A Cloud as an Interface

ErLounge, San Francisco CA USA
May 5th 2011

Amazon Web Services

- Fault Tolerance for an Instance
- Light-Weight Integration of Diverse Source Code Grouped into Services
- Scalable Service Communication

A Cloud Computing Software Solution

- Private Usage with Implicit Security
- Public Usage with Scalable Connection Handling
- Extends the Erlang Process/Actor Model into C/C++, Java, Python, Ruby, …
- Provides Dynamic Configuration
- Fault Tolerant Supervision of Supported Programming Languages

Flexible Integration

- CloudI API
- HTTP
- ZeroMQ
- Databases
  - CouchDB
  - Memcached
  - PostgreSQL
  - MySQL
  - TokyoTyrant

Control + Isolation

- CloudI Job API for Dynamic Configuration
- Access Control List (ACL) Configuration Controls All Service Communication
- Services are Managed in the Same Way as Child Processes of an Erlang OTP Supervisor

Where to Start

- http://cloudi.org/faq.html
- http://groups.google.com/group/cloudi-questions

Code Reuse: Erlang Trie Data Structure

- dict Module Interface Supported
- List of Integers (string) to any Type Mapping
- Lookup Performance Equivalent to the Process Dictionary (often the fastest Erlang data structure)
- Useful for String Lookups that are both more Efficient and more Scalable than ETS
- Used in CloudI's list_pg.erl and list_pg_data.erl to Provide a more Scalable pg2 Module
- https://github.com/okeuday/trie

Code Reuse: Erlang Native UUID Generation

- UUID Versions 1, 3, 4, and 5 Implemented
- Version 1 Generation is Unique to an Erlang pid and Distributed Erlang Node
- Version 1 is Used to Uniquely Identify CloudI Service Messages
- https://github.com/okeuday/uuid