

ERLANG WRIT LARGE

ADAM KOCOLOSKI

ERLANG FACTORY SF 2012



CLOUDANT

INTRODUCTIONS

- **Physicist by training**
- **CouchDB developer**
- **BigCouch architect**
- **Founder & CTO, Cloudant**



@kocolosk

adam@cloudant.com



BIGCOUCH AND CLOUDANT



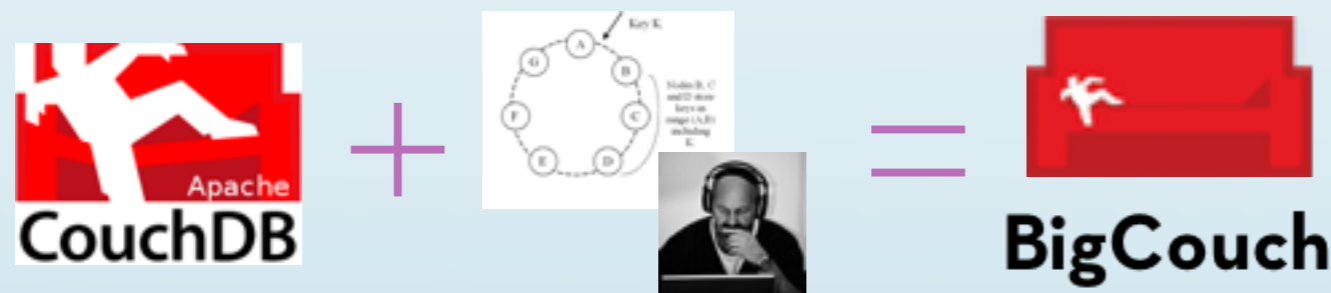
Document based

Marriage of Apache CouchDB and Amazon's Dynamo

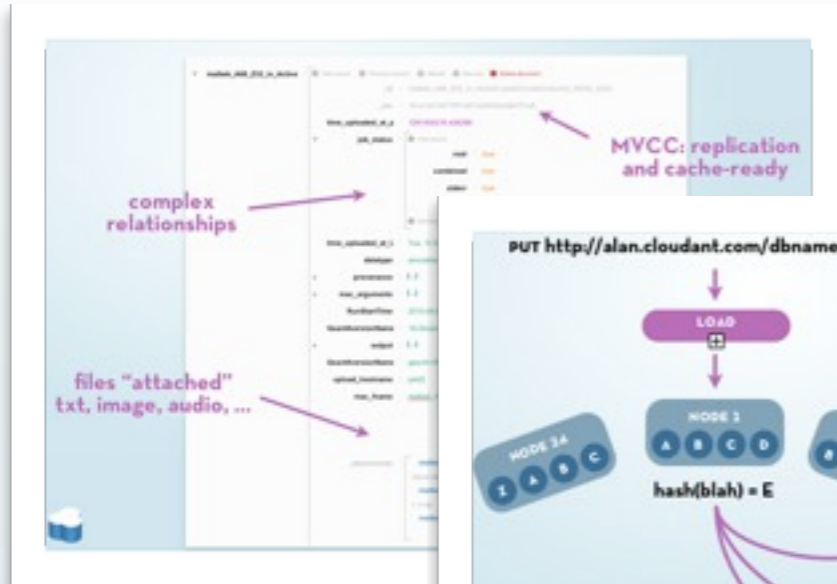
Transparent, elastic scaling of data across the cloud

Geographically distributed, multi-master with replication

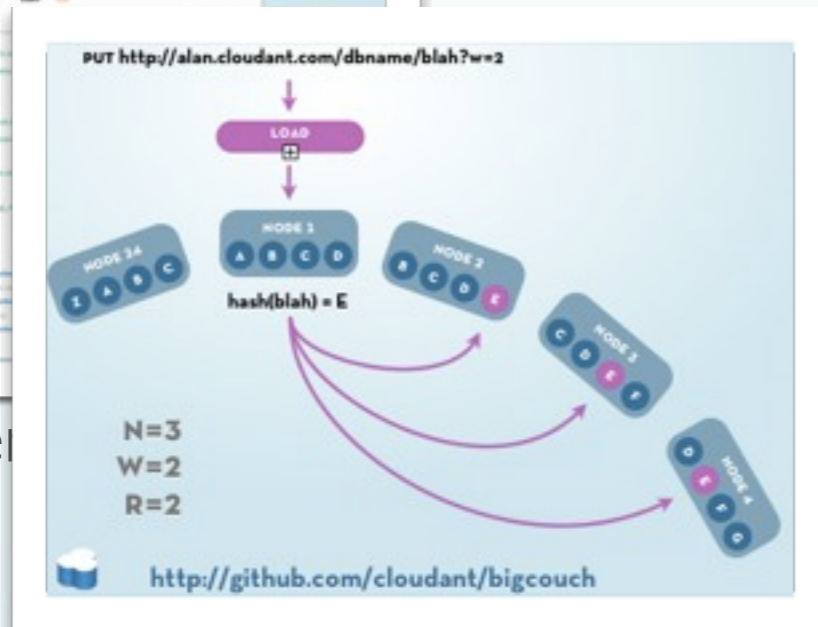
Durable and fault-tolerant with no single point of failure



BIGCOUCH AND CLOUDANT



Document



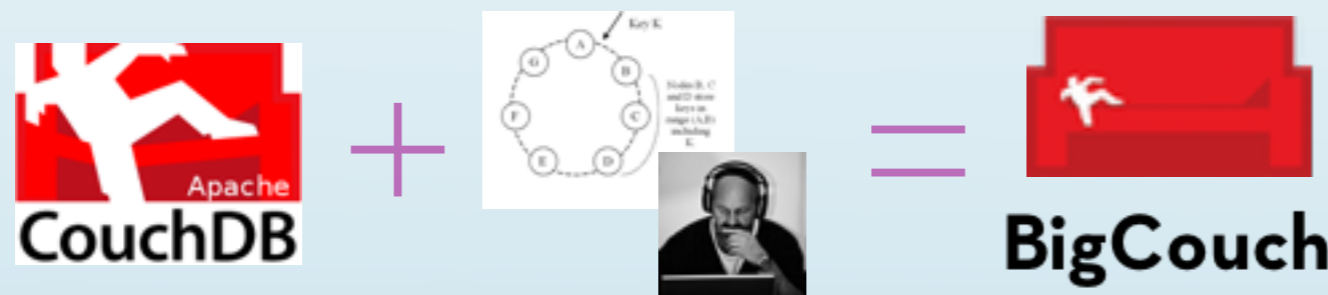
Clustered

Marriage of Apache CouchDB and Amazon's Dynamo

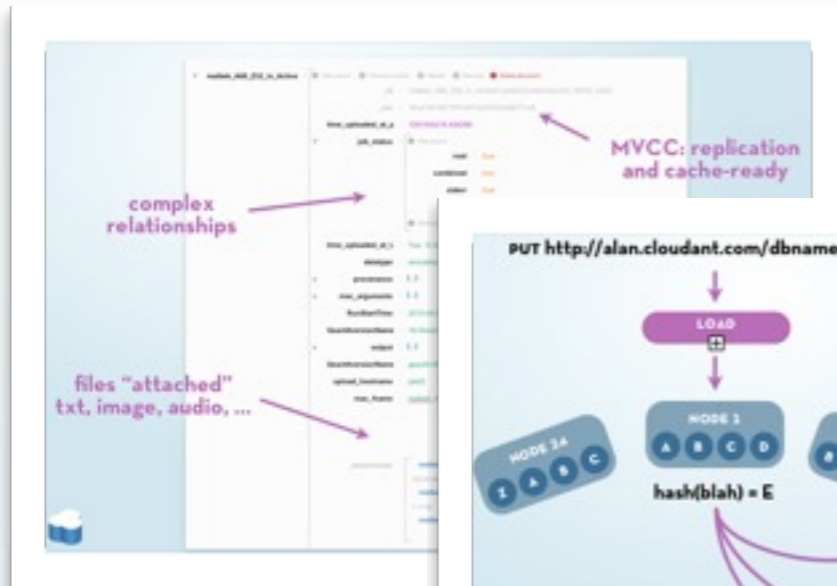
Transparent, elastic scaling of data across the cloud

Geographically distributed, multi-master with replication

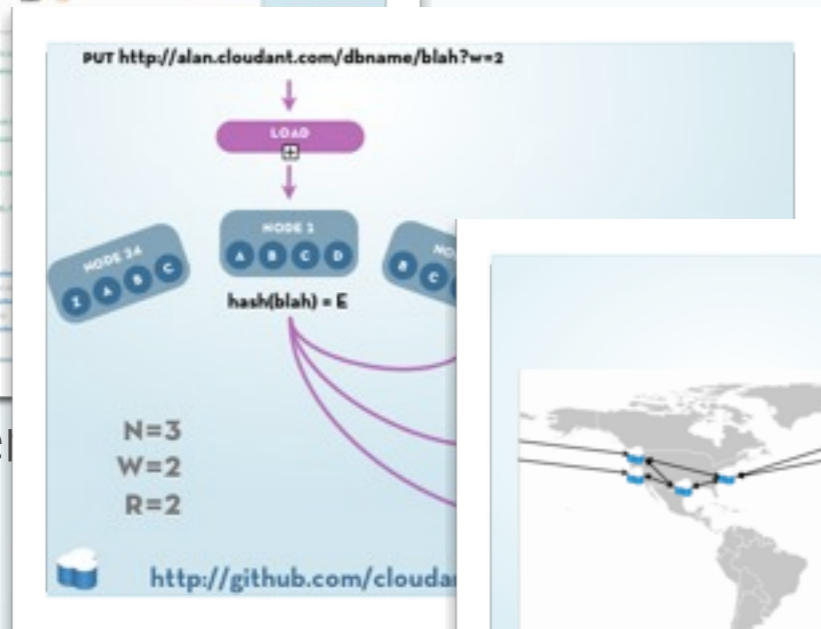
Durable and fault-tolerant with no single point of failure



BIGCOUCH AND CLOUDANT



Document



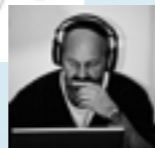
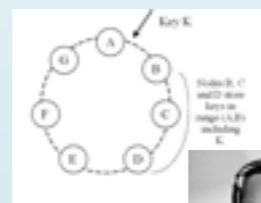
Cluster



Globally Distributed



+



=



BigCouch

Marriage of Apache CouchDB and Amazon's Dynamo

Transparent, elastic scaling of data across the cloud

Geographically distributed, multi-master with replication

Durable and fault-tolerant with no single point of failure



THIS TALK

- **Draw parallels between principles of good Erlang/OTP applications and good distributed datastores**
- **Dig deep into high-throughput distributed Erlang**
- **Share some war stories**



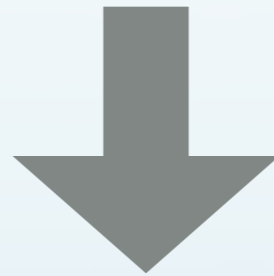
IMMUTABILITY

SUPERVISION

DISTRIBUTION

HOW RPC CALLS WORK

```
rpc:call(Node, my_cool_app, do_stuff, []).
```



```
gen_server:call({rex, Node}, {call, my_cool_app, ...}, infinity).
```



HOW RPC CALLS WORK

```
handle_call_call(Mod, Fun, Args, Gleader, To, S) ->
  RpcServer = self(),
  %% Spawn not to block the rpc server.
  {Caller, _} =
  erlang:spawn_monitor(
    fun () ->
      set_group_leader(Gleader),
      Reply =
        %% in case some sucker rex'es
        %% something that throws
        case catch apply(Mod, Fun, Args) of
          {'EXIT', _} = Exit ->
            {badrpc, Exit};
          Result ->
            Result
        end,
        RpcServer ! {self(), {reply, Reply}}
    end),
  {noreply, gb_trees:insert(Caller, To, S)}.
```



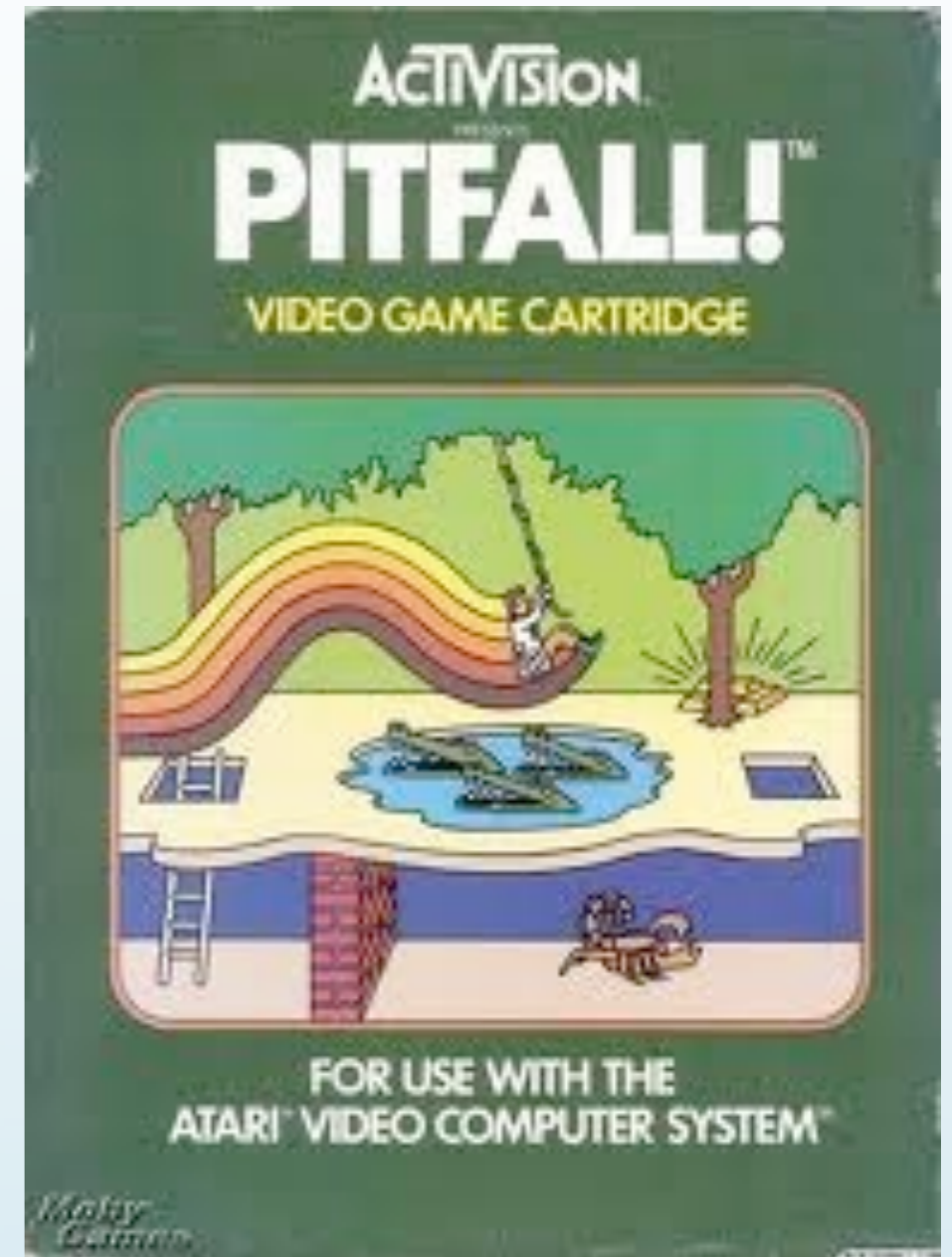
HOW RPC CALLS WORK

```
handle_info({Caller, {reply, Reply}}, S) ->
  case gb_trees:lookup(Caller, S) of
  {value, To} ->
    receive
      {'DOWN', _, process, Caller, _} ->
        gen_server:reply(To, Reply),
        {noreply, gb_trees:delete(Caller, S)}
    end;
  none ->
    {noreply, S}
  end;
```



PITFALLS

- Erlang deals with overloaded ports by suspending the sending process
- Communications to all nodes handled in the same server loop, so...
- Any overloaded / unresponsive node can suspend **all** rpc communications throughout your cluster



SOLUTIONS

- **Reply directly to caller from spawned process**
- **Use a non-blocking send**

```
nb_send(Pid, Msg) ->  
  case erlang:send(Pid, Msg, [noconnect, nosuspend]) of  
    noconnect ->  
      spawn(erlang, send, [Pid, Msg]);  
    nosuspend ->  
      spawn(erlang, send, [Pid, Msg]);  
    ok ->  
      ok  
  end.
```



<https://github.com/cloudant/rexi>

DISTRIBUTION SOCKETS

```
(dbcore@db1.testing123.cloudant.net)5> erlang:system_info(dist_ctrl).  
[{'dbcore@db6.testing123.cloudant.net',#Port<0.579>},  
 {'dbcore@db5.testing123.cloudant.net',#Port<0.1002>},  
 {'dbcore@db4.testing123.cloudant.net',#Port<0.1121>},  
 {'dbcore@db2.testing123.cloudant.net',#Port<0.1140>},  
 {'dbcore@db3.testing123.cloudant.net',#Port<0.1342>}]
```



SOCKET STATS

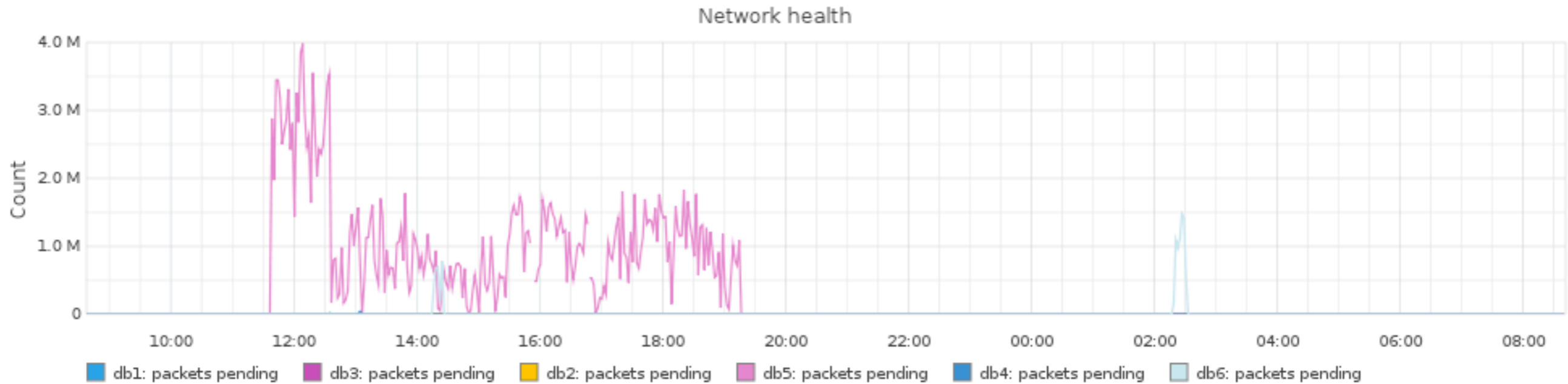
inet:getstat(Port).

```
{ok, [{recv_oct, 4293578011},  
      {recv_cnt, 5271209},  
      {recv_max, 4096226},  
      {recv_avg, 4888},  
      {recv_dvi, 830},  
      {send_oct, 507104463},  
      {send_cnt, 5146137},  
      {send_max, 3594585},  
      {send_avg, 11782},  
      {send_pend, 0}]}
```

Undocumented metrics FTW



FUN WITH SOCKET OPTIONS



```
lists:foreach(fun({_Node, Port}) ->  
    inet:setopts(Port, [{sndbuf, 33554432}, {recbuf, 33554432}])  
end, erlang:system_info(dist_ctrl)).
```

THANKS!