Overview of Big Data Reference Architecture
Software and Demonstration

Dr. Gregor von Laszewski
Assistant Director of Community Grids Lab, Adjunct Associate Professor
Indiana University

NIST Campus
Gaithersburg, Maryland
June 1, 2017
Presentation Overview

- Volume Presentation Outline
- Volume 1, Definitions (Nancy Grady, SAIC)
- Volume 2, BD Taxonomies (Nancy Grady, SAIC)
- Volume 3, Use Cases and General Requirements (Geoffrey Fox, Indiana University)
- Volume 6, Reference Architecture (David Boyd, InCadence Corp.)
- Volume 4, Security and Privacy (Arnab Roy, Fujitsu; Mark Underwood, AVP, Strategic Initiatives, Controls and Countermeasures)
- Volume 8, Reference Architecture Interface (Gregor von Laszewski, Indiana University)
- Reference Architecture Software Implementation Environment and Demonstration (Gregor von Laszewski, Indiana University)
- Volume 7, Standards Roadmap (Russell Reinsch, Center for Government Interoperability)
- Volume 9, Adoption and Modernization (Russell Reinsch, Center for Government Interoperability)
NBDIF Volume Overview

Vol. 1 BD Definitions
Defines common language

Vol. 2 BD Taxonomies
Hierarchy of NBDRA components

Vol. 3 Use Cases & Vol. 5 Arch Survey
Info gathered; requirements extracted

Vol. 6 NBDRA
Developed NBDRA

Vol. 4 S&P
Interwoven topics of S&P examined

Vol. 7 Standards Roadmap
Examine standards wrt NBDRA

Vol. 8 NBDRA Interfaces
Implementation of NBDRA

Vol. 9 Adoption & Modernization
Volume Presentation Outline

• For each volume
  – Scope of the volume
  – Brief recap of version 1
  – Highlights of version 2 accomplishments
  – Summary of version 2 areas needing contributions
  – Topics that could be considered for version 3
Reference Architecture Software Implementation
Environment and Demonstration

• **Cloudmesh provides a first reference implementation**
  – Features include IaaS, Hadoop, and software stack deployment.
  – It was tested based on Application from Use Case document.
  – Code is hosted in github and is available.

• **Focus on Cloudmesh command shell and REST service as it is**
  – Scriptable
  – Interpretable into other frameworks
  – Accessible through other frameworks via REST.

• **Disclaimer:** we move from our original cm implementation to cms to distinguish the two efforts. The new implementation can use the NIST specification and generates a REST service automatically.
Cloudmesh Architecture

- Abstraction essential to Cloudmesh design
- Abstractions at different levels and interaction points
  - IaaS
  - Container
  - HPC
  - PaaS
- Virtual Cluster
- Integration with Providers
  - IU OpenStack, NSF Chameleon cloud, NSF Comet, AWS, Azure, SLURM/XSEDE, …
- Used by hundreds of users
Cloudmesh Layered Architecture

- Easy extensibility
- Developed with command shell in mind
- Developed with REST in mind
- Horizontal Integration
  - Access – Data – Compute
- Vertical Integration
- `Security - Choreography`
Deployment Abstractions

- Possible interaction with different DevOps frameworks
- Leveraging large DevOps community

- Warning we found that there are many DevOps "templates" but not all of them are usable:
  - lack generality
  - do not work
  - too complex
  - not properly documented
Continuous Improvement vs. Continuous Deployment via DevOps

- DevOps is integrated
- Leads to improvement when not only targeting application but also deployment environment.
Simple Interface Usecase: Boot a vm on
Simple Interface Use Case: Boot & Provision
Phase 1: Interface Objects
Specification -> Reference implementation

1. Specification
2. Cloudmesh schema generates ...
3. ... a valid schema from the specification
4. The schema is used to automatically generate a REST service
Showcase document

- https://laszewski.github.io/papers/NIST.SP.1500-8-draft.pdf
Account Management
Account Management
Example: Extension to Architecture

• Accounting across hybrid services
• Integrating of accounting records for individuals (in case group account does not provide this feature)
• User Management issues
  – Removal of “Dracula Users”: I suck you dry and consume all your hours as I will ignore your policies will fully (yes, they do exist)
  – Removal of “Uniformed User”: let the know what an experiment costs upfront before you start it.
• Provider Management Issues
  – Provide feedback to providers: We found that some providers gave us incomplete information in regards to their accounting practice
  – Comparison of cost between providers
• Application Benchmarking
  – If we do make it too easy some will ignore alternatives, Expose benchmarking results to the community
Account Management
Account management

- Register
- Deposit
- Use
- Deactivation
Fingerprint Example

Gregor von Laszewski, laszewski@gmail.com

Badi’ Abduhl-Wahid
Fingerprint Application

- Requires
  - Application knowledge
  - Deployment/DevOps knowledge

- What if application user could do also the deployment?
  - Use newest software
  - Use newest hardware
  - Benchmark different setups
Use Case Fingerprint: Deployment is complex
$ emacs cloudmesh.yaml
$ cms default cloud=NAME
$ cms default image=NAME
$ cmd default flavor=NAME
$ cms vm boot
$ cms vm login
$ cms vm delete
### Cloudmesh Shell – Manage Hybrid Clouds

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{cms aws boot}$</td>
<td></td>
</tr>
<tr>
<td>$\text{cms vm boot}$</td>
<td></td>
</tr>
<tr>
<td>$\text{cms default cloud=chameleon}$</td>
<td></td>
</tr>
<tr>
<td>$\text{cms vm boot}$</td>
<td></td>
</tr>
<tr>
<td>$\text{cms default cloud=IUCloud}$</td>
<td></td>
</tr>
<tr>
<td>$\text{cms vm boot}$</td>
<td></td>
</tr>
</tbody>
</table>

- Boot Cloud A
- Boot Cloud B
- Boot Cloud C
Cloudmesh Shell – Create a Hadoop Cluster

- Set cloud
- Define cluster
- Define hadoop Cluster
- Sync definition to db
- Deploy the cluster

$ cm default cloud=chameleon
$ cm cluster define - -count=10 - -flavor=m1.large
$ cm hadoop define spark
$ cm hadoop sync # ~30 sec
$ cm hadoop deploy # ~ 7 min
Cloudmesh Shell – Create a Hadoop Cluster

$ cm default cloud=IUCloud
$ cm cluster define - -count=10
   - -flavor=m1.large

$ cm nist fingerprint # ~ 30 min

• Set cloud
• Define cluster
• Run NIST usecase

Additional resources:
https://github.com/cloudmesh/classes/blob/master/docs/source/notebooks/fingerprint_matching.ipynb