Compiling Abstract Specifications into Concrete Systems – Bringing Order to the Cloud

ANCOR
- Automated eNterprise network COmpileR -

Ian Unruh, Alexandru G. Bardas,
Rui Zhuang, Xinming Ou, Scott DeLoach
Cloud Users - Desired Features

• **Flexibility**
  – access to the raw resources *e.g.*, compute, storage

• Reliable **automation** capabilities
  – Non-scenario-dependent
  – Automatic deployment and maintenance
  – Dynamic cluster expansion and contraction

• **Migration** between different cloud providers
  – Capturing infrastructure and application requirements in a specification
Current Cloud Computing Offerings

• Allow customers to decide how much management they want:
  
  o Infrastructure as a Service (IaaS)
    • e.g., Amazon Web Services, OpenStack
  
  o Platform as a Service (PaaS)
    • e.g., Heroku, Microsoft Azure
  
  o Software as a Service (SaaS)
    • e.g., SalesForce, Google Apps
Current Cloud Computing Offerings

• Allow customers to decide how much management they want:
  
  o Infrastructure as a Service (IaaS)
    • e.g., Amazon Web Services, OpenStack
  
  | ✔  | Flexibility       |
  |    | (access to the raw resources) |

  | ✗  | Automation       |
  |    | (non-scenario-dependent) |

  | ✗  | Migration        |
  |    | (capturing requirements in a specification) |
Current Cloud Computing Offerings

- Allow customers to decide how much management they want:
  - Software as a Service (SaaS)
    - *e.g.*, SalesForce, Google Apps

<table>
<thead>
<tr>
<th></th>
<th>Flexibility</th>
<th>Automation</th>
<th>Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>(access to the raw resources)</td>
<td>(non-scenario-dependent)</td>
<td>(capturing requirements in a specification)</td>
</tr>
</tbody>
</table>
Current Cloud Computing Offerings

• Allow customers to decide how much management they want:

  o Platform as a Service (PaaS)
    • e.g., Heroku, Microsoft Azure

<table>
<thead>
<tr>
<th>Flexibility</th>
<th>✔ Automation</th>
<th>✗ Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>(access to the raw resources)</td>
<td>(non-scenario-dependent)</td>
<td>(capturing requirements in a specification)</td>
</tr>
</tbody>
</table>
Proposed Solution

- An **abstraction** that captures *what* a cloud user needs instead of low-level details on *how* to implement those needs

- There must be a process to **automatically** compile the abstraction into a valid concrete system
Proposed Solution

• An abstraction that captures *what* a cloud user needs instead of low-level details on *how* to implement those needs

• There must be a process to automatically compile the abstraction into a valid concrete system
ANCOR

What?

- Requirements Model
- Operations Model
- System Model

How To...

Compilation Process

Cloud Platform
(OpenStack)
**ANCOR**

**WHAT?**
- Requirements Model
- System Model

**Operations Model**
- Compiler
- Configuring
- Provisioning
- Orchestration (Mcollective)
- CMT (Puppet)
- OpenStack API Library (Fog)

**COMPIlATION PROCESS**
- Cloud Platform (OpenStack)
ANCOR

WHAT?

Compiler

Cloud Platform (OpenStack)

Configuring

Provisioning

Requirements Model

System Model

Operations Model

Orchestrator (Mcollective)

OpenStack API Library (Fog)

CMT (Puppet)

Conductor

Configuration Management Tools (CMTs)

ANCOR COMPILATION PROCESS

HOW TO...
Deploying an eCommerce Website

Scalable and highly available eCommerce website architecture
ANCOR Requirement Modeling Language (ARML)

```plaintext
1. goals:
2. ecommerce:
3.   name: eCommerce frontend
4.   roles:
5.     - weblb
6.     - webapp
7.     - worker
8.     - work_queue
9.     - db_master
10.    - db_slave

11. roles:
12.   weblb:
13.       name: Web application load balancer
14.       min: 2
15.       is_public: true
16.       implementations:
17.         default:
18.         profile: role::weblb::default
19.       exports:
20.         http: { type: single_port, protocol: tcp, number: 80 }
21.       imports:
22.         webapp: http

23. webapp:
24.   name: Web application
25.   min: 3
26.   implementations:
27.     default:
28.     profile: role::webapp::default
29.   exports:
30.     http: { type: single_port, protocol: tcp }
31.   imports:
32.     db_master: querying
33.     db_slave: querying
34.     work_queue: redis

35. worker:
36.   name: Sidekiq worker application
37.   min: 2
38.   implementations:
39.     default:
40.     profile: role::worker::default
41.   imports:
42.     db_master: querying
43.     db_slave: querying
44.     work_queue: redis

45. work_queue:
46.   name: Redis work queue
47.   implementations:
48.     default:
49.     profile: role::work_queue::default
50.   exports:
51.     redis: { type: single_port, protocol: tcp }

52. db_master:
53.   name: MySQL master
54.   implementations:
55.     default:
56.     profile: role::db_master::default
57.   exports:
58.     querying: { type: single_port, protocol: tcp }

59. db_slave:
60.   name: MySQL slave
61.   implementations:
62.     default:
63.     profile: role::db_slave::default
64.   min: 2
65.   exports:
66.     querying: { type: single_port, protocol: tcp }
67.   imports:
68.     db_master: querying
```

eCommerce Website Requirements Specification
Ancor Requirement Modeling Language (ARML)

eCommerce Website Requirements Specification
Ancor Requirement Modeling Language (ARML)

goals:
  ecommerce:
    name: eCommerce frontend
    roles:
      - weblb
      - webapp
      - worker
      - work_queue
      - db_master
      - db_slave

ecommerce:
  name: Sidekiq worker application
  roles:
  - worker
  - webapp
  - db_master
  - db_slave
Ancor Requirement Modeling Language (ARML)

eCommerce Website Requirements Specification
Ancor Requirement Modeling Language (ARML)

```
roles:
  weblb:
    name: Web application load balancer
    min: 2
    is_public: true
    implementations:
      default:
        profile:
          "role::ecommerce::
           weblb::default"
    exports:
      http: { type: single_port,
               protocol: tcp,
               number: 80 }
    imports:
      webapp: http
```
ANCOR Workflow

1. Passing the IT system specification to ANCOR
ANCOR Workflow

1. Passing the IT system specification to ANCOR
2. Specification is stored in the Requirements Model
ANCOR Workflow

1. Passing the IT system specification to ANCOR
2. Specification is stored in the Requirements Model
3. Compute low-level details of the system and store them in the System Model
ANCOR Workflow

1. Passing the IT system specification to ANCOR
2. Specification is stored in the Requirements Model
3. Compute low-level details of the system and store them in the System Model
4. Start deploying the system on the cloud infrastructure using cloud platform and CMT APIs
ANCOR Workflow

1. Passing the IT system specification to ANCOR
2. Specification is stored in the Requirements Model
3. Compute low-level details of the system and store them in the System Model
4. Start deploying the system on the cloud infrastructure using cloud platform and CMT APIs
5. Update the System Model so it is always consistent with the deployed system
ANCOR Benefits

✓ Infrastructure and application “portability”
✓ Up-to-date application dependencies
✓ Building highly dynamic systems
✓ Automated fine-grained firewall configuration
✓ Security assessments
✓ Performance evaluations
✓ Creating customized PaaS
ANCOR

• Current implementation and more information:

http://arguslab.github.io/ancor/
Conclusion

• Separating user requirements from the implementation details has the potential of changing the way IT systems are deployed and managed in the cloud.

• ANCOR – framework that captures the high-level requirements and translates them into a working IT system on a cloud infrastructure.

http://arguslab.github.io/ ancor/
LISA Labs

Today 4:00PM – 5:30PM

Alex Bardas: bardasag@ksu.edu
Related Work

• **Automation Solutions**
  – Automating instance management *e.g.*, AWS OpsWorks
  – Deploying/migrating applications on different cloud providers *e.g.*, Cliqr, Cloud Velocity, CloudSwitch
  – Managing and automating instances deployment *e.g.*, Right-Scale, Service-Now

• **Abstraction Approaches** (PaaS specific)
  – Windows Azure Service Definition Schema (*.csdef*), Google AppEngine YAML-based specification

• **Managing Infrastructure** (support CMT integration)
  – OpenStack Heat, AWS CloudFormation, Terraform
More Related Work

• Docker container-based solutions:
  – Maestro-NG, Flynn, Deis, OpenShift, etc.

• Ubuntu Juju:
  – Works at a similar abstraction level
ANCOR vs. Juju

• Similarities:
  • Work at a similar abstraction level
  • Have a way of capturing the dependencies between software applications (services)

• Differences:
  – Using existent CMT modules and workflow:
    • ANCOR: minimal changes
    • Juju: usually, significant changes (integration with Juju Tools)
  – Dependent services
    • ANCOR: more “centralized” management scheme
    • Juju: negotiation scheme
  – Current feature sets e.g., OS support