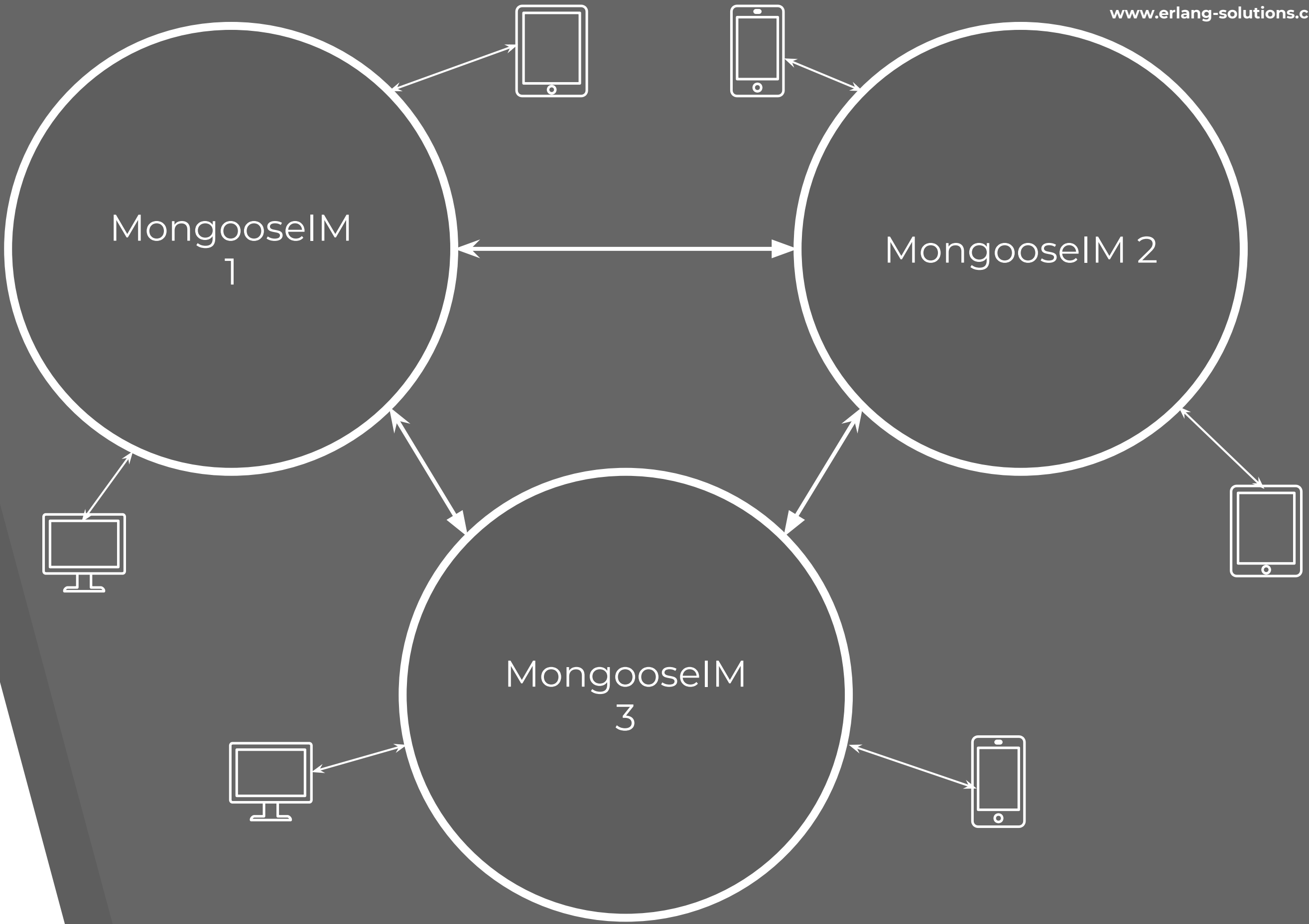
- Good for development
- Good for small installations
- Easy to manage
- Easy to monitor

Single node - disadvantages

- Single point of failure
- Makes it harder to prepare for production use

4.

Cluster



Building MongooseIM cluster

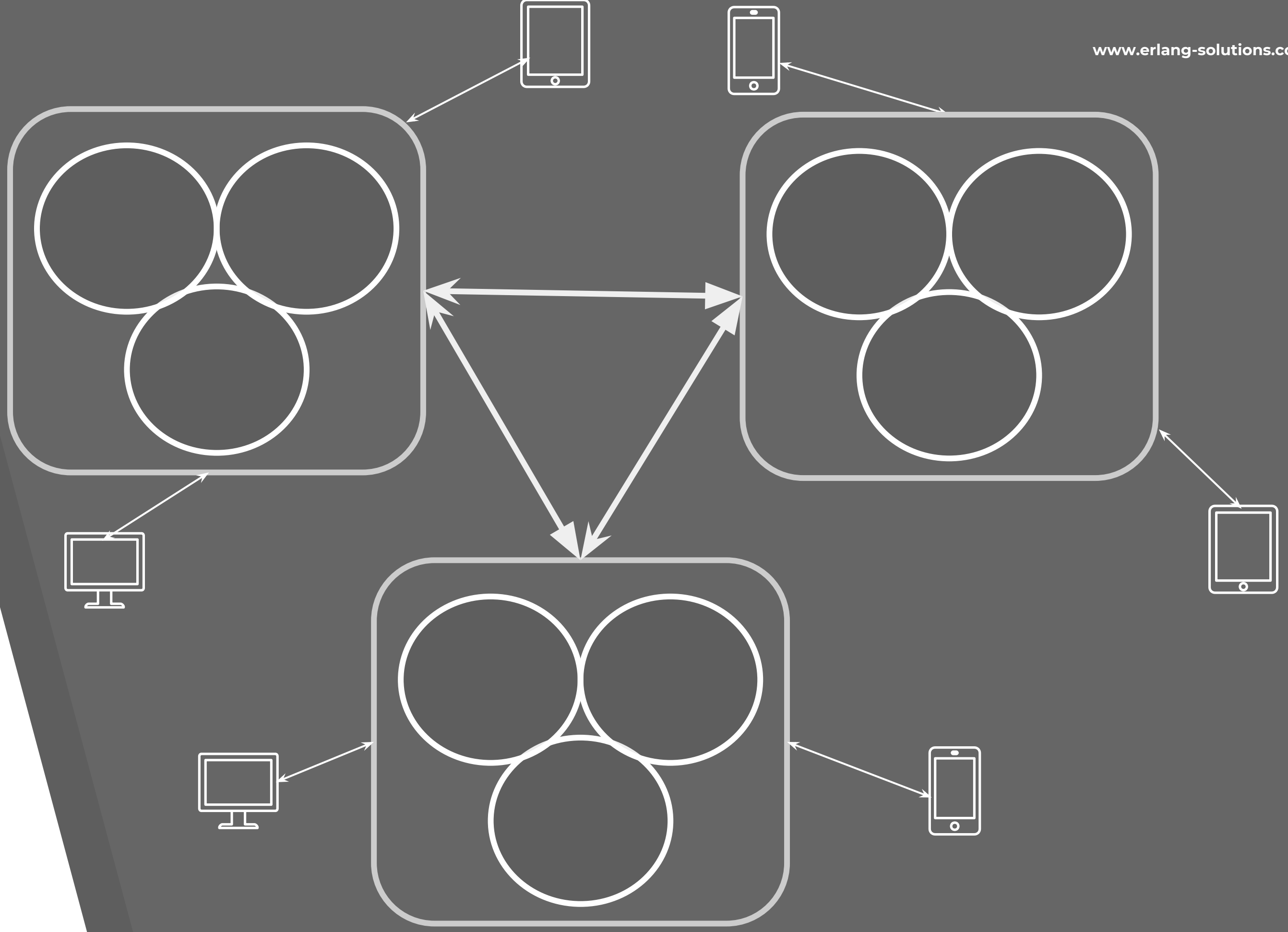
- Erlang distribution layer
- All nodes connected to others in the cluster
- Session information replicated on all nodes
  - Mnesia table
- Keep room (memory, CPU) to handle traffic from crashed / stopped node
- It’s recommend to keep persistent data in external DB
- http://mongooseim.readthedocs.io/en/latest/operation-and-maintenance/Cluster-configuration-and-node-management/

Cluster limitations

- The cluster is too large
  - 15+ nodes
- The database is overloaded
- Latency matters

5.

Multiple Clusters



How clusters can communicated

- Depends on the use case
- Can we shard users?
- Should users be able to connect any cluster?

How clusters can communicated - XMPP federation

- Users are bound to a cluster
  - Always connects to the same cluster
- Every cluster operates its own xmpp domain
  - eu1.domain.com
  - us1.domain.com
- Users in different clusters has different JID
  - user1@eu1.domain.com
  - user2@us1.domain.com
- Standard XMPP federation between clusters
  - Always possible with MongooseIM

How clusters can communicated - GEO distribution

- Users can be connect to any cluster
  - Usually the closest one
- Every cluster operates the same domain
  - domain.com
- Session/routing info replication
  - Redis with Dynomite replication layer
  - Extended XMPP federation between clusters

How clusters can communicated - GEO distribution

- Already in the master branch: http://mongooseim.readthedocs.io/en/latest/modules/mod_global_distrib/
- Still experimental, works for basic features
  - Message routing
  - Session management
  - MUC
  - External components
- Part of 3.0 release

Thank you!

Michał Piotrowski
michal.piotrowski@erlang-solutions.com
@michalwski