Erlang在大规模分布式系统中的问题与应对

Using Erlang in a large-scale distributed system

梁宇鹏 Eric Liang

环信 首席架构师兼IM技术总监
Chief Architect & IM Director of Easemob

关于我们 About us

- 环信 Easemob.com
  - 移动客服 SaaS 和 即时通讯云 IM PaaS
  - 帮助移动 APP 添加聊天功能
- 梁宇鹏 @一乐 Eric Liang
  - 环信首席架构师 兼 IM 技术总监
  - 原新浪微博 通讯技术专家
  - 专注IM领域，XMPP 开源项目
  - Jabberd2/C、Ejabberd/Erlang、Openfire/Java

大纲 Outline

- 即时通讯云服务的架构演进
  - The evolution of our IM cloud service’s architecture
- 关于Erlang你应该了解的
  - Something you should know about Erlang
- 一些工具和提示 Tools and tips
- 我们使用Erlang的原则 The principles we using Erlang

我们的服务规模 Scale

- An IM platform, which is soft real-time
  - for 82149 apps / 564M devices / 1B users
  - Millions of simultaneous online users
- A typical Erlang cluster of 5 running
  - on 221 machines with 3176 cores and 8608G memory
  - in 2 IDCs (latency < 10ms)

扩展是为了生存，优化是为了生活

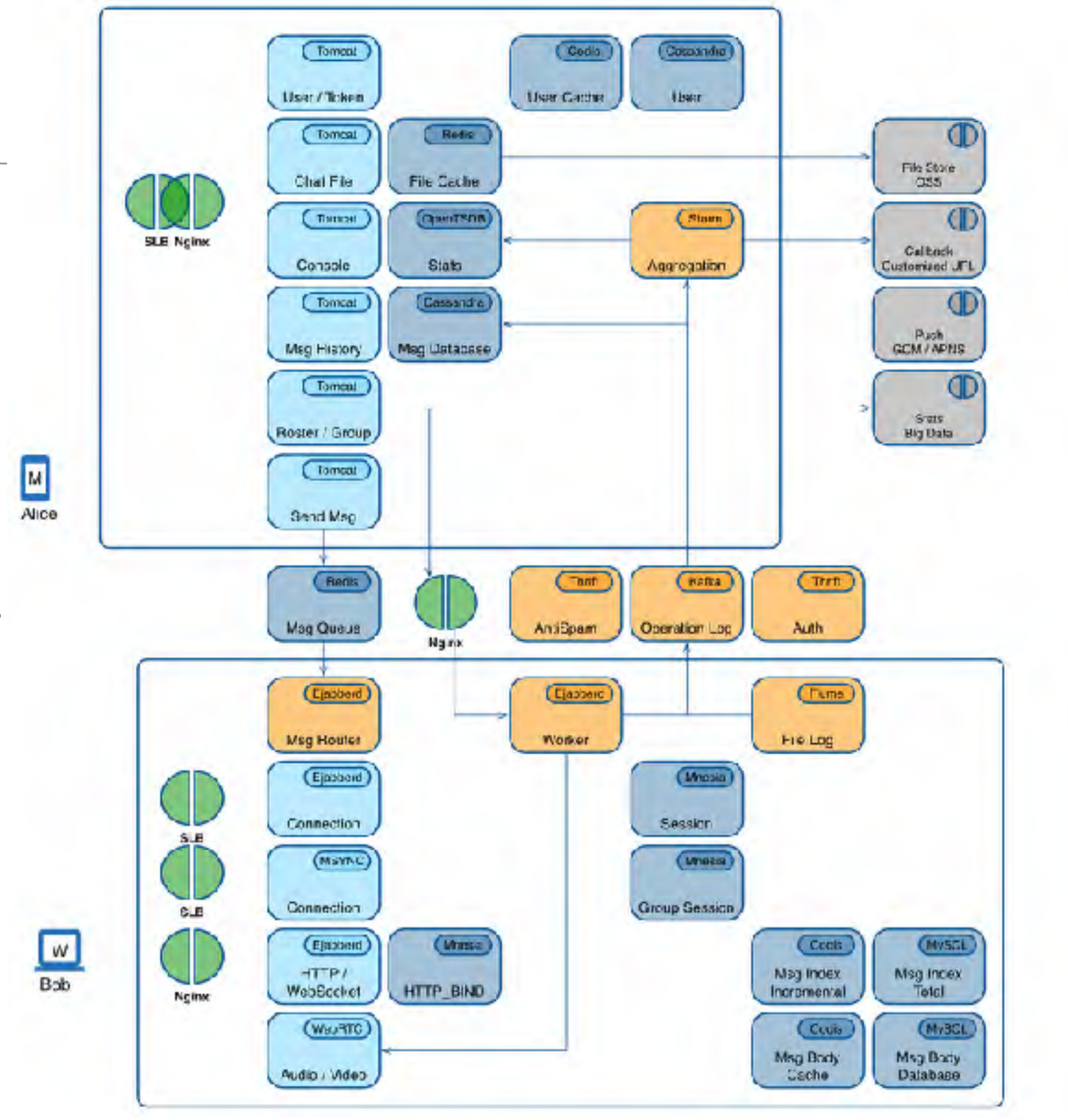
Scalability is for survival, optimization for life.

不一样的架构演进 A different way of evolution

- 每月一翻的增长速度，云服务的虚拟机环境
- 我们选择水平扩展 Scale out with cheaper VMs
  - 梁宇鹏：指数级增长业务下的服务架构改造
    - http://t.cn/R2DCcS7
- 而不是 WhatsApp 看重的垂直扩展 Scale up
  - Rick Reed: Scaling to Millions of Simultaneous Connections
  - http://t.cn/Rft09QR

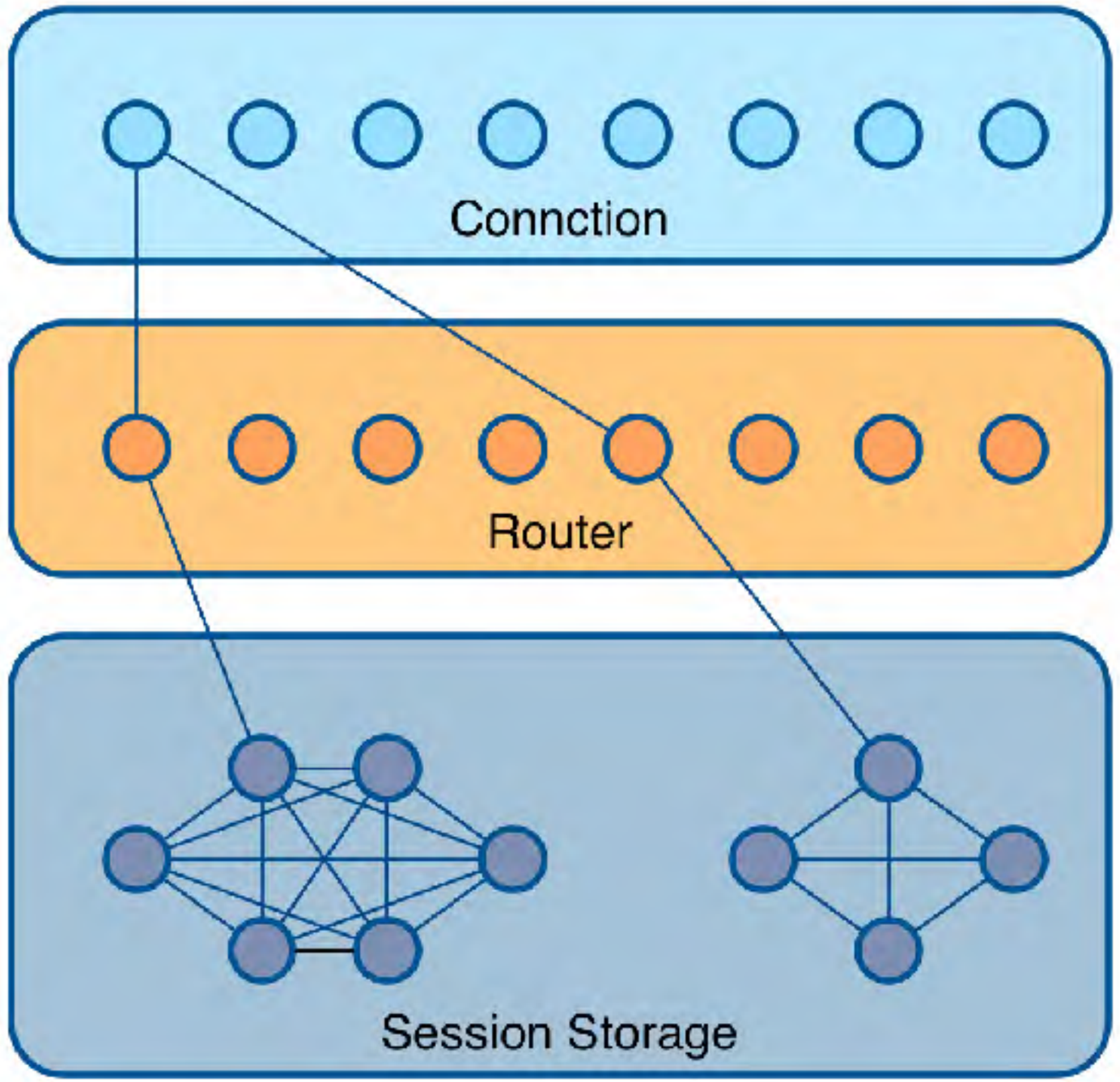
架构 Architecture

- 整个平台分为两部分
  - 无状态的HTTP服务
    - Java within Tomcat
    - Kafka / Storm / Cassandra
- 长连接的IM服务
  - Erlang within Ejabberd/ Mochiweb
  - Mnesia Codis MySQL



架构 Architecture

- IM服务分层设计
- Layered IM Services
    - 无状态的连接层
        - Stateless Connection
    - 无状态的路由层
        - Stateless Router
- 分组的会话存储层
    - Grouped Session Storage
    - Multi-tenancy
    - Mnesia and Codis



问: Mnesia能够处理多大的数据和访问量级?

Q: What volume of data can Mnesia handle? How is the performance?

会话存储层设计 Session storage

- 200K query per second and millions of records
- Mnesia能处理么？Can Mnesia handle this?
  - 能，没有持久存储的情况下。但是节点当机的时候会遇到全局锁，导致所有节点都会阻塞引发整体集群无法访问
  - Yes, without data persistence. But will get whole-table lock which will block all nodes when machine crashes.
  - 所以我们用了自带的分区功能来减少每个节点的数据量，让重启更快。
  - so we use fragments to reduce data volume of each node, to make the restart faster.

会话存储层设计 Session storage

```
create_table(fragmented) ->
    FragProps = [{node_pool, [node()]}, {n_fragments, 32}, {n_ram_copies, 1},
                 {hash_module, mnesia_frag_hash_partial}],
    mnesia:create_table(session,
                        [{frag_properties, FragProps},
                         {attributes, record_info(fields, session)}]);
create_table(normal) ->
    mnesia:create_table(session,
                        [{ram_copies, [node()]},
                         {attributes, record_info(fields, session)}]).
```

会话存储层设计 Session storage

- 前缀查询 prefix query
  - JabberID = {user, server, resource}
- 自定义分区哈希模块 customized fragment hash module
  - https://github.com/ericliang/ejabberd/blob/easemob/src/mnesia_frag_hash_partial.erl

Mnesia with fragment 的缺点

- 无法应对几倍的峰值，超过容量容易雪崩且难恢复
  - Requests pile up lead to VM out-of-memory when encountering peak rate, and hard to recover
- 无法自动伸缩，数据再平衡也不方便
  - no auto scaling and inconvenient to rebalance data
- 作为一个多租户的云服务，我们也引入了Codis :D

大纲 Outline

- 即时通讯云服务的架构演进
  - The evolution of our IM cloud service's architecture
- 关于Erlang你应该了解的
  - Something you should know about Erlang
- 一些工具和提示 Tools and tips
- 我们使用Erlang的原则 The principles we using Erlang

关于Erlang你应该了解的

- Something you should know about Erlang
  - Erlang的分布式设计 Distributed Erlang
  - 进程调度 Process scheduling
  - 谨慎使用进程组 Be cautious of process group

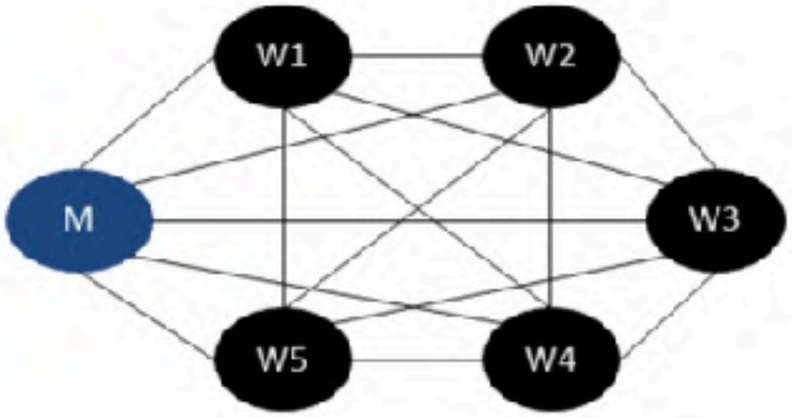
问：Erlang能够支撑多大规模的分布式集群?

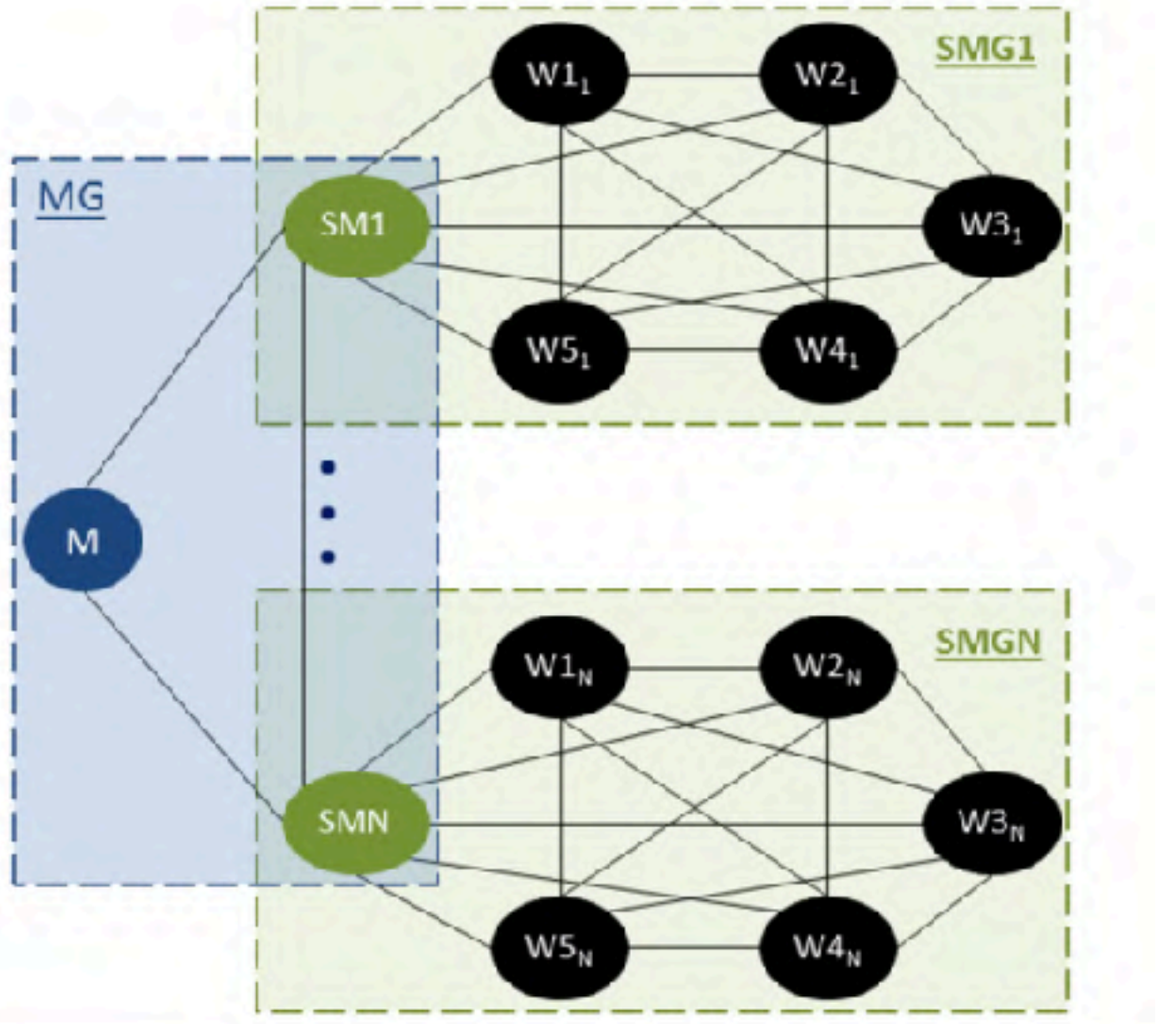
Q：What scale could an Erlang cluster be?

Erlang的分布式设计 Distributed Erlang

- Pid可以用来在网络中找到进程 Node for VM and Pid for process routing
  - Pid = spawn(Node, Module, Function, Args)
  - rpc:call(Node, Module, Function, Args)
- 全连接网络 Fully connected network
  - -hidden flag vs connect_all false
  - Natalia Chechina: not scalable onto 1000s of nodes

全连接网络 Full connected network





- Natalia Chechina: Scalable Distributed Erlang http://t.cn/RftOMeg

全连接网络 Fully connected network

- 生产环境单集群200多台机器是没有问题的
  - No problem with 200+ nodes in one cluster in production
- RPC通道堵塞，会导致调用进程挂起
  - but the RPC channel for busy_dist_port
  - 对系统的影响就是节点无法响应，进一步用户登陆超时
    - Node not responding and then user login timeout
- 调整参数 -zdbbl http://t.cn/RftWtCi

进程调度 Process scheduling

- 公平调度 fair
  - 抢占式 vs 协作式 Preemption vs Cooperation
- 时间片计算
  - Reduction counting
  - Queue with round-robin
  - http://t.cn/zRT28UI http://t.cn/zY1PB0e

问：公平调度有什么用?

Q：What does a FAIR scheduling mean?

进程调度 Process scheduling

- 公平，才有了对实时性的保障
  - Fair scheduling leads to real-time system.
- 调度的本质，是计算资源的分配
  - Scheduling is essentially computing resource allocation.
  - 进程成为了资源的单位
  - Process becomes resource unit.

线程池的例子 Examples on process pool

- 协作式 Cooperation
  - 性能不会下降，固定线程池即可
- 抢占式 Preemption
  - 随着系统进程数的增加，分配到后端连接池的CPU资源相对减少，需要增加进程数

- 使用Thrift RPC进行用户验证
- 使用ODBC连接池进行数据存取

谨慎使用进程组 Be cautious of process group

- 进程组的两个典型实现 Pg2 / Gproc
- 其实在做服务的自动注册和发现
  - is actually for service auto-registry and discovery
- Christopher Meiklejohn: Erlang pg2 Failure Semantics
  - http://t.cn/RfcczsD

谨慎使用进程组 Be cautious of process group

- 没有成熟认真地对待高并发高负载服务
  - Not serious on high performance heavy-load services
  - 网络闪断、分区、后端服务故障下很难恢复
    - Hard to recover from network flash off, partition, service outage
- 大多数pg2使用者也不认真
  - Neither is the pg2 user, usually.

谨慎使用进程组 Be cautious of process group

- Ekaf是一个Erlang的Kafka client，会用pg2把所有的调用方绑在一起，导致在网络闪断或者Kafka server出问题时，全集群的锁定。
- 解决方案：把进程信息改为本地存储
- https://github.com/ericliang/ekaf/tree/pg2l

~~谨慎~~ 不要使用pg2，和那些用了pg2的库。

~~Be cautious of~~ Do not use pg2, and libraries with it.

大纲 Outline

- 即时通讯云服务的架构演进
  - The evolution of our IM cloud service’s architecture
- 关于Erlang你应该了解的
  - Something you should know about Erlang
- 一些工具和提示 Tools and tips
- 我们使用Erlang的原则 The principles we using Erlang

一些工具和提示 Tools and tips

- Erlang VM would crash, use Erlang/OTP 18 [erts-7.3]
  - use kerl to build and install different instances
  - https://github.com/kerl/kerl
- TCP parameter: keepalive
  - due to bugs in load balancer of some cloud

```
thrift_client_util:new(Host, Port, user_service_thrift,
                       [{framed, true},
                        {connect_timeout, ?NET_TIMEOUT},
                        {recv_timeout, ?NET_TIMEOUT},
                        {sockopts, [{keepalive, true}]}])
```

一些工具和提示 Tools and tips

- 热升级好用，两分钟内全集群发布

```
lists:foreach(fun(Module) ->
                      code:purge(Module),
                      code:load_file(Module)
              end, UpdateModules),
```

- or

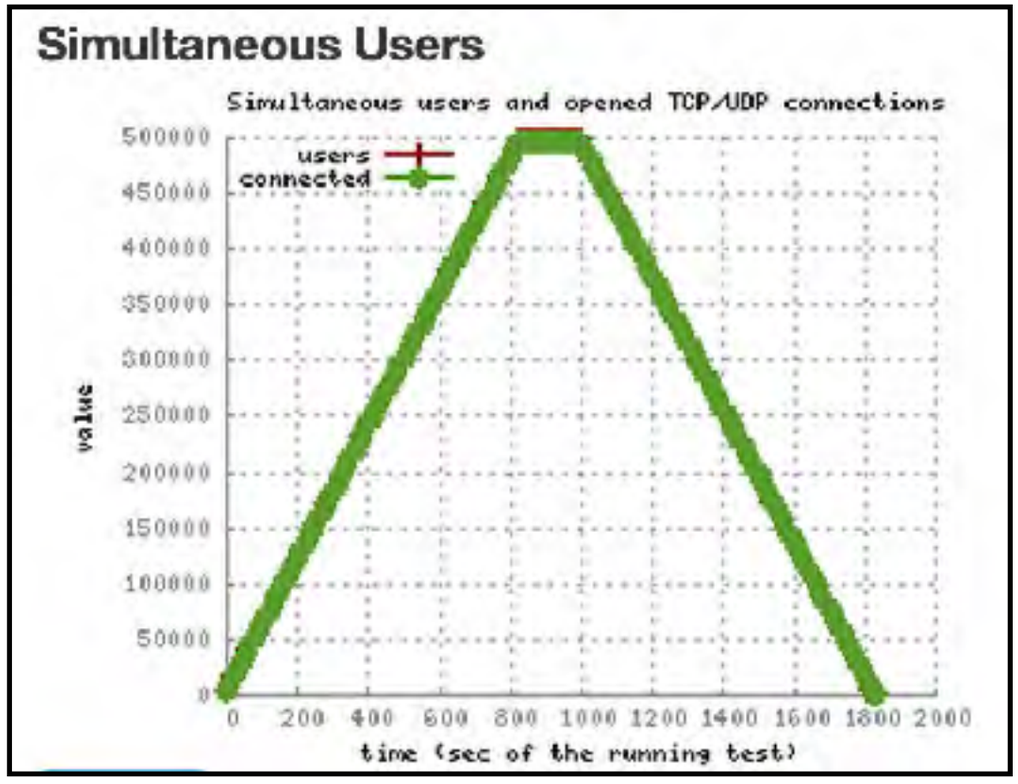
```
case release_handler:install_release(Vsn) of
    {ok, OldVsn1, []} ->
        release_handler:make_permanent(Vsn),
        "GOOD";
    Else ->
        Else
end
```

一些工具和提示 Tools and tips

- Lager
  - gen_event manager可能是瓶颈
  - 进程堵塞会导致整个VM调度性能下降
  - 用handler来进行限流不行，改用并行的独立Watcher进程

一些工具和提示 Tools and tips

- Dialyzer for static analysis
  - http://erlang.org/doc/man/dialyzer.html
- Tsung for stress test
  - http://tsung.erlang-projects.org



我们使用Erlang的原则 Principles

- 相信Erlang本身的设计 Trust the fundamental design of Erlang
  - 让系统尽量简单 Occam's Razor
    - Keep the system as simple as possible, no extra components if unnecessary
- 充分利用Erlang自身特性，先交付后优化
  - Take full advantages of Erlang, for faster delivery
- 大胆实践不信谣传
  - Bold in practice, don’t believe the rumors

我们使用Erlang的原则 Principles

- 拥抱开源，回报社区 By the community, for the community
  - 不发明轮子，优先使用开源软件
    - Do not reinvent the wheel, use open source project first
- 很多软件还不够成熟，用之前需要测试
  - Be aware of the immaturities of open-source softwares, test them before adoption.
- 也正因如此，再小的贡献也是贡献
  - For the same reason, make contributions, regardlessly big or small.

谢谢

欢迎加入环信！