Building Software Defined Systems on HPC and Clouds

Hyungro Lee
School of Informatics and Computing
Indiana University
Bloomington, IN 47405
lee212@cs.indiana.edu

Abstract

Systems for modern applications require dynamic computing resources for various workload with a different type of datasets and a collection of software work together to complete tasks on IaaS, PaaS, SaaS or FaaS. Building a cluster of virtual machines is inevitable to accelerate computation speed for these applications but there are challenging tasks to deploy, configure and manage systems in a virtualized environment or high performance computing. DevOps tools provide automated software deployment and continuous integration yet computing environments and resources for applications need to be prepared manually by system administrator and application developer because of dependency hell. Template-based infrastructure provisioning permits an repeatable build of a virtual system but the execution of applications is separated from the system built which has no trace of workloads. In this dissertation, we combine these two approaches to fully automate from preparing environments to running workload of applications in a structured way which results in building software defined systems (SDS) on HPC and Clouds.

Semantic templates are introduced to complete virtual cluster provisioning with an example of big data applications and role-based DevOps tools are examined to deploy software stacks. There are several tools and libraries to be implemented with several container technologies to enable dynamic computing environments on both HPC and Clouds. In addition, specific application domains i.e. bioinformatics are explored to demonstrate practical experiences of applying use cases with the philosophy of software defined systems.

Keywords: software defined systems, big data, template deployment, configuration management, scientific applications, virtualization

1 Introduction

From Infrastructure-as-a-Service to Functions-as-a-Service, many efforts have been made to provide computing resources in virtualized environments but with less complication of building infrastructure and preparing environments. Lightweight linux containers are widely adopted in supporting interdisciplinary field of research and collaboration because of its kernel level of an isolated environment. IaaS is a still best approach to operate fine-grained resource provisioning regarding to CPU, memory, storage and network. This thesis will explore the rapid evolution of virtualization technologies from IaaS to serverless computing with container technologies to find optimized configuration of systems with a general software deployment. The next generation system must utilize DevOps tools for deploying software stacks on a cluster of virtual machines and infrastructure provisioning for supporting various applications. In recent years, building big data clusters have become an inevitable task for performing analysis with the increased computational requirements for large datasets and the anticipated systems will be perfect for these situations to every dimension of infrastructure provisioning and software deployment.

Supporting big data analytics is more difficult for a few reasons: (1) big data applications run with large datasets and a collection of software, (2) building and managing big data deployments on clusters require expertise of infrastructure and (3) Apache Hadoop based big data software stacks are not suitable for HPC. In an effort to resolve the first two issues, pulic datasets and data warehouse on the cloud have offered to ensure instant data access with SQL support and enterprise big data solutions have hosted by cloud providers e.g. Amazon, Google, and Microsoft to save time on deploying multiple software stacks without facing installation errors. These services, however, are only available to their customers and make hard to swtich to another when applications and pipelines are built on top of the services. Pre-developed infrastructure for big data stacks are only suitable for particular use cases and are unable to customize or re-define by users. There are efforts to simplify a deployment with a specification such as automated deployment engines using TOSCA (Wettinger, Breitenbücher, and Leymann, 2015), but they do not integrate multiple clouds with a cluster deployment or a workload execution. The bigdata deployment on clusters require more than single software to install and configure with different roles i.e. masters and workers. If it is built on virtualized resources, scailing up or down is necessary to maximize resource utilization but with exceptional performance.

These issues can be resolved using a template deployments for infrastructure and software which uses YAML or JSON documents to describe resources and tools to install and configure. For example, Amazon OpsWorks uses Chef to deploy software stacks and Cloudformation uses YAML document to deploy Amazon resources. Similarily, Microsoft Resource Manager Templates uses JSON document to deploy Azure resources and Google Cloud deployment Manager uses python or Jinja2 templating language to deploy Google Cloud platform resources. OpenStack heat is originated from AWS Cloudformation to deploy resources but extendedd with the integration among openstack services e.g. Telemetry for autoscaling. These templates have been used for infrastructure deployment and software installation with input parameters which enables reproducible deployment on virtual environments. We extend the use of a template with workload execution recording to track workflow steps and replicate results and also a role based deployment for building clusters. This will be beneficial to share scientific pipelines which typically contain complicated and long-running processes. Our approach is to record status and results of components in the workflow while it is running and store the execution information in templates for later use. This allow users to re-run the workflow on different systems but start from where the workflow stopped without executing whole processes again.

This proposal consists of the following sections. First, background introduces basics of a template deployment with big data software stacks, binary containers for hpc, template use cases for public clouds and bioinformatics. Next, design section provides a prototype of a template deployment and checkpoint/restart in user space (CRIU) regards to big data software stacks with clusters. Implementation section demonstrates plans to apply the big data template deployments towards clouds and HPC, such as amazon, azure, openstack, google compute and Slurm with integrated specifications. Schedule section provides todo lists with esitmated timelines to be completed. Last, summary section outlines final dissertation with brief idea.

1.1 Thesis Statement

Software defined systems with DevOps and Template infrastructure provisioning is an effective way of enabling big data software stacks on the cloud and hpc with container technologies.

2 Background

2.1 Software Deployment for dynamic computing environments

Software development has evolved with rich libraries and building a new computing environment requires set of packages to be successfully installed with minimal efforts. The environment preparation on different infrastructure and platforms is a challenging task because each preparation have individual instructions which build a similar environment, not identical environment. Traditional method of software deployment is using shell scripts to define installation steps with a system package manager command such as apt, yum, dpkg, dnf and make but it is not suitable to deal with large number of packages actively updated and added to community in a universal way. Python Package Index (PyPI) has almost 95,490 packages (as of 12/26/2016) with 40+ daily new packages and github.com where most software packages, libraries and tools are stored has 5,776,767 repositories available with about 20,000 daily added repositories. DevOps tools i.e. Configuration management software supports automated installation with repeatable executions and better error handling compared to bash scripts but there is no industry standards for script formats and executions. Puppet, Ansible, Chef, CFEngine and Salt provide community contributed repositories to automate software installation, for example, Ansible Galaxy has 9329 roles available, Chef Supermarket has 3,135 cookbooks available although there are many duplicates. We call this is (automated) software deployment and building dynamic computing environments on virtual clusters is the main objective of this dissertation. Software defined systems (or virtual clusters) has discussed (Fox, 2013) to connect distributed big data and computing resources of Cloud and HPC, which will result in developing a suite of software deployment tools at scale. Note that this effort is mainly inspired by the previous research activities (Fox, Qiu, and Jha, 2014; Qiu et al., 2014; Fox and Chang, 2014; Fox et al., 2015; Fox et al., 2015; Fox et al., 2016).

2.2 Template

A template has been used to describe a deployment of software packages and infrastructure on virtual environments across multiple cloud providers because of its simple instructions as well as content shareability. YAML (superset of JSON) is a popular language to serialize data delivery especially as for configuration files and object persistence along with the template deployment. As an example of infrastructure deployments, Amazon Cloudformation, a template deployment service, uses YAML or JSON specification to describe a collection of Amazon virtual resources, Google Compute Cloud uses YAML with Jinja2 or Python languages to define a set of google compute resources whereas Microsoft Azure Resource Manager uses JSON to deploy Azure resources and Topology and Orchestration Specification for Cloud Applications (TOSCA) uses XML (Wettinger, Breitenbücher, and Leymann, 2014) to define a topology and a relationship. Saltstack and Ansible, a software deployment tool written in Python, use YAML to manage configuration and software installation from instructions defined in YAML text files.

Listing 1: AWS CloudFormation Example

```yaml
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType:
        Ref: InstanceType
      SecurityGroups:
      - Ref: InstanceSecurityGroup
      KeyName:
        Ref: KeyName
```

The code example in Listing 1 is a plain text to deploy a Amazon EC2 instance written in a YAML format which includes a nested data structure by indentations and key value pairs for lists (starts with dash) and dictionaries.

Listing 2: Ansible Example

```yaml
---
```

```
- hosts: opencv

  tasks:
  - name: compiler package
    apt: name=build-essential state=present update_cache=yes
  ...
```

Ansible, automation tool, uses YAML syntax with Jinja2 template to describe instructions of software installation and the code example in Listing 2 shows a code snippet of Ubuntu's APT (advanced packaging tool) installing build-essential Debian package during the OpenCV software installation.

There are several reasons to use a template for a deployment. First, installing software and building infrastructure typically demand lots of commands to run and additional configurations to setup and a template is suitable for these tasks with its data structures using key-value pairs, lists and dictionaries to contain all instructions to reproduce a same environment and to replicate an identical software installation on different locations at another time. In addition, with the advent of devops, a template deployment enables cooperation between a template developer and a template operator because a complicated set of resources and services is simplified by a single template file and delivered to an operator as an automated means of provisioning a same environment. Moreover, YAML or JSON is a simple text format for storing data which is easy to share and modify with anyone who interested in a template. There are still plenty of benefits that we can find when a template deployment is used.

Big Data applications typically require efforts on deploying all of the software prerequisites and preparing necessary compute resources. A template deployment reduced these efforts by offering an automated management on both tasks; software deployment and infrastructure provisioning, therefore we can focus on big data applications to develop.

The concept of serverless computing also applies to deploy applications with templates e.g. Listing 1. For instance, Amazon serverless compute, AWS Lambda, invokes serverless application code (also called function) based on the description of the template but uses a specific model e.g. Listing 3 for components of serverless applications. In detail, there is a main function (Handler), runtime environment (Runtime), and an actual code in a compressed format (CordUri).

Listing 3: AWS Serverless Application Model (SAM) Example

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Resources:
  MyFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: hello_python.handler
      Runtime: python2.7
      CodeUri: 's3://my-bucket/function.zip'
```

2.3 Container technologies

Container technology has brought a lightweight virtualization with a Linux kernel support to enable a portable and reproducible environment across laptops and HPC systems. Container runtime toolkit such as Docker (Merkel, 2014), rkt (rkt, 2016) and LXD (lxd, 2016) has been offered since 2014 which uses an image file to initiate a container including necessary software packages and libraries without an hypervisor which creates an isolated environment using a virtual instance but with an isolated namespace on a same host operating system using the Linux kernel features such as namespaces, cgroups, seccomp, chroot and apparmor. Recent research (Felter et al., 2015) shows that containers outperform traditional virtual machine deployments yet running containers on HPC systems is still an undeveloped area. Shifter (Jacobsen and Canon, 2015) and Singularity (Kurtzer, 2016) have introduced to support containers on HPC with a portability and MPI support along with docker images. These efforts will be beneficial to scientific applications to conduct CPU or GPU intensive computations with easy access of container images. For example, a neuroimaging pipelines, BIDS Apps (Gorgolewski et al., 2016), is applied to HPCs using Singularity with existing 20 BIDS application images and Apache Spark on HPC

Cray systems (Chaimov et al., 2016) is demonstrated by National Energy Research Scientific Computing Center (NERSC) using shifter with a performance data of big data benchmark. Both researches indicate that scientific and big data workloads are supported by container technologies on HPC systems for reproducibility and portability.

Listing 4: Dockerfile Example

```
FROM ubuntu:14.04

MAINTAINER Hyungro Lee <lee212@indiana.edu>

RUN apt-get update && apt-get install -y build-essential
...
```

Dockerfile uses a custom template to describe installation steps of building docker images in a bash like simple format. There are certain directives to indicate particular objective of the commands, for example, FROM indicates a base image to use and RUN indicates actual commands to run.

2.4 Supporting scientific applications

With a rapid increase in the size of data sets and complexity of applications, research community considers accessibility, reproducibility, resource and data sharing (Grillner et al., 2016) on HPC systems and cloud computing to process large data sets with parallel and distributed frameworks on a set of compute nodes. Container technologies are now emerged (Hale et al., 2016) to enable large scale analysis with a minimum hassle on software deployments and infrastructure provisioning yet there are not many tools available to fully engage scientific application on containers efficiently. A small number of container images for scientific applications currently exist and most of the images are dedicated for a standalone mode which is not suitable for processing large data sets with serious computational workloads. Cluster deployments using containers are needed for big data applications which provide significant speedups with parallel job executions in either embarrassingly parallel or message passing interface. A number of scientific pipelines also require the support from the containers to enable reproducibility (Boettiger, 2015; Leipzig, 2016) in different platforms because most scientific pipelines have dependency issues from multiple software even in HPC systems with containers. Listing 5 shows a BLAST tool described by the Common Workflow Language Specification (CWL; `https://github.com/common-workflow-language`) which is for running a tool on a shared platform including cloud computing and Docker.

Listing 5: Common Workflow Language Specication (CWL) Example

```
cwlVersion: v1.0
class: CommandLineTool
requirements:
- $import: envvar-global.yml
- class: InlineJavascriptRequirement
- class: ShellCommandRequirement
- class: DockerRequirement
- $import: blast-docker.yml

inputs:
  db:
    type: String
    inputBinding:
      position: 1
    doc: BLAST database name
...
outputs:
  output:
  type: File
  outputBinding:
```

```
    glob: $(inputs.out)
baseCommand: blastn
```

3 Template Deployment and Orchestration

Template deployment is a means of installing software and building infrastructure by reading a file written in a templating language such as YAML, JSON, Jinja2 or Python. The goal of a template deployment is to offer easy installation, repeatable configuration, shareability of instructions for software and infrastructure on various platforms and operating systems. A template engine or an invoke tool is to read a template and run actions defined in a template towards target machines. Actions such as installing software package and setting configurations are described in a template file using its own syntax. For example, YAML uses spaces as indentation to describe a depth of a dataset along with a dash as a list and a key-value pair with a colon as a dictionary and JSON uses a curly bracket to enclose various data types such as number, string, boolean, list, dictionary and null. In a DevOps environment, the separation between a template writing and an execution helps Continuous Integration (CI) because a software developer writes deployment instructions in a template file while a system operations professional executes the template as a cooperative effort. Ansible, SaltStack, Chef or Puppet is one of popular tools to install software using its own templating language. Common features for those tools are installing and configuring software based on definitions but with different strategies and frameworks. One observation is that the choice of implementation languages for those tools influences the use of a template language. The tools written by Python such as Ansible and SaltStack use YAML and Jinja which are friendly to a Python language with its library support whereas the tools written by Ruby such as Chef and Puppet use Embedded Ruby (ERB) templating language. In scientific community, a template has been used to describe data and processes of pipelines and workflow because a template contains detailed information of them in writing and assists sharing and connecting between different layers and tools. Parallel execution on distributed environments is also supported in many tools yet enabling computations in a scalable manner needs expertise to prepare and build the environments. We propose a template orchestration to encourage scientists in using distributed compute resources from HPC and cloud computing systems in which provisioning infrastructure is documented in a template and complicated pipelines and workflows are packaged by container technologies for reproducibility.

3.1 Template deployment for Big Data Applications

Software installations and configurations for particular domains have become hard to maintain because of an increased number of software packages and complexity of configurations between them to connect. Template deployment for installing and provisioning systems across from a single machine to large number of compute nodes is proposed to achive consistent and reliable software deployment and system provisioning.

First, we plan to implement a deployment tool with default components for big data software such as Apache Hadoop, Spark, Storm, Zookeeper, etc. therefore a software deployment can be achieved by loading existing templates instead of starting from scratch. The software deployment intends to support various linux distribution with different versions, therefore the software stacks are operational state in many environments without a failure.

Listing 6: Template Deployment for Big Data

```
stacks:
  - software A
  - software B
  - ...
```

Each item i.e. software indicates a single template file to look up deployment instructions. Dependencies indicates that related items to complete a deployment and the environment variables are shared while dependencies are deployed. If container image is available on the web, container image deployment is expected using the URI location to save compile time.

Listing 7: Sample of software template

```
instruction:
  - install package A
  - download data B
```

```
location:
  <URI>
dependency:
  - software A
  - library B
environment_variables:
  - HOME_DIR=/opt/software_a
```

Infrastructure deployment is provisioning of cloud computing which includes virtual machine images, server types, network groups, etc. in preparation of virtual resources for the software stacks. Infrastructure deployment for multiple cloud platforms includes Microsoft Azure Resource Manager Templates, Amazon CloudFormation Templates, and Google Compute Instance Templates. Each cloud provider owns individual models for their services therefore a template of the deployment is solely executable in each provider although similar infratructure is necessary for the software stacks.

Listing 8: Support for cloud providers

```
infrastructure:
  - default: aws
  - available:
    - aws
    - gce
    - azure
    - openstack
aws:
  services:
    image:
      - image A
      - image B
      - image B version 2
    server:
      - server type A
    network:
      - network interface a
      - network ip address a
```

We plan to integrate container based deployments with popular tools such as Docker therefore image based software deployment is also supported to enhance reproducibility and mobility on different environments.

Listing 9: Template Deployment with Containers

```
format:
  - default: docker
  - available:
    - docker
    - ansible
    - shell
    - rkt
```

Template has been used to document instructions for particular tasks such as software installation and configuration or infrastructure provisioning on cloud computing, however, shareability of templates is not improved which requires for better productivity and reusability. We plan to design a template hub to collect, share, search and reuse well written templates with a common language e.g. yaml or json, therefore building software stacks and provisioning infrastructure both are repeatable in any place at any time.

In addition, provenance data and process state will be reserved.

3.2 Infrastructure Provisioning on Clouds

Infrastructure provisioning has supported with templates in many cloud platforms i.e. Amazon Cloudformation, Microsoft Azure Resource Manager, OpenStack Heat and Google Compute Instance Templates. Infrastructure described in a template will be created for simple tasks running in a standalone machine or multiple tasks in clusters.

3.2.1 Simple Azure - Python Library for Template Deployment on Windows Azure

Implementation of infrastructure provisioning is provided with Azure use case. Simple Azure is a Python library for deploying Microsoft Azure Services using a Template. Your application is deployed on Microsoft Azure infrastructure by Azure Resource Manager (ARM) Templates which provides a way of building environments for your software stacks on Microsoft Azure cloud platform. Simple Azure includes 407 community templates from Azure QuickStart Templates to deploy software and infrastructure ranging from a simple linux VM deployment (i.e. 101-vm-simple-linux) to Azure Container Service cluster with a DC/OS orchestrator (i.e. 101-acs-dcos). It supports to import, export, search, modify, review and deploy these templates using the Simple Azure library and retrieve information about deployed services in resource groups. Initial scripts or automation tools can be triggered after a completion of deployements therefore your software stacks and applications are installed and configured to run your jobs or start your services. Starting a single Linux VM with SSH key from Azure QuickStart Template is described in listing 10:

Listing 10: Simple Azure

```
>>> from simpleazure import SimpleAzure
>>> saz = SimpleAzure()

# aqst is for Azure QuickStart Templates
>>> vm_sshkey_template = saz.aqst.get_template('101-vm-sshkey')

# arm is for Azure Resource Manager
>>> saz.arm.set_template(vm_sshkey_template)
>>> saz.arm.set_parameter("sshKeyData", "ssh-rsa AAAB... hrlee@quickstart")
>>> saz.arm.deploy()
```

3.3 Semantics

Advances in big data ecosystem will require to connect scattered data sources, applications and software in meaningful semantics. It is necessary to develop structured semantics as an effort of support in discovering big data tools, datasets and applications all connected because semantics is more understandable to both human and machine with a standard syntax for expressing contents in RDF (Resource Description Framework) model or JSON-LD (Linked Data using JSON) (Labrinidis and Jagadish, 2012; Bizer et al., 2012; Simmhan et al., 2013). It also provides a guideline to construct big data software stacks to community in which preparing development environments is complicated with newly introduced software and datasets. This is particularly useful given the increasing number of tools, libraries and packages for further development of big data software stacks. One example in the listing 11 shows two applications, C++ Parser for MNIST Dataset and a Python package to convert IDX file format provided by Yann LeCun's dataset, are available for MNIST database of handwritten digits on github. There are couple of tasks to implement semantics for template deployment:

1. collect big data software, applications, and datasets
2. produce JSON-LD documents
3. derive Rest API to search, list and register
4. implement a library to explore documents about big data ecosystem

Listing 11: Sample of linked data between dataset and software

```
{
    "@context": "http://schema.org/",
    "@type": "Dataset",
    "distribution": "http://yann.lecun.com/exdb/mnist/",
    "workExample": [
      {
        "@type": "SoftwareSourceCode",
        "codeRepository": "https://github.com/ht4n/CPPMNISTParser",
        "description": "C++ Parser for MNIST Dataset",
        "dateModified": "Sep 1, 2014",
        "programmingLanguage": "C++"
      },
      {
        "@type": "SoftwareSourceCode",
        "codeRepository": "https://github.com/ivanyu/idx2numpy",
        "description": "A Python package which provides tools to convert files to and from IDX format",
        "dateModified": "Sep 16, 2016",
        "programmingLanguage": "Python"
      }
    ]
  }
```

4 Container Technology

With the increased attention of Docker container software and reproducibility, the use of virtualization has been moved from the hypervisor to a linux container technology which shares kernel features but in a separated name space on a host machine with a near native performance (Felter et al., 2015). The recent researches (Benedicic et al., 2016) indicate that the HPC community takes account of container technologies to engage scientists in solving domain problems with less complication of deploying workflows or pipelines on multiple nodes as new implementations have been introduced (Kurtzer, 2016; Jacobsen and Canon, 2015; Priedhorsky and Randles, 2016). Container technology with HPC, however, is focused on supporting compute-intensive applications i.e. Message Passing Interface (MPI) although many scientific problems are evaluated with big data software and applications. Investigation on container technology with big data ecosystem is necessary to nurture the data-intensive software development on HPC with a rich set of data analysis applications.

Modern container software run with container images to create isolated user space based on pre-configured environments. Authoring container image definition is a first step to prepare custon environments via containers and to share with others. Dockerfile is a text file to create a docker container image with intstructions for package installation, command executions, and environment variable settings. Definition File of Singularity also contains similar instructions to build container images. Application Container Image (ACI) of CoreOS rkt is generated by a shell script and acbuild command line tool but building container images is similar to docker. The main objective of using these container image definitions (formats?) is to reveal user commands and settings explicitly therefore the development environment can be shared easily and conversion between other platforms is doable. The initial goal of using container technology in this dissertation is building a container-based big data ecosystem by offering a template-based deployment for container images. It would also enable a concise and descriptive way to launch complex and sophisticated scientific pipelines using existing container images or deployment scripts. Performance tests are followed to demonstrate efficiency of the deployments with big data applications on modern container technologies. We desire to measure overhead introduced by container software i.e. shifter, singularity on HPC environments with comparison of CPU, memory, filesystem, and network usages.

Template based deployment is adopted in container technologies, for example, Singularity uses a custom syntax, SpecFile to describe the creation of a container image with directives which are similar to Dockerfile. Listing 12 shows an example of Caffe Deep Learning Framework Singularity image creation.

Listing 12: Singularity Example

```
DistType "debian"
MirrorURL "http://us.archive.ubuntu.com/ubuntu/"
OSVersion "trusty"

Setup
Bootstrap

...(suppressed)...
RunCmd git clone -b master --depth 1 https://github.com/BVLC/caffe.git
RunCmd sh -c "cd caffe && mkdir build && cd build && cmake -DCPU_ONLY=1 .. "
RunCmd sh -c "cd caffe/build && make -j1"

RunCmd ln -s /caffe /opt/caffe
RunScript python
```

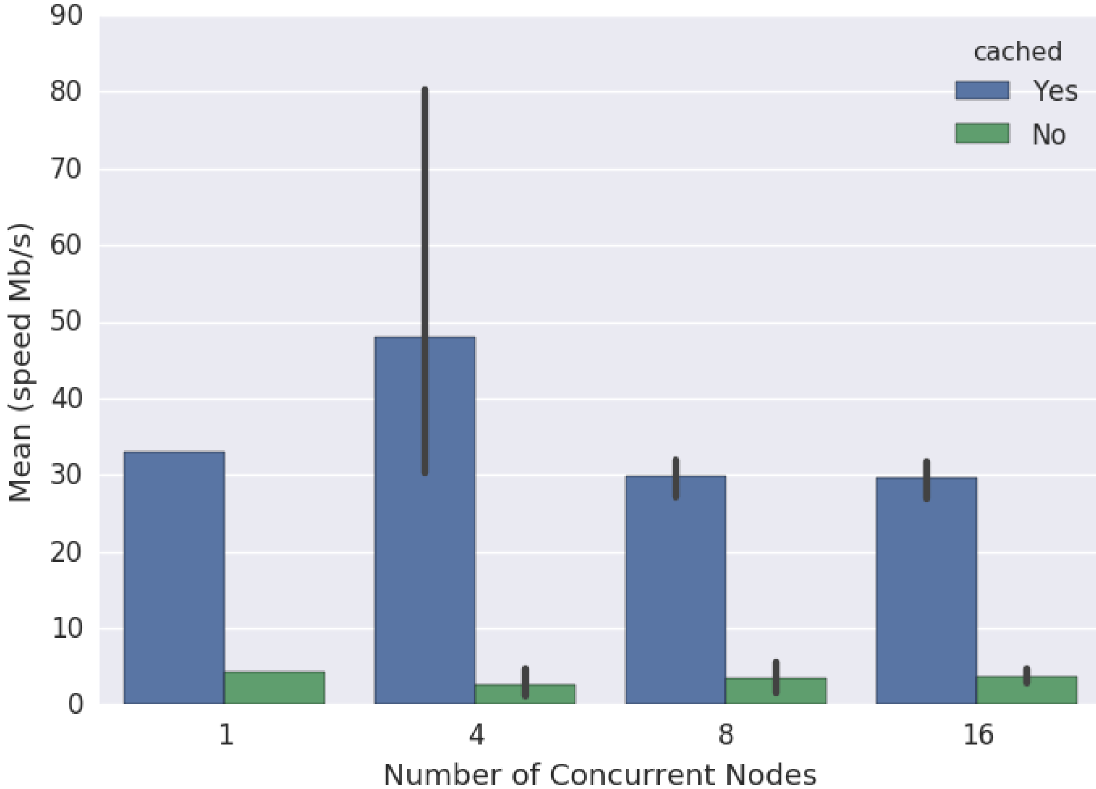
4.1 Common Installed Packages

One of the benefits of using template deployment is that a list of installed software packages is included in the instruction, therefore common packages are revealed for particular collections. Table 1 is an example of debian packages described in Dockerfiles related to NIST collection and dpkg, debian package command, has been used to collect package information.

Name	Description	Dependencies	Size (Kb)	Priority
build-essential	Informational list of build-essential packages	dpkg-dev, libc6-dev, gcc, g++, make	20 (14464)	optional
python-dev	header files and a static library for Python (default)	python, python2.7-dev, libpython-dev	45 (1024)	optional
autoconf	automatic configure script builder	m4, debianutils, perl	1890 (17956)	optional
software-properties-common	manage the repositories that you install software from (common)	python3-dbus, python-apt-common, python3-software-properties, gir1.2-glib-2.0, ca-certificates, python3:any, python3-gi, python3	184 (3125)	optional
python	interactive high-level object-oriented language (default version)	libpython-stdlib, python2.7	680 (384)	standard
automake	Tool for generating GNU Standards-compliant Makefiles	autoconf, autotools-dev	1484 (2074)	optional
zlib1g-dev	compression library - development	libc6-dev, zlib1g	416 (12516)	optional
apt-utils	package management related utility programs	libgcc1, libapt-inst1.7, libstdc++6, apt, libdb5.3, libc6, libapt-pkg4.16	688 (21070)	important
g++	GNU C++ compiler	cpp, gcc, g++-5, gcc-5	16 (51922)	optional
binutils	GNU assembler, linker and binary utilities	zlib1g, libc6	12728 (10924)	optional
gcc	GNU C compiler	cpp, gcc-5	44 (22199)	optional
python-numpy	Numerical Python adds a fast array facility to the Python language	python, python2.7:any, libblas3, liblapack3, libc6	8667 (17873)	optional
nodejs	evented I/O for V8 javascript	libssl1.0.0, libc6, libstdc++6, zlib1g, libv8-3.14.5, libc-ares2	3043 (20625)	extra
pkg-config	manage compile and link flags for libraries	libglib2.0-0, dpkg-dev, libc6	140 (17322)	optional
python-imaging	Python Imaging Library compatibility layer	python-pil, python:any	45 (1248)	optional

Table 1: Top 15 Debian-based Packages used in Dockerfiles for the NIST collection on Github, size with parenthesis indicates total size including dependency packages

Figure 1: Accelerated Common Package Installation using Software Package Proxy



Type	Hits	Misses	Total
Requests	104993 (95.26%)	5220 (4.74%)	110213
Data	12627.32 MiB (99.78%)	27.95 MiB(0.22%)	12655.27 MiB

Table 2: Cache Efficiency for Software Package Installation measured by apt-cacher-ng

4.2 Evaluation

As a part of dissertation, performance tests of container technologies with big data applications from NIST Collection. There are six applications in the collection: Fingerprint Matching, Human and Face Detection, Twitter Live Analysis, Data Warehousing, Healthcare Information, and Geospatial information. Performance data on CPU, memory, storage and netowrk will be measured on HPC and cloud computing with container software i.e. docker, rkt, singularity and shifter.

Preloading common packages shows possible optimization for the template deployment according to the figure 1. With a considerable reduce on network traffic for downloading packages, 10x speedup is approximately observed over multiple access to Debian software package mirror sites. Statistics for the cache reuse (Table 2) indicates that the most benefit of the speedup is gained from the cached packages. In addition, standard deviation for download speed is higher in using remote mirrors than cached proxy server in which network consistency and reliability are ensured with low standard deviation for download speed.

5 Bioinformatics

Bioinformatics pipeline frameworks have used templating languages to describe workflow processes with input and output parameters, for example, Common Workflow Language Specification (CWL)(cwl, 2016) uses YAML syntax and JSON format parameter files to define workflow logics with its required tools and parameters from command line interface (CLI). There are several implementations supporting the CWL such as Rabix(Kaushik et al., 2016), Arvados(arv, 2016), Galaxy, Taverna and Kronos(Taghiyar et al., 2016) to use templating languages in their workflow engines with ease accomodation of tool dependencies. In our case, we have a plan to use templating language to enable parallel processing on the cloud and HPC per independent component of workflows with expectation of better performance on computation and higher resource utilization on shared resource pool.

A few efforts have been made (Lee et al., 2012; Chae et al., 2013; Lee et al., 2016) to apply bioinformatics systems to cloud computing and HPC. Template deployment for bioinformatics frameworks would be developed with these work to extend existing tools with new technologies.

6 Case Studies: NIST Big Data Projects

NIST Big Data Public Working Group (NBD-PWG) (Technology. et al., 2015; Fox and Chang, 2014) reported 51 use cases across nine application domains including Government Operation, commercial, Defense, Healthcare and Life Sciences, Deep Learning and Social Media, The Echosystem for Research, Astronomy and Physics, Earth, Envionmental and Polar Science and Energy to understand Big Data requirements and advance the development of big data framework. We ought to keep up the same effort to support scientific community in regard to analyzing data with modern technologies and the part of this dissertation is gathering more use cases and requirements by reviewing publicly available big data applications.

6.1 Fingerprint Recognition

Fingerprint matching software (Flanagan, 2010; Flanagan, 2014) has been developed by National Institute of Standards and Technology (NIST) with special databases to identify patterns of fingerprint. NIST Biometric Image Software (NBIS) includes MINDTCT, a fingerprint minutiae detector and BOZORTH3, a minutiae based fingerprint matching program to process biometric analysis. MINDTCT program extracts the features of fingerprint such as ridge ending, bifurcation, and short ridge from the FBI's Wavelet Scalar Quantization (WSQ) images and BOZORTH3 runs fingerprint matching algorithm with the images generated by MINDTCT as part of fingerprint identification processing (Wegstein, 1982). In this use case, Apache Spark runs fingerprint matching on the Hadoop cluster with NIST Fingerprint Special Database 4 (Watson and Wilson, 1992) and stores results in HBase with the support of NoSQL database, Apache Drill. Addtional dataset from FVC2004 can be used as well with 1440 fingerprint impressions (Maio et al., 2004). Individual software represents a stack or a role in the context in which a set of tasks to complete a software deployment is included. Suggested software stacks (roles) for Fingerprint matching are:

- Apache Hadoop
- Apache Spark
- Apache HBase
- Apache Drill
- Scala

6.2 Human and Face Detection with OpenCV

Human and face detection have been studied during the last several years and models for them have improved along with Histograms of Oriented Gradients (HOG) for Human Detection (Dalal and Triggs, 2005a). OpenCV is a Computer Vision library including the SVM classifier and the HOG object detector for pedestrian detection and INRIA Person Dataset (Dalal and Triggs, 2005b) is one of popular samples for both training and testing purposes. In this use case, Apache Spark on Mesos clusters are deployed to train and apply detection models from OpenCV using Python API. Individual software represents a stack or a role in this context in which a set of tasks to complete a software deployment is included. Suggested software stacks (Roles) for human and face detection with OpenCV are:

- Apache Mesos
- Apache Spark
- OpenCV
- Zookeeper
- INRIA Person Dataset
- Python Analytics with HOG and Haar Cascades

6.3 Twitter Live Analysis

Social messages generated by Twitter have been used with various applications such as opinion mining, sentiment analysis (Pak and Paroubek, 2010), stock market prediction (Bollen, Mao, and Zeng, 2011), and public opinion polling (Cody et al., 2016) with the support of natual language toolkits e.g. nltk (Bird, 2006), coreNLP (Manning et al., 2014) and deep learning systems (Kim, 2014). Services for streaming data processing are important in this category. Apache Storm is widely used with the example of twitter sentiment analysis, and Twitter Heron, Google Millwheel, LindkedIn Samza, and Facebook Puma, Swift, and Stylus are available as well (Chen et al., 2016). Suggested software stacks (roles) for Twitter Live Analysis are:

- Apache Hadoop
- Twitter Heron
- Apache Storm
- Apache Flume
- Natural Language Toolkit (NLTK)

6.4 Big Data Analytics for Healthcare Data and Health Informatics

Several attempts have been made to apply Big Data framework and analytics in health care with various use cases. Medical image processing, signal analytics and genome wide analysis are addressed to provide efficient diagnostic tools and reduce healthcare costs (Belle et al., 2015) with big data software such as Hadoop, GPUs, and MongoDB. Open source big data ecosystem in healthcare is introduced (Raghupathi and Raghupathi, 2014) with examples and challenges to satisfy big data characteristics; volume, velocity, and variety (Zikopoulos, Eaton, and others, 2011). Cloud computing framework in healthcare for security is also discussed with concerns about privacy (Stantchev, Colomo-Palacios, and Niedermayer, 2014). Suggested software stacks (roles) for Big Data Analytics for Healthcare Data and Health Informatics are:

- Apache Hadoop
- Apache Spark
- Apache Mahout
- Apache Lucene/Solr
- MLlib

6.5 Spatial Big Data, Spatial Statistics and Geographic Information Systems

The broad use of geographic information system (GIS) has been increased over commercial and scientific communities with the support of computing resources and data storages. For example, Hadoop-GIS (Aji et al., 2013), a high performance spatial data warehousing system with Apache Hive and Hadoop, offers spatial query processing in parallel with MapReduce, and HadoopViz (Eldawy, Mokbel, and Jonathan, 2016), a MapReduce framework for visualizing big spatial data, supports various visualization types of data from satellite data to countries borders. Suggested software stacks (roles) for Spatial Big Data, Spatial Statistics and Geographic Information Systems are:

- Apache Hadoop
- Apache Spark
- GIS-tools
- Apache Mahout
- MLlib

6.6 Data Warehousing and Data Mining

Researches in data warehousing, data mining and OLAP have investigated current challenges and future directions over big data software and applications (Cuzzocrea, Bellatreche, and Song, 2013) due to the rapid increase of data size and complexity of data models. Apache Hive, a warehousing solution over a hadoop (Thusoo et al., 2009), has introduced to deal with large volume of data processing with the other research studies (Chen, 2010; He et al., 2011) and NoSQL platforms (Chevalier et al., 2015) have discussed with data warehouse ETL pipeline (Goodhope et al., 2012). Suggested software stacks (roles) for Data Warehousing and Data Mining are:

- Apache Hadoop
- Apache Spark
- MongoDB
- Hive
- Pig
- Apache Mahout
- Apache Lucene/Solr
- MLlib

6.6.1 Big Data Statistics from GitHub Repositories

Github.com has been used to provide version control and manage source code development along with diverse collaborators across countries. The popularity of github as a collaboration tool has been significantly increased and 4,995,050 repositories exist as of 12/27/2016 with 20-30 thousands daily added repositories. Therefore we report a repository statistics to understand software development related to big data applications and tools and to create a list of most common tools regarding to big data deployments. A development language distribution, most common libraries and packages, observations over a certain period and detection on recently added projects and tools are main part of the queries using github search API. We defined a set of keywords for projects to retrieve related github repositories, for example, fingerprint matching, fingerprint recognition, fingerprint verification, and biometric fingerprint are used to search github projects related to fingerprint recognition. Python and Java are most common languages among the six NIST projects (Table 3), although matlab is dominant in the fingerprint project. We also noticed that scientific python packages are commonly used to enable numerical computation, data analysis and visualization for these big data applications (Figure 2), whereas there are dependent packages for each project (Table 4). Tweepy, twitter API, is used in the twitter live analysis cases with NLTK, the natural language processing toolkit to complete sentiment analysis with tweets. Similarly, GIS projects use particular libraries for spatial analysis such as geopy and shapely. We observe that deep learning python packages e.g. caffe have recently added to github repositories. Statistics (tables 5 to 10) show that popular github repository examples related to the six nist projects started in 2016. Each github project has different language preferences with various libraries and packages therefore recent activities can be observed, for example, deep learning software such as Keras, Theano, mxnet and Caffe is adopted among multiple projects.

6.7 Datasets

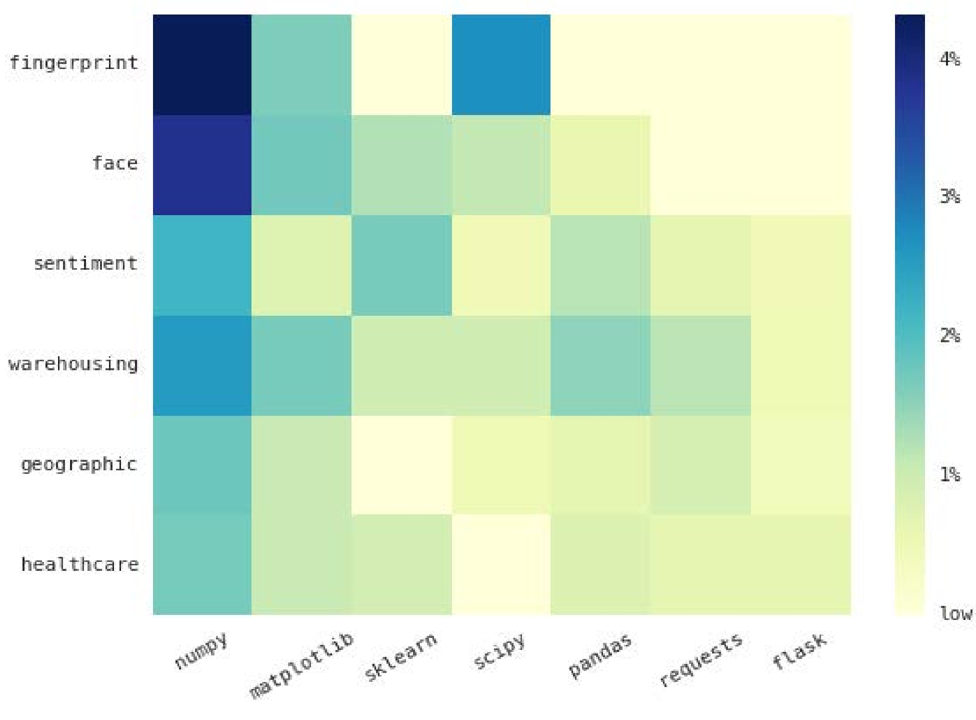
Finding relevant datasets for particular applications is another challenge for the big data ecosystem because of its difficulty of collecting data from different sources (Kim, Trimi, and Chung, 2014), complexity and diversity (Hashem et al., 2015). Community contributed lists of public datasets (Cohen and Lo, 2014) provide structured information with a specific location to access data and a category to describe itself. We intend to generate linked json data for datasets and applications in big data ecosystem based on these lists because it connects scattered data and software in an organized way. Table 11 shows the data source from different sectors, academia(.edu or .ac.), government(.gov), organization(.org), industry(.com or .net), and international(country suffix), among the seven categories of the lists. Entire categories are available online: `https://github.com/lee212/bd_datasets`. Listing 11 also shows a example of the linked data between MNIST dataset and two software available on github.com.

Topic	C++	Python	Java	Matlab	JS	C#	C	R	Ruby	Scala	Count*
Fingerprint (6.1)	15%	11%	13%	20%	3%	16%	8%	0%	1%	5%	43
Face (6.2)	26%	21%	12%	9%	7%	5%	2%	2%	1%	.02%	538
Twitter (6.3)	2%	35%	15%	.6%	9%	2%	1%	10%	3%	1%	1429
Warehousing (6.6)	3%	27%	18%	2%	10%	3%	1%	10%	4%	1%	3435
Geographic (6.5)	5%	15%	27%	4%	15%	3%	5%	7%	3%	16%	6487
Healthcare (6.4)	2%	13%	19%	2%	14%	5%	1%	10%	6%	2%	132

Table 3: Language Distribution of Topics related to those in the NIST collection on Github

* Count: average number of github.com repositories.

Figure 2: Scientific Python Packages used in NIST Projects (collected from Github)



Python Package	Description	Fingerprint	Face	Twitter	Warehousing	Geographic	Healthcare
cv2	OpenCV	✓	✓				
skimage	Image Processing		✓				
PIL	Python Imaging Library		✓				
caffe	Deep Learning		✓				
nltk	Natural Language Toolkit			✓			
tweepy	Twitter for Python			✓			
BeautifulSoup	Screen-scraping library			✓	✓		
gensim	Topic Modelling			✓	✓		
geopy	Geocoding library					✓	
shapely	Geometric Analysis					✓	
django	Web framework				✓		✓

Table 4: Additional Python packages found in NIST Collection

Title	Description	Language	Start Date	Popularity	Dependency
OpenFace	an open source facial behavior analysis toolkit	c++	March, 2016	725 (305)	OpenCV, dlib, boost, TBB
Picasso face detection transformation	An Android image transformation library providing cropping above Face Detection (Face Centering) for Picasso	Java	July, 2016	528(56)	Square Picasso
MTCNN face detection alignment	Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Neural Networks	Matlab	September, 2016	226(162)	Caffe, Pdollar toolbox
facematch	Facebook Face Recognition wrapper	JavaScript	January, 2016	132 (41)	fbgraph, request, body-parser, express
mxnet mtcnn face detection	MTCNN face detection	Python	October, 2016	99 (47)	OpenCV, mxnet

Table 5: Example Projects Recently Created Regarding to Face Detection

Title	Description	Language	Start Date	Popularity	Dependency
CNN fingerprint	fingerprint verification using convolution neural networks	Python	December, 2016	0 (1)	Keras, Theano
fingerprint recognizer	Web app for human fingerprint recognition with math on Node.JS 7	JavaScript	December, 2016	0 (1)	jimp
neurodactyl	C++ software tool for fingerprint recognition based on neural networks	C++	November, 2016	0 (0)	OpenCV, Bozorth3, FANN

Table 6: Example Projects Recently Created Regarding to Fingerprint Matching

Title	Description	Language	Start Date	Popularity	Dependency
tidytext	Text mining using dplyr, ggplot2, and other tidy tools	R	March, 2016,	310 (42)	dplyr, ggplot2, tidyr, broom
Bayesian sentiment analysis	Pragmatic & Practical Bayesian Sentiment Classifier	Kotlin	August, 2016	213 (19)	Apache Lucene
Hotel review analysis	Sentiment analysis and aspect classification for hotel reviews using machine learning models with MonkeyLearn	Python	April, 2016	154 (34)	Scrapy, ElasticSearch, Kibana, nltk, pandas
Sentiments	Sentiments is an iOS app written in Swift that analyzes text for positive or negative sentiment	Swift	February, 2016	146 (8)	Alamofire, SwiftyJSON, HPE Haven OnDemand
Sentiment Analysis Twitter	We use different feature sets and machine learning classifiers to determine the best combination for sentiment analysis of twitter	Python	October, 2016	139 (42)	twitter, mdp

Table 7: Example Projects Recently Created Regarding to Twitter Analysis

Title	Description	Language	Start Date	Popularity	Dependency
gpq	A collection of tools for mining government data	Jupyter Notebook	June, 2016	128 (10)	Google BigQuery
reair	a collection of easy-to-use tools for replicating tables and partitions between Hive data warehouses	Java	March, 2016	91 (42)	Hadoop, Hive
SpawnTracker	Probably the most efficient large area long duration tracker for pokemon go data mining	Protocol Buffer, Python	July, 2016	34 (9)	s2sphere, GeoJSON, protobuf, pgoapi
idbr	An R interface to the US Census Bureau International Data Base API	R	January, 2016	23 (12)	dplyr, ggplot2, ggthemes
get-tiger	Make workflow for downloading Census geodata and joining it to survey data	Makefile	February, 2016	20 (0)	GDAL

Table 8: Example Projects Recently Created Regarding to Data Warehousing

Title	Description	Language	Start Date	Popularity	Dependency
Pokemon Go Move	Pokemon GO GPS Emulator with Built-In Pokemon/Pokestop/Gym Map	Python	July, 2016	401 (74)	geopy, s2sphere
d3-geo	Geographic projections, spherical shapes and spherical trigonometry	Javascript	March, 2016	112 (35)	d3-array
Geospatial messenger	Geospatial messenger application written with Spring Boot + Kotlin + PostgreSQL	Kotlin	March, 2016	105 (22)	Spring Boot, PostgreSQL
DotSpatial	Geographic information system library written for .NET	C#	April, 2016	94 (52)	
geo.lua	A helper library for Redis geospatial indices	Lua	February, 2016	74 (7)	

Table 9: Example Projects Recently Created Regarding to Geographic Information Systems

Title	Description	Language	Start Date	Popularity	Dependency
Temperate	a healthcare application that aims to make healthcare more accessible to everyone, everywhere	JavaScript, PHP	January, 2016	91 (7)	MySQL
Computational Healthcare	Analyze large healthcare datasets & build machine learning models using TensorFlow	Python	December, 2016	41 (15)	
healthcareai	R tools for healthcare machine learning	R	June, 2016	24 (10)	SQL Server
datasus	An Interface for the Brazilian Public Healthcare Data Repository (DATASUS) for the R Language	R	June, 2016	11 (6)	
RETAIN	Interpretable Predictive Model in Healthcare using Reverse Time Attention Mechanism	Python	August, 2016	6 (2)	Theano, CUDA

Table 10: Example Projects Recently Created Regarding to Healthcare Data

Category	Academia	Government	Organization	Industry	International	Total
GIS	1	3	5	9	5	23
Healthcare	0	6	3	1	1	11
Image Processing	11	0	4	2	5	18
Natural Language	7	0	8	7	6	26
Social Networks	8	0	7	5	5	24
Climate/Weather	2	6	3	2	4	16
Energy	2	2	5	1	5	15

Table 11: Public Dataset sectors of acamedia, government, organization, industry and international

Task	Completion
Deployment	
Proposal	February, 2017
Serverless Computing	
- IEEE Cloud	February 2017
Big Data Deployment by DevOps and SDS	
- IEEE Big Data Congress	March 2017
Software Defined Systems	April 2017
Case Study: NIST Project	December 2016
Scientific Applications	
Bioinformatics Integration	May 2017
Case Study	May 2017
Dissertation	
Writing	June 2017
Defense	July 2017

Table 12: Timeline for completion of this thesis

7 Research Plan

Table 12 provides a summary of the remaining tasks and their expected completion date. Some comments and risk assessments follow.

IEEE Cloud, IEEE Big Data Congress The target venues are addressed

8 Dissertation Chapters

1. Introduction
2. Background
3. Template-based Infrastructure Provisioning
4. DevOps Software Deployment
5. Event-driven Computing with CRIU
6. NIST Use Cases
7. Curated Package Recommender for Dynamic Computing Environment
8. Integration with Bioinformatics
9. Software Defined Systems with Serverless Paradigm
10. Conclusions

9 Summary

Software defined systems presents manageable, dynamic and flexible computing resources with the serverless paradigm to ensure simplicity of data processing but guaranteed performance of computation through infrastructure provisioning. The combination of DevOps tools and Templates removes a barrier of using systems from complicated software stacks and the shareability and elasticity are inherited to the software defined systems on both HPC and Clouds.

10 Related Work

10.1 Template deployment

Several infrastructure provisioning tools have emerged to offer transparent and simple management of cloud computing resources over the last few years. Templates which are structured documents in a YAML or JSON format define infrastructure with required resources to build and ensure identical systems to create over time. A collection of Amazon cloud services are provisioned through Cloudformation (clo, 2010) templates which is an Amazon infrastructure deployment service. OpenStack Heat (osh, 2012) was started with similar template models to Amazon but has extended with other OpenStack services e.g. Telemetry, monitoring and autoscaling service to build multiple resources aa a single unit. The Topology and Orchestration Specification for Cloud Applications (TOSCA) (Wettinger, Breitenbücher, and Leymann, 2014; Binz et al., 2014) proposes standardization over different cloud platforms with XML-based language and several studies have been made with TOSCA (Kopp et al., 2013; Breiter et al., 2014; Qasha, Cala, and Watson, 2015). These tools have been addressed with issues in a few studies (Yamato et al., 2014; Fox et al., 2015) and one of identified issues is that individual specification of supported resources, functions, type names, and parameters prevents building and sharing infrastructure blueprints across cloud platforms.

10.2 DevOps Tools

In the DevOps phase, configuration management tools automates software deployment to provide fast delivery process between development and operations (Ebert et al., 2016). Instructions to manage systems and deploy software are written in scripts although different formats i.e. YAML, JSON, and Ruby DSL and various terminologies i.e. recipes, manifests, and playbooks are used. There are notable tools available to achieve automated software deployment. Puppet and Chef are identified configuration management tools written in Ruby and these tools manage software on target machines regarding to installation, execution in a different state e.g. running, stopping or restarting, and configuration through the client/server mode (also called master/agent). Ansible is also recognized as a configuration management tool but more focusing on software deployment using SSH and no necessity of agents on target machines. With the experience from class projects and NIST use cases, a few challenging tasks are identified in DevOps tools, a) offering standard specification of scripts to ease script development with different tools, and b) integrating container technologies towards microservices.

10.3 Container technology

While existing container software, e.g. docker, rkt, lxd, offers various features with outstanding performance there are number of new tools recently developed with the support on HPC. Shifter from NERSC on Cray XC30 with GPU (Benedicic et al., 2016) has introduced and singularity from LBNL (Kurtzer, 2016) as well. These new implementations are typically for heavy workloads which requires checkpoint/restart for long running applications and easy deployment of required software stacks in a user space.

References

2010. Amazon CloudFormation. https://aws.amazon.com/cloudformation/. [Online; accessed 17-February-2017].

2012. OpenStack Heat. https://wiki.openstack.org/wiki/Heat. [Online; accessed 17-February-2017].

2016. Common Workflow Language Specification. https://github.com/common-workflow-language/common-workflow-language. [Online; accessed 09-November-2016].

2016. Coreos/rkt: a container engine for linux designed to be composable, secure, and built on standard. https://github.com/coreos/rkt. [Online; accessed 09-November-2016].

2016. Distributed computing platform for data analysis on massive data sets. https://arvados.org. [Online; accessed 09-November-2016].

2016. Ubuntu lxd: a pure-container hypervisor. https://github.com/lxc/lxd. [Online; accessed 09-November-2016].

Aji, Ablimit, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, and Joel Saltz. 2013. Hadoop gis: a high performance spatial data warehousing system over mapreduce. *Proceedings of the VLDB Endowment*, 6(11):1009–1020.

Belle, Ashwin, Raghuram Thiagarajan, SM Soroushmehr, Fatemeh Navidi, Daniel A Beard, and Kayvan Najarian. 2015. Big data analytics in healthcare. *BioMed research international*, 2015.

Benedicic, Lucas, Miguel Gila, Sadaf Alam, and Thomas C Schulthess. 2016. Opportunities for container environments on cray xc30 with gpu devices.

Binz, Tobias, Uwe Breitenbücher, Oliver Kopp, and Frank Leymann. 2014. Tosca: portable automated deployment and management of cloud applications. In *Advanced Web Services*. Springer, pages 527–549.

Bird, Steven. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Bizer, Christian, Peter Boncz, Michael L Brodie, and Orri Erling. 2012. The meaningful use of big data: four perspectives–four challenges. *ACM SIGMOD Record*, 40(4):56–60.

Boettiger, Carl. 2015. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79.

Bollen, Johan, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Breiter, Gerd, Michael Behrendt, M Gupta, Simon Daniel Moser, R Schulze, I Sippli, and Thomas Spatzier. 2014. Software defined environments based on tosca in ibm cloud implementations. *IBM Journal of Research and Development*, 58(2/3):9–1.

Chae, Heejoon, Inuk Jung, Hyungro Lee, Suresh Marru, Seong-Whan Lee, and Sun Kim. 2013. Bio and health informatics meets cloud: Biovlab as an example. *Health Information Science and Systems*, 1(1):6.

Chaimov, Nicholas, Allen Malony, Shane Canon, Costin Iancu, Khaled Z Ibrahim, and Jay Srinivasan. 2016. Scaling spark on hpc systems. In *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, pages 97–110. ACM.

Chen, Guoqiang Jerry, Janet L Wiener, Shridhar Iyer, Anshul Jaiswal, Ran Lei, Nikhil Simha, Wei Wang, Kevin Wilfong, Tim Williamson, and Serhat Yilmaz. 2016. Realtime data processing at facebook. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1087–1098. ACM.

Chen, Songting. 2010. Cheetah: a high performance, custom data warehouse on top of mapreduce. *Proceedings of the VLDB Endowment*, 3(1-2):1459–1468.

Chevalier, Max, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2015. Implementing multidimensional data warehouses into nosql. In *17th International Conference on Enterprise Information Systems (ICEIS15), Spain*.

Cody, Emily M, Andrew J Reagan, Peter Sheridan Dodds, and Christopher M Danforth. 2016. Public opinion polling with twitter. *arXiv preprint arXiv:1608.02024*.

Cohen, Joseph Paul and Henry Z Lo. 2014. Academic torrents: A community-maintained distributed repository. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, page 2. ACM.

Cuzzocrea, Alfredo, Ladjel Bellatreche, and Il-Yeol Song. 2013. Data warehousing and olap over big data: current challenges and future research directions. In *Proceedings of the sixteenth international workshop on Data warehousing and OLAP*, pages 67–70. ACM.

Dalal, Navneet and Bill Triggs. 2005a. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE.

Dalal, Navneet and Bill Triggs. 2005b. Inria person dataset.

Ebert, Christof, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. 2016. Devops. *IEEE Software*, 33(3):94–100.

Eldawy, Ahmed, M Mokbel, and Christopher Jonathan. 2016. Hadoopviz: A mapreduce framework for extensible visualization of big spatial data. In *IEEE Intl. Conf. on Data Engineering (ICDE)*.

Felter, Wes, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. 2015. An updated performance comparison of virtual machines and linux containers. In *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On*, pages 171–172. IEEE.

Flanagan, Patricia. 2010. Nist biometric image software (nbis).

Flanagan, Patricia. 2014. Fingerprint minutiae viewer (fpmv).

Fox, Geoffrey. 2013. Distributed data and software defined systems.

Fox, Geoffrey and Wo Chang. 2014. Big data use cases and requirements. In *1st Big Data Interoperability Framework Workshop: Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data*, pages 18–21. Citeseer.

Fox, Geoffrey, Judy Qiu, and Shantenu Jha. 2014. High performance high functionality big data software stack.

Fox, Geoffrey, Judy Qiu, Shantenu Jha, Saliya Ekanayake, and Supun Kamburugamuve. 2015. Big data, simulations and hpc convergence. In *Workshop on Big Data Benchmarks*, pages 3–17. Springer.

Fox, Geoffrey, Judy Qiu, Shantenu Jha, Saliya Ekanayake, and Supun Kamburugamuve. 2016. White paper: Big data, simulations and hpc convergence. In *BDEC Frankfurt workshop. June*, volume 16.

Fox, Geoffrey C, Judy Qiu, Supun Kamburugamuve, Shantenu Jha, and Andre Luckow. 2015. Hpc-abds high performance computing enhanced apache big data stack. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 1057–1066. IEEE.

Goodhope, Ken, Joel Koshy, Jay Kreps, Neha Narkhede, Richard Park, Jun Rao, and Victor Yang Ye. 2012. Building linkedin's real-time activity data pipeline. *IEEE Data Eng. Bull.*, 35(2):33–45.

Gorgolewski, Krzysztof J, Fidel Alfaro-Almagro, Tibor Auer, Pierre Bellec, Mihai Capota, M Mallar Chakravarty, Nathan W Churchill, R Cameron Craddock, Gabriel A Devenyi, Anders Eklund, et al. 2016. Bids apps: Improving ease of use, accessibility and reproducibility of neuroimaging data analysis methods. *bioRxiv*, page 079145.

Grillner, Sten, Nancy Ip, Christof Koch, Walter Koroshetz, Hideyuki Okano, Miri Polachek, Mu-ming Poo, and Terrence J Sejnowski. 2016. Worldwide initiatives to advance brain research. *Nature neuroscience*, 19(9):1118–1122.

Hale, Jack S, Lizao Li, Chris N Richardson, and Garth N Wells. 2016. Containers for portable, productive and performant scientific computing. *arXiv preprint arXiv:1608.07573*.

Hashem, Ibrahim Abaker Targio, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. 2015. The rise of big data on cloud computing: Review and open research issues. *Information Systems*, 47:98–115.

He, Yongqiang, Rubao Lee, Yin Huai, Zheng Shao, Namit Jain, Xiaodong Zhang, and Zhiwei Xu. 2011. Rcfile: A fast and space-efficient data placement structure in mapreduce-based warehouse systems. In *2011 IEEE 27th International Conference on Data Engineering*, pages 1199–1208. IEEE.

Jacobsen, Douglas M and Richard Shane Canon. 2015. Contain this, unleashing docker for hpc. *Proceedings of the Cray User Group*.

Kaushik, Gaurav, Sinisa Ivkovic, Janko Simonovic, Nebojsa Tijanic, Brandi Davis-Dusenbery, and Deniz Kural. 2016. Graph theory approaches for optimizing biomedical data analysis using reproducible workflows. *bioRxiv*, page 074708.

Kim, Gang-Hoon, Silvana Trimi, and Ji-Hyong Chung. 2014. Big-data applications in the government sector. *Communications of the ACM*, 57(3):78–85.

Kim, Yoon. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kopp, Oliver, Tobias Binz, Uwe Breitenbücher, and Frank Leymann. 2013. Winery–a modeling tool for tosca-based cloud applications. In *International Conference on Service-Oriented Computing*, pages 700–704. Springer.

Kurtzer, Gregory M. 2016. Singularity 2.1.2 - Linux application and environment containers for science, August.

Labrinidis, Alexandros and Hosagrahar V Jagadish. 2012. Challenges and opportunities with big data. *Proceedings of the VLDB Endowment*, 5(12):2032–2033.

Lee, Hyungro, Minsu Lee, Wazim Mohammed Ismail, Mina Rho, Geoffrey Fox, Sangyoon Oh, and Haixu Tang. 2016. Mgescan: a galaxy based system for identifying retrotransposons in genomes. *Bioinformatics*, page btw157.

Lee, Hyungro, Youngik Yang, Heejoon Chae, Seungyoon Nam, Donghoon Choi, Patanachai Tangchaisin, Chathura Herath, Suresh Marru, Kenneth P Nephew, and Sun Kim. 2012. Biovlab-mmia: a cloud environment for microrna and mrna integrated analysis (mmia) on amazon ec2. *IEEE transactions on nanobioscience*, 11(3):266–272.

Leipzig, Jeremy. 2016. A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics*, page bbw020.

Maio, Dario, Davide Maltoni, Raffaele Cappelli, Jim L Wayman, and Anil K Jain. 2004. Fvc2004: third fingerprint verification competition. In *Biometric Authentication*. Springer, pages 1–7.

Manning, Christopher D, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Merkel, Dirk. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2.

Pak, Alexander and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326.

Priedhorsky, Reid and Tim Randles. 2016. Charliecloud: Unprivileged containers for user-defined software stacks in hpc. Technical report, Los Alamos National Laboratory (LANL).

Qasha, Rawaa, Jacek Cala, and Paul Watson. 2015. Towards automated workflow deployment in the cloud using tosca. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 1037–1040. IEEE.

Qiu, Judy, Shantenu Jha, Andre Luckow, and Geoffrey C Fox. 2014. Towards hpc-abds: an initial high-performance big data stack. *Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data*, pages 18–21.

Raghupathi, Wullianallur and Viju Raghupathi. 2014. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2(1):1.

Simmhan, Yogesh, Saima Aman, Alok Kumbhare, Rongyang Liu, Sam Stevens, Qunzhi Zhou, and Viktor Prasanna. 2013. Cloud-based software platform for big data analytics in smart grids. *Computing in Science & Engineering*, 15(4):38–47.

Stantchev, Vladimir, Ricardo Colomo-Palacios, and Michael Niedermayer. 2014. Cloud computing based systems for healthcare. *The Scientific World Journal*, 2014.

Taghiyar, M Jafar, Jamie Rosner, Diljot Grewal, Bruno Grande, Radhouane Aniba, Jasleen Grewal, Paul C Boutros, Ryan D Morin, Ali Bashashati, and Sohrab P Shah. 2016. Kronos: a workflow assembler for genome analytics and informatics. *bioRxiv*, page 040352.

Technology., National Institute of Standards, , Information Technology Laboratory., NIST Big Data Public Working Group (NBD-PWG), National Institute of Standards (U.S.), and Technology. 2015. Nist big data interoperability framework : volume 3, use cases and general requirements.

Thusoo, Ashish, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. 2009. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629.

Watson, CI and CL Wilson. 1992. Nist special database 4. *Fingerprint Database, National Institute of Standards and Technology*, 17.

Wegstein, Joseph H. 1982. *An automated fingerprint identification system.* US Department of Commerce, National Bureau of Standards.

Wettinger, Johannes, Uwe Breitenbücher, and Frank Leymann. 2014. Standards-based devops automation and integration using tosca. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 59–68. IEEE Computer Society.

Wettinger, Johannes, Uwe Breitenbücher, and Frank Leymann. 2015. Dyn tail-dynamically tailored deployment engines for cloud applications. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 421–428. IEEE.

Yamato, Yoji, Masahito Muroi, Kentaro Tanaka, and Mitsutomo Uchimura. 2014. Development of template management technology for easy deployment of virtual resources on openstack. *Journal of Cloud Computing*, 3(1):1.

Zikopoulos, Paul, Chris Eaton, et al. 2011. *Understanding big data: Analytics for enterprise class hadoop and streaming data.* McGraw-Hill Osborne Media.