Building Scalable VoIP Applications with Erlang

Presented by James Aimonetti

About Me

James Aimonetti

- Senior Distributed Software Engineer
- I <3 Erlang
- Background in Comp Sci & Mathematics
- Sports Enthusiast (basketball, kettlebells, yoga)
- mc_ on Freenode in #2600hz, #erlang

Whistle

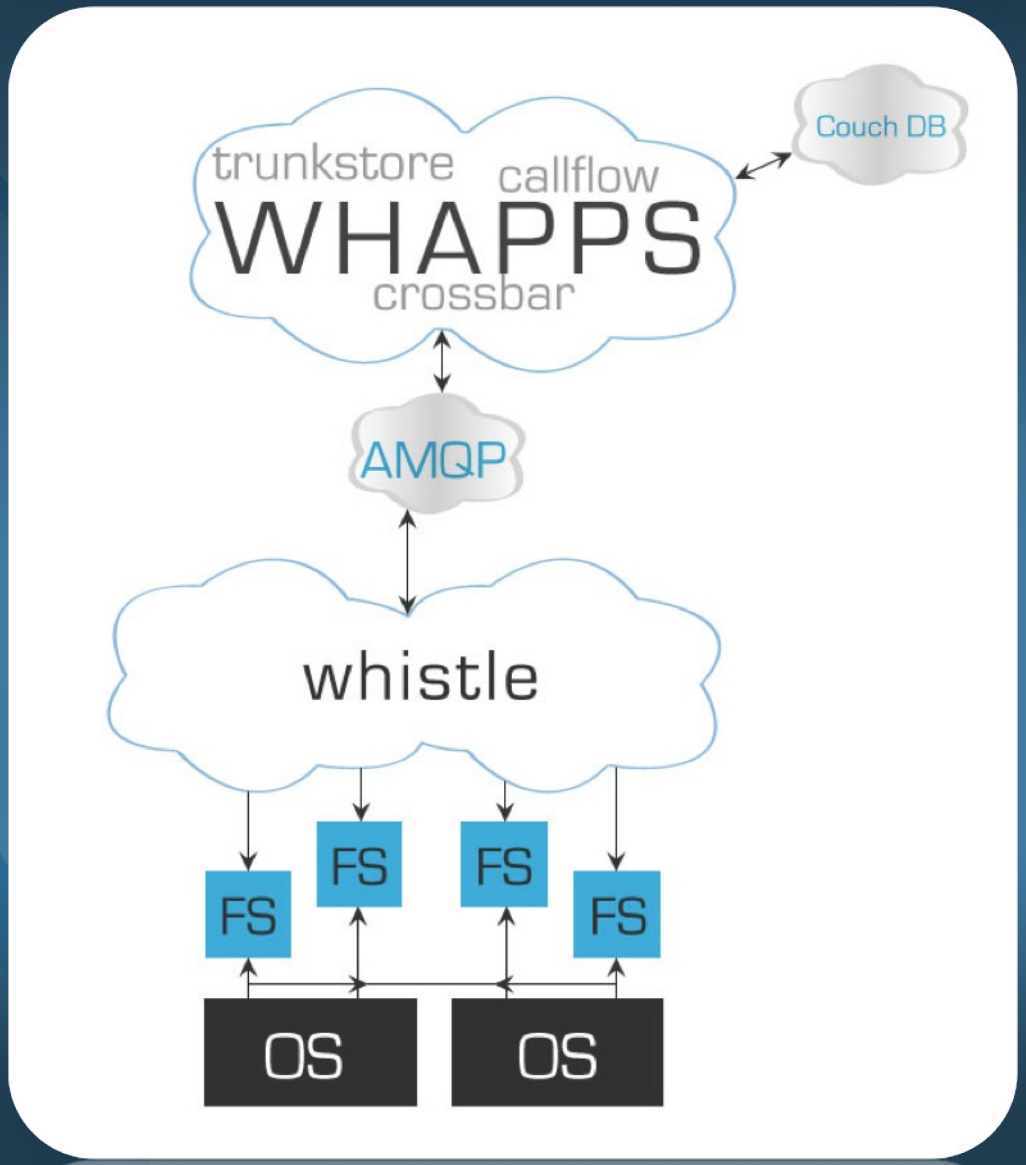
Signaling the start of next generation telephony

Whistle Goals

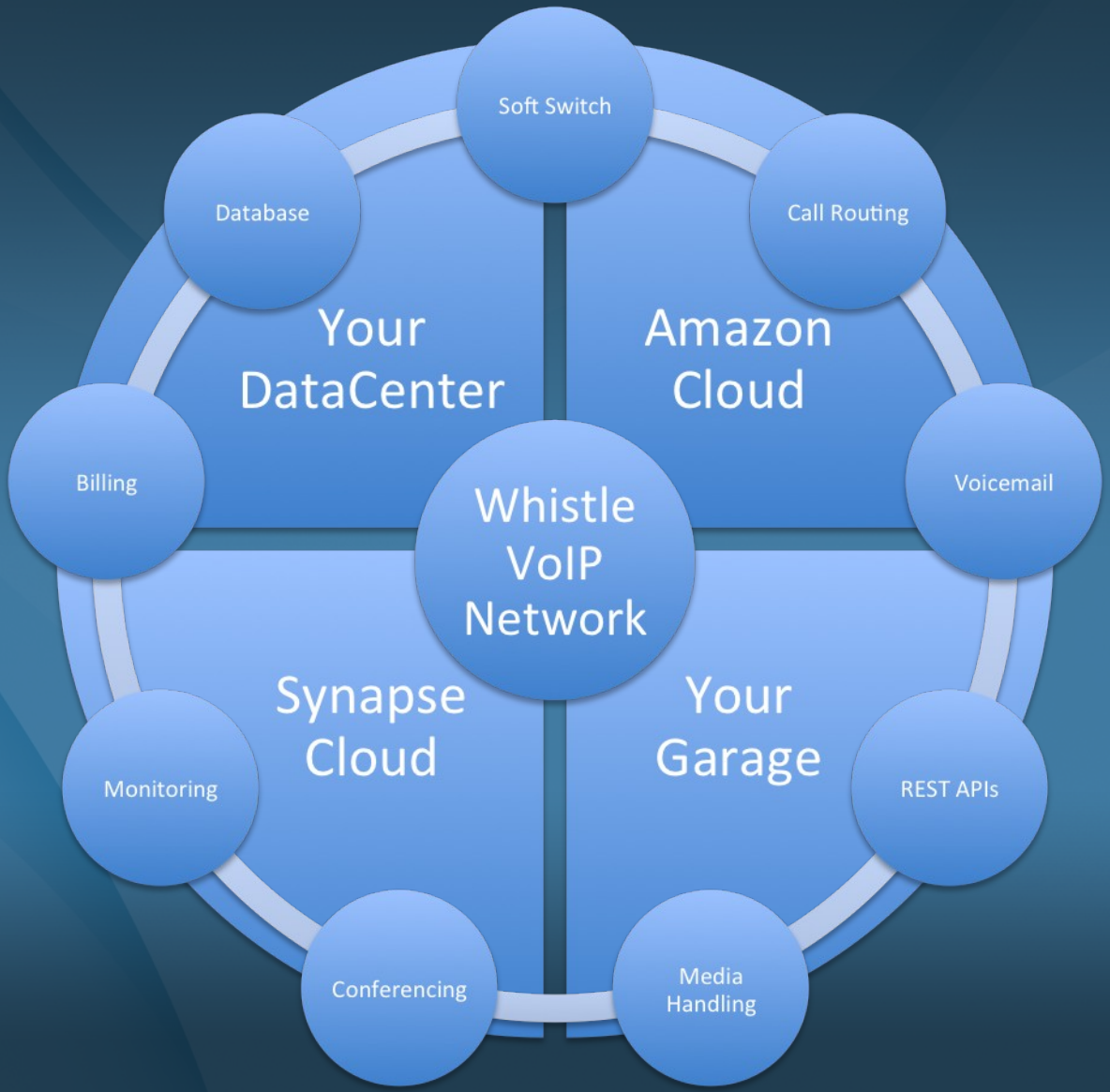
2600hz

- Easy To Scale
- Easy to be Redundant
- Easy to program applications at a high level

The Full Picture



Scalability



Whistle VoIP Platform

OpenSIPS

- SIP load balancer
- FreeSWITCH cluster can be changed transparently
- Redundant DNS for failover

Whistle VoIP Platform

FreeSWITCH

- Powerful built-in eventing system
- mod_erlang_event gives access
- Dumbed down and getting dumber

Whistle VoIP Platform

2600hz

Whistle (ecallmgr)

- Registers hooks with mod_erlang_event
- Converts FreeSWITCH events into Whistle AMQP API messages
- Receives Whistle AMQP API messages
- Converts AMQP messages to FreeSWITCH-specific instructions
- Other switches possible

Whistle VoIP Platform

Whistle (ecallmgr)

- Three start events
  - Authentication (directory)
  - Routing (dialplan)
  - Resource Request (origination)

WhApp Libraries

- whistle_couch
  - Abstract CouchDB (and BigCouch admin) access
  - Uses mochijson2-formatted json_objects
    - {struct, [{<<"key">>, <<"value">>}]} = {"key":"value"}
    - {struct, [{<<"key">>, [<<"v1">>, 2]}]} = {"key":["v1",2]}
  - Handles compaction automatically for you

WhApp Libraries

- whistle_amqp
  - Abstract AMQP access
  - Uses mochijson2-formatted json_objects as payloads
  - Handles connections/channels automatically.
  - whistle_api.erl validates and converts mochijson2-encoded terms to JSON payloads

Whistle Helper Modules

- props.erl
  - Proplists module using lists:keyfind for an extra speed boost
- whistle_util
  - Convert DID formats (E.164, NPAN, others)
  - Convert types (to_binary, to_list, to_float, to_integer)
- wh_cache
  - Simple caching mechanism
- wh_timer
  - Simple timing mechanism

Whistle Helper Modules

- wh_json
  - get_value/{2,3}
  - set_value/3
  - delete_key/2
- wh_log macros
  - Standardizing logging output to syslog
  - ?LOG(“Msg here”).
  - ?LOG(“Msg + ~s”, [<<”formatting”>>]).
  - ?LOG(CallID, “Msg + ~s”, [formatting])
  - Expands to “~s|log|~s.~b (~w):” ++ MsgStr, [CallID, ?MODULE, ?LINE, self() | Args]
  - CallID defaults to <<”0000000000”>>, otherwise pulled from process dictionary: get(callid)

WhApps

- WhApps live in an Erlang VM together
- All have access to same whistle_couch and whistle_amqp
- Typically one WhApp type per VM
  - High-level stuff is locally registered
- whapps_controller:start_app/1, stop_app/1

Let's Build One - Registrar

- Subscribe for authentication requests
  - Lookup SIP credentials
  - Publish authentication response
- Subscribe for successful registrations
  - Store registration
- Subscribe for registration queries
  - Publish query responses
- Expire registrations

Let's Build One - Registrar

- Directory Structure
  - $WHISTLE/whistle_apps/apps/registrar
    - src
      - registrar.erl
      - registrar.app.src
      - registrar_app.erl
      - registrar_sup.erl
      - reg_server.erl
    - ebin
    - priv
      - couchdb
        - » View documents

Let's Build One - Registrar

reg_server.erl sequence of events

- init/1 – ensure Dbs exist, load updated CouchDB design documents
- handle_info(timeout, State) – prime the cache, start a cleanup timer, create and bind an AMQP queue

Let's Build One - Registrar

reg_server.erl sequence of AMQP event

- handle_info({_, #amqp_msg{}}, State) – spawn a process to figure out which event the message represents
- process_req(MsgType, JsonObject, State) – pattern match the MsgType, validate the JsonObject, and handle the request if valid

Let's Build One - Registrar

Code Samples:

- Setup AMQP:

```
Q = amqp_util:new_queue(?REG_QUEUE_NAME, [{exclusive, false}]),
amqp_util:bind_q_to_callmgr(Q, ?KEY_REG_SUCCESS),
amqp_util:bind_q_to_callmgr(Q, ?KEY_REG_QUERY),
amqp_util:bind_q_to_callmgr(Q, ?KEY_AUTH_REQ),
amqp_util:basic_consume(Q, [{exclusive, false}])
```

Let's Build One - Registrar

Code Samples:

- Prime Cache:

```
{ok, Docs} = couch_mgr:all_docs(?REG_DB),
Expires = whistle_util:current_tstamp() + 3600,
[ wh_cache:store_local(Cache, wh_json:get_value(<<"id">>, View), Expires) || View <- Docs ]
```

- Cleanup registrations

```
Now = whistle_util:current_tstamp(),
Expired = wh_cache:filter_local(Cache, fun(_, V) -> V < Now end),
lists:foreach(fun({K,_}) ->
  {ok, D} = couch_mgr:open_doc(?REG_DB, K),
  couch_mgr:del_doc(?REG_DB, D),
  wh_cache:erase(Cache, K)
end, Expired).
```

Let's Build One - Registrar

Let's See The Whole Server

More Information

- http://github.com/2600hz/whistle
- http://wiki.2600hz.com/
- http://deploy.2600hz.com
- http://store.2600hz.com