The concept of “Software Defined Systems” requires a convenient and hopefully “universal” way to specify the system. This requires that multiple compute, storage and network components be specified and linked together. Currently there are emerging approaches to this but no agreed standards. As often, the current situation sees a mix of standards and libraries that programmatically implement functionalities that hide underlying differences from the user. Standards come from the community like OCCI (cloud compute interface from OGF – the Open Grid Forum) and the Amazon EC2 de facto standard for cloud compute. Another major standard at this level is CDMI (Cloud Data Management Interface) from SNIA (Storage Networking Industry Association). Apache JClouds addresses compute and blob storage programmatically for Amazon, CloudStack, Docker, Google Compute Engine, HP, OpenStack, RackspaceAmazon, CloudStack, Docker, Google Compute Engine, HP, OpenStack, and Rackspace. Apache Whirr builds on JClouds and provides a common cloud service API. More comprehensively Apache Libcloud is a python library supporting universal access to compute, blob and object stores, object storage (such as S3 in Amazon), content delivery networks, load balancers and DNS for more than 30 cloud providers; as they it is “One Interface To Rule Them All”. Note this is at a higher level than the popular library libvert that provides a common API to the virtualization and container technologies.

TOSCA or Topology and Orchestration Specification for Cloud Applications, http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html, is an OASIS standard language to describe a topology of cloud based web services, their components, relationships, and the processes that manage them. It is most precisely a possible standard for software defined systems and is used in Cisco Intelligent Automation for Cloud (IAC) <http://www.cisco.com/go/iacloud> which also uses the system configuration managers Chef and Puppet. The Amazon AWS CloudFormation Template has similar goals to TOSCA but restricted to a single cloud provider.

We are now moving onto tools that support DevOps and indeed that’s not surprising. To be useful DevOps needs to specify the systems on which actions are performed and so this rapidly changing field is very relevant for “Software Defined Systems”. As well as Chef and Puppet, Ansible and Salt are respected entries in the crowded system configuration manager space with comparisons on the Internet but no “consensus winner”. A related system Juju orchestrates software services and their provisioning defined by charms across multiple clouds but is restricted to Ubuntu O/S. Foreman integrates with Chef and Puppet and supports complete life-cycle systems management and monitoring. OpenStack Heat implements deployment of virtual clusters with its own template defining resources and relations between them. This is integrated with Chef and Puppet. The complexity of system specification is illustrated by Boto that supplies Python interface to Amazon cloud functionalities; there are 40 API’s. Well liked systems such as Rocks provide DevOps style services for traditional clusters. This emphasizes that our resource specification need to include core cluster scheduling (from Yarn to Slurm and Moab) and monitoring (Ganglia, Nagios, Inca ..) functionality

Note that TOSCA has similarities to the well-known OASIS standard WS-BPEL and this is not surprising. TOSCA specifies the deployment of a system and WS-BPEL its execution. We note that after initial interest, the Grid/HPC community did not find WS-BPEL useful and in fact use programmatic approaches with the user interface being workflow systems like Kepler, Pegasus, Swift and Taverna, which hide the mechanics of the orchestration. Here graphical user interfaces are often used to define systems. Maybe this is future for “Software Defined Systems”?

A project Cloudmesh from Indiana University integrates many of the above tools into a Python based interface and Nimbus from Chicago developed similar capabilities. Add Utah

We have reviewed a rich variety of systems that testifies to the importance of area but apart from a few de facto standards like Libcloud, Chef or Puppet, there is a rather chaotic situation that handicaps interoperability of executing jobs on different providers and further makes it difficult to support “reproducible data/computational science”. How can one specify execution today in a way that it can be re-executed later to verify and extend results?

The tools discussed above are reviewed at <http://bigdataopensourceprojects.soic.indiana.edu/#section2> (layers 6 and 7)