Building the SLATE Platform

Joe Breen
University of Utah
joe.breen@utah.edu

Lincoln Bryant
University of Chicago
lincolnb@uchicago.edu

Gabriele Carcassi
University of Michigan
carcassi@umich.edu

Jiahui Chen
University of Utah
jiahui.chen@utah.edu

Robert W. Gardner*
University of Chicago
rwg@uchicago.edu

Ryan Harden
University of Chicago
hardenrm@uchicago.edu

Martin Izdimirski
University of Utah
nitram.izdi@gmail.com

Robert Killen
University of Michigan
rkillen@umich.edu

Ben Kulbertis
University of Utah
ben.kulbertis@utah.edu

Shawn McKee
University of Michigan
smckee@umich.edu

Benedikt Riedel
University of Chicago
briedel@uchicago.edu

Jason Stidd
University of Utah
jason.stidd@utah.edu

Luan Truong
University of Utah
luan.truong@utah.edu

Ilija Vukotic
University of Chicago
ivukotic@uchicago.edu

ABSTRACT

We describe progress on building the SLATE (Services Layer at the Edge) platform. The high level goal of SLATE is to facilitate creation of multi-institutional science computing systems by augmenting the canonical Science DMZ pattern with a generic, "programmable", secure and trusted underlayment platform. This platform permits hosting of advanced container-centric services needed for higher-level capabilities such as data transfer nodes, software and data caches, workflow services and science gateway components. SLATE uses best-of-breed data center virtualization and containerization components, and where available, software defined networking, to enable distributed automation of deployment and service lifecycle management tasks by domain experts. As such it will simplify creation of scalable platforms that connect research teams, institutions and resources to accelerate science while reducing operational costs and development cycle times.

CCS CONCEPTS

• **Computer systems organization → Grid computing, Edge Computing**;

KEYWORDS

Distributed computing, Containerization, Edge computing

ACM Reference format:
Joe Breen, Lincoln Bryant, Gabriele Carcassi, Jiahui Chen, Robert W. Gardner, Ryan Harden, Martin Izdimirski, Robert Killen, Ben Kulbertis, Shawn McKee, Benedikt Riedel, Jason Stidd, Luan Truong, and Ilija Vukotic. 2018. Building the SLATE Platform. In *Proceedings of Practice and Experience in Advanced Research Computing, Pittsburgh, PA, USA, July 22–26, 2018 (PEARC '18),* 7 pages.
https://doi.org/10.1145/3219104.3219144

*Corresponding Author

PEARC '18, July 22–26, 2018, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-6446-1/18/07…$15.00
https://doi.org/10.1145/3219104.3219144

1 MOTIVATION

Multi-institutional research collaborations propel much of the science today. These collaborations require platforms connecting experiment facilities, computational resources, and data distributed among laboratories, research computing centers, and in some cases commercial cloud providers. The scale of the data and complexity of the science drive this diversity. In this context, research computing teams strive to empower their universities with emergent technologies which bring new and more powerful computational and data capabilities that foster multi-campus and multi-domain collaborations to accelerate research. Recently, many institutions invested in their campus network infrastructure with these goals in mind. Yet even for the most advanced, well-staffed, and well-equipped campus research computing centers the task is daunting. Acquiring and maintaining the full scope of cyber-engineering expertise necessary to meet the complex and expanding demands of data and computationally driven science is too costly and does not scale to the full spectrum of science disciplines. The diversity of computation, data and research modalities all but ensures that scientists spend more time on computation and data management related tasks than on their domain science while research computing staff spend more time integrating domain specific software stacks with

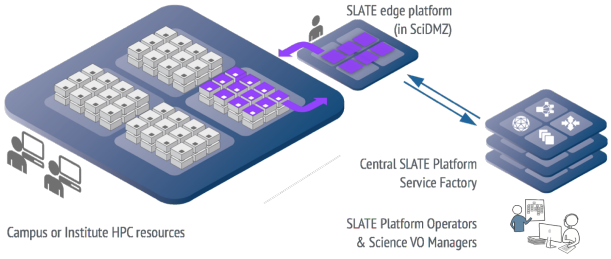

Figure 1: SLATE implemented in the local HPC context

limited applicability and sustainability beyond the immediate communities served. Capabilities elsewhere are not available locally, and vice versa. How should campus and HPC resource providers evolve their cyberinfrastructure to more easily incorporate data infrastructure building blocks developed in other contexts?

2 APPROACH

We address this challenge in complexity and scaling by introducing a Services Layer At The data center "Edge" (SLATE) which enables distributed automation, centralized delivery and operation of data, software, gateway and workflow infrastructure. Much as Google re-imagined the data center [41] and initiated a wave of data center virtualization development, we view advanced "cyberinfrastructure as code" as an appropriate metaphor for transforming the way advanced science platforms are built and operated. SLATE will augment the Science DMZ pattern [33] by adding a secure and trusted "underlayment" platform to host advanced data and software services needed for higher-level, connective functions between data centers. These connective services could include, for example, a domain specific content delivery network endpoint, a collaboration data cache, a job (workflow) scheduler, a service component of a distributed science gateway, an http-based software cache, or a resource discovery service (for higher level, meta scheduling systems).

A major focus of SLATE will be development of community accepted Science DMZ patterns capable of supporting advanced and emerging technologies while respecting local site security policies and autonomy. In this project, we are focusing on production services for data mobility, management, and access as driven by data-intensive science domains. For operators of distributed data management services, content delivery networks, and science gateways SLATE will closely resemble the NIST definition [38] of a PaaS (Platform-as-a-Service) though the resemblance does not limit the underlying infrastructure to a cloud context. We are leveraging experience and lessons learned in the deployment and operation of data systems linking over 60 data centers for the LHC computing grid. Figure 1 gives a schematic of the SLATE concept in the local HPC context.

The SLATE concept should accommodate large, well-equipped HPC centers, research computing facilities at institutions with fewer resources, as well as commercial cloud providers. Modern HPC centers which support data intensive science typically have a Science DMZ which hosts dedicated data transfer nodes, perfSONAR [1, 28] measurement hosts, and security and enforcement policies needed for high performance, wide area applications. A dedicated SLATE edge cluster will augment the existing Science DMZ by offering a platform capable of hosting advanced, centrally managed research computing edge services including, where the local infrastructure permits it, the deployment of virtual circuits and other software defined networking constructs [2, 37]. For a small institution with limited resources, the SLATE concept may provide a complete Science DMZ infrastructure to more quickly integrate these institutions into centrally managed research platforms. The SLATE concept will allow local resource administrators the ability to simply install the infrastructure while offering central research groups the services needed for mdanagement of software and science tools used on the platform. Thus, a local researcher in a small institution could focus on the science and connecting any requisite science instrumentation, while the local IT staff would not have the burden of trying to understand the science requirements, application software dependencies, data and workflow particulars, etc. A good science use-case comes from a medium sized collaboration to detect dark matter.

2.1 A Platform for Dark Matter Searches

Observations of the cosmic microwave background fluctuation, large-scale galaxy surveys, and studies of large-scale structure formation indicate that a large fraction of the matter in the universe is not visible. An exotic but as yet undiscovered elementary particle could explain these observations. Several experiments have been built in last two decades to prove the existence of such elusive particles but their detection has proved challenging as we do not have yet a clear picture of what they are and if they really exist.

The XENON1T [27] experiment, a two-phase xenon Time Projection Chamber has been built in the Laboratori Nazionali del Gran Sasso (LNGS) in Italy to study fundamental questions about the existence and make-up of dark matter. Commissioning began during the first few months of 2016, with the first large scale science run beginning in December 2016. Because of its one ton fiducial mass and ultra-low background, the XENON1T experiment is probing properties of dark matter in yet unexplored regions.

A data processing and analysis hub is hosted by the University of Chicago Research Computing Center to complement processing and analysis facilities in Europe (Stockholm is the European analysis hub). A distributed data management service for the collaboration was built so that experiment and simulation data sets at various stages of processing could be distributed and shared easily throughout the 22 member institutes. Figure 2 shows the deployment of the XENON1T Rucio service [35]. The deployment is a network of five storage endpoints (GridFTP-based data transfer nodes) in Europe and the U.S providing a highly scalable global namespace, a reliable and fast transfer service, a subscription ("placement rules") service, and a deletion service for data lifecycle management. The raw experimental data are uploaded at the experiment lab and automatically replicated to specific HPC centers for processing, and data products from those systems are registered into the system for distribution to the analysis hubs. The Rucio client tools provide a uniform data access model for the various endpoints.

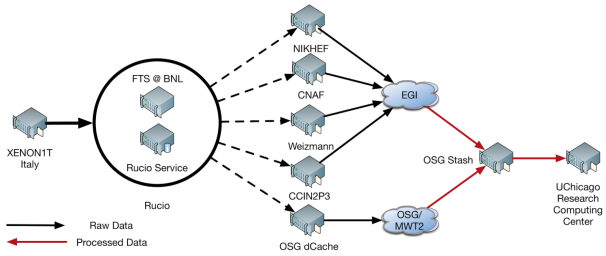

Figure 2: The multi-institution data management platform for the XENON collaboration

A central server for managed file transfer is hosted at Brookhaven National Laboratory, and central file and dataset catalogs and service agents are hosted by the University of Chicago. At each of the storage endpoints is a GridFTP service. Each of these services is managed individually so that the platform itself requires efforts from 10 individual administrators. With a SLATE hosted platform, the services, configuration, monitoring and optimization could be managed by a single operator, requiring only basic server management at the endpoints.

We envision the SLATE platform as a key component for scaling and operating data placement services like Rucio for multi-institution science collaborations like XENON1T, especially where local expertise in complex scientific software stacks may be limited.

3 ARCHITECTURE

SLATE leverages advances made by previous testbeds [12, 29, 30, 36, 39] and similar platforms [8, 19] and integrates best-of-breed data center virtualization and service orchestration technologies from the open source community. The SLATE team has created a platform architecture [23] with a centralized service deployment model in mind. Users coming to the SLATE platform will interact with a centralized API which the SLATE team is is currently developing. This API will have a RESTful design which both the SLATE command line tool and the web portal will use. The SLATE provisioning service, while firstly targeting the (bare metal) SLATE edge nodes, will have the ability to handle multiple infrastructure types, including various public and private cloud providers.

The SLATE team has designed the platform to make deployment and operation as streamlined as possible. Figure 3 gives a schematic picture of the SLATE architecture. The SLATE Platform Portal will provide views for science data operators as well as for operators of the underlying infrastructure at a local institution.

3.1 Hardware Platform

The development of the SLATE platform is to facilitate high-throughput, next generation scientific computing applications to interact with both on-premise instruments and facilities, as well as, external upstream services such as those hosted in public clouds or other institutions. One of the key elements of the SLATE platform is a hardware configuration that will minimize the amount of local system administration expertise needed to run complex software stacks, and ideally work as a trusted appliance that lives on the Science DMZ. To that end, we are focusing on three key pieces for the hardware platform deployed at each site:

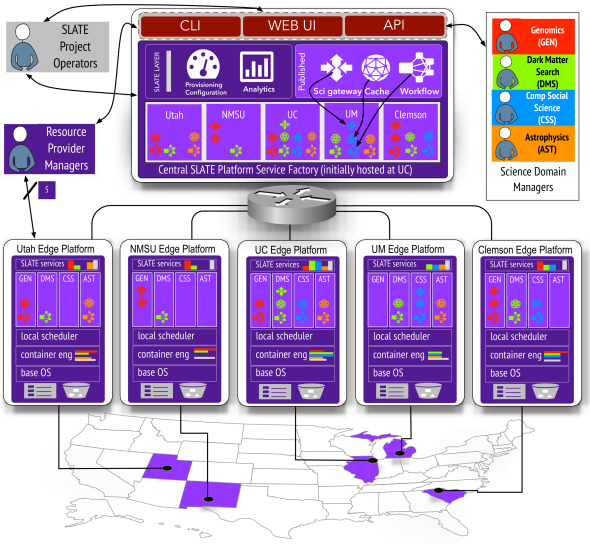

Figure 3: SLATE architecture

- A high performance networking device, either deploying a new device or reusing an existing high performance network device
- An out-of-band management server, to allow SLATE platform administrators to perform system maintenance and for SLATE central services to stage OS updates and configuration
- A scalable compute infrastructure, able to accomodate small sites while allowing for horizontal expansion for larger organizations

3.2 Container Orchestration and Federated Services

In order to realize the vision of deploying and operating edge service applications across disparate resources consistently and at scale, SLATE has adopted a Linux container-based approach to software deployment. This approach, as made popular by Docker and similar products, enables rapid development and deployment by presenting a uniform application environment, and encourages developers to build stateless services for scalability. In the SLATE context, this approach will allow edge service application developers to create services which they can build once and run on any SLATE node. This approach also simplifies the ability to scale and load balance across resources not only between sites, but across compute platforms within a site as well.

To facilitate deployments across geographically distributed hardware, the SLATE platform has the capability to federate individual sites. This federation of the individual sites allows scientific edge service application developers to deploy their application at one location and have it replicate across all sites authorized to host the service. Updating an application will follow the same principles, thereby allowing a consistent deployment among all collaborator sites. To facilitate federated applications, the SLATE platform

allocates and schedules edge service resources, ensures network connectivity between deployments, configures DNS entries and manages load balancing between sites. In addition, SLATE gives platform and site operators some control over how edge service resources slice and allocate to respective scientific groups sharing the platform.

3.3 User-Facing Services and Tools

In addition to providing a consistent hardware platform and container-based edge service application deployment function, SLATE provides a set of tools to manage the full edge service application lifecycle. This lifecycle includes:

- Application development: taking an existing scientific edge service application and configuring it to run on the SLATE infrastructure
- Application registration: managing a catalog of edge services that have been approved to be installed on a SLATE platform, either for testing or for general consumption
- Application usage: taking an application from the catalog, configuration and deploying on the infrastructure

SLATE provides a development environment, which comprises a set of images that can run a virtualized version of a SLATE platform on a development machine. This development environment utilizes technologies such as VirtualBox, minikube, etc., and allows SLATE application developers to work on packaging the application in an isolated environment.

SLATE also provides an edge service application catalog which the SLATE platform administrator can use to manage the list of applications to install on the SLATE platform. We use technologies already established in the cloud community such as Kubernetes [15, 31], Helm[17] and GitHub[13].

As a final piece, SLATE will also need to provide tools to install/monitor/uninstall applications from each SLATE platform. The tools will also need to allow the configuration of the application for the specific use within a science domain.

A good part of the functionality will be provided to the user by interacting to the SLATE service through either a command line interface (CLI), a web user interface (WebUI) or an application programming interface (API). The service will implement the actual logic and expose it through a REST interface which is then consumed by the three different client types. The SLATE service will try to rely as much as possible on the state already provided by the tools it builds upon thus avoiding the problem of synchronize the duplicated state.

3.4 Edge Service Applications

A fundamental goal of the SLATE platform is to host edge service applications in support of scientific computing, such as science gateways. The SLATE platform is able to host various services such as data transfer nodes with Globus Connect[40], caching applications such as Squid[24] and XCache[25], resource sharing applications for job submission and storage sharing, as well as applications used for data organization, delivery and analysis, needed for scalable, multi-institutional collaborations.

All project software running on the SLATE platform will run inside of an application container. This approach allows great flexibility, as experts in a particular domain can provide a specific image for their application. To facilitate allocation and scheduling, all edge service application containers will have to declare a set of resource requirements. These requirements may include disk utilization, minimum throughput (MB/s), firewall ports, memory and CPU. The SLATE platform will be able to compare these resource declarations to the available resources, and provide a candidate list of resource targets for the application. The platform may go further and suggest changes that could be made to the application for greater proliferation.

3.5 Logging and Monitoring

Though SLATE comprises geographically disparate hardware, the SLATE team maintains central logging and monitoring in order to assess the health of the distributed system. Centralized monitoring comprises active measurement, i.e. perfSONAR, passive monitoring and log aggregation through tools such as: Check_mk[5], Elastic Stack[10], and OSSIM[3]

3.6 Security

The nature of the SLATE platform and its role in providing "programmable" access to edge resources makes it critical that we carefully consider all aspects of security in the design and implementation of SLATE. Sites that deploy SLATE need to be confident that their edge resources will achieve appropriate utilization and will be robust against attacks.

A key emphasis is that SLATE, by providing a well defined API, enables straightforward vulnerability testing. SLATE provides a specific set of capabilities which external entities can review and analyze in order to help ensure the platform is resilient to attack, and that the platform is following applicable security best practices. This description is in stark contrast with the situation many sites find themselves when supporting multiple science domains, each with their own unique software stacks and methods. SLATE provides the opportunity to create a specific, well-defined way to interact with a site's resources which a developer can test and validate.

The SLATE project has scheduled an engagement with the Center for Trusted Scientific Cyberinfastructure (CTSC)[4] and we are already engaging campus security officers to help ensure we are designing a functional and secure cyberinfrastructure component. This external security discussion is in addition to an internal project-driven focus on providing security as a fundamental design principle in our development, prototyping and testing of SLATE.

4 DEMONSTRATION OF ALPHA SLATE DISTRIBUTED PLATFORM

4.1 Status of Sites

As part of its distributed alpha platform, the SLATE team has deployed clusters at three sites: University of Chicago, University of Michigan, and the University of Utah. For rapid prototyping purposes, the team has deployed these clusters on virtual machines that the team can tear down and re-instantiate at-will. The virtual machines also allow scaling investigation to act as input to the final

hardware design. Currently, we have started architecting the design of the command-line client and will implement a web portal at a later date.

Since the rapid deployment of science services across multiple sites is a key obective, the SLATE team has been investigating container deployment technologies. While several container orchestration platforms exist, the team is prototyping Kubernetes as an orchestration mechanism based on team members' initial work, and, based on prevalence in different communities and the market. The team has tested Kubernetes on different operating systems such as CentOS, Ubuntu, RancherOS[21], and CoreOS[7]. The objective of testing the different operating systems was to find a match between what sites use, what sites support, and how to best support and integrate the SLATE platform. Most HPC environments deploy RedHat or its derivatives (such as CentOS or Scientific Linux) due to the long term stability. However, many Kubernetes deployments run on Ubuntu due to its commercial popularity, more recent kernel, and development tools. Specific container-focused Linux distributions such as RancherOS and CoreOS have gained popularity due to being relatively light-weight, having customized Kubernetes deployment systems, and custom tooling for update management.

The alpha configurations at each site chose different operating systems in order to gain some experience with each. This work has led to an investigation of creating an alpha image on CoreOS as the packaged thrust. One of the key advantages of CoreOS is the ease of patch management without taking down an entire site for upgrades. A challenge of CoreOS that the team is still exploring relates to driver support for different hardware, such as GPUs.

4.2 SLATE Development Image

The SLATE team has derived the alpha base image from CoreOS Container Linux, which has a number of features that make it attractive for the SLATE platform. The team can deploy Container Linux as a RAM-resident image, so that containers deploy to disk while the operating system remains ephemeral. For SLATE, this configuration allows for atomic updates via images that can be built and distributed by a centralized upstream system. In the current iteration, a build host with the Container Linux SDK has been prepared, and SLATE-specific changes are pushed to the image at build time. Currently, the team has limited the SLATE-specific changes to installing and configuring a single-node Kubernetes deployment via kubeadm.

In lieu of forking Container Linux itself, we apply our changes to a base Container Linux image at boot time, using CloudInit [6]. CloudInit provides a YAML [26]-based configuration language where we define services, configuration files, scripts for fetching binaries from our upstream, etc. Currently, we are deploying the CloudInit configuration as part of the development image, but once we deploy the SLATE hardware platform, we will use a vanilla CoreOS image and apply the SLATE-specific changes at boot. These images and configurations will either stage into, or proxy through, the out-of-band management device.

4.3 Initial Federation Testing

The SLATE alpha federation deployment leverages the Kubernetes Federation project[16]. This project provides two major building blocks: the ability to sync resources across clusters and the ability to do cross-cluster discovery for DNS and load balancing purposes. SLATE is primarily leveraging the first building block in order to obtain the ability to present a single point of deployment for an application.

As part of the initial development work to test the concepts of federation, the SLATE team built a small federation platform for testing. This mini-federation platform allows for initial understanding of the federation technology and quick prototyping of ideas. The SLATE team has also federated two discrete sites at the University of Michigan and the University of Utah. A small NGINX application has been successfully deployed at the University of Michigan and the University of Utah using the federation tooling.

Testing the current state of federations as provided by Kubernetes has revealed a number of key areas that require work. An initial issue was that some of the tooling that we used to deploy Kubernetes was not sufficiently flexible for federations to work properly. SLATE tooling has taken this issue and other issues into consideration for future deployment iterations.

Aside from bugs and minor issues in the Kubernetes federation control plane, we found that two key areas were lacking in our tests. First, the federation tooling required that we maintain consistent naming and labeling across multiple sites. This requirement implies that either some upstream contributions will need to integrate into Kubernetes itself, or the SLATE subsystem responsible for site-level configuration will need to ensure the naming and labeling maintains the proper consistency, external to Kubernetes. The second issue is that the federation tooling, as it currently exists, requires a high level of permissions (and therefore trust) between sites and site administrators. This particular issue presents serious security considerations for SLATE deployments across disparate resources.

4.4 Progress of CLI Development

To expedite the development of the command-line tools and services, we are currently developing the CLI as a single tool with no distinction between frontend and backend. By getting the tool into the hands of developers quickly, this allows us to rapidly iterate upon the semantic definition of the commands we need to provide. However, the current implementation keeps the interface fully separate from the backend logic, such that a client/server model can implement later as the other SLATE platform components develop.

We have chosen to develop the command-line tooling in Go[14], as this language is the language of choice to develop Kubernetes and related projects in this space, such as Docker[9] and etcd[11]. One of the challenges we have encountered in developing the SLATE CLI is that many of the underlying tools and components are themselves under heavy development and thus moving very rapidly. While this rapid development is a positive sign that the Kubernetes and associated projects have a healthy and active developer community, this rapid change has forced us to take a conservative stance with regard to how we interface with these tools. Currently, the we have prototyped the following commands:

```
slate dev start
```

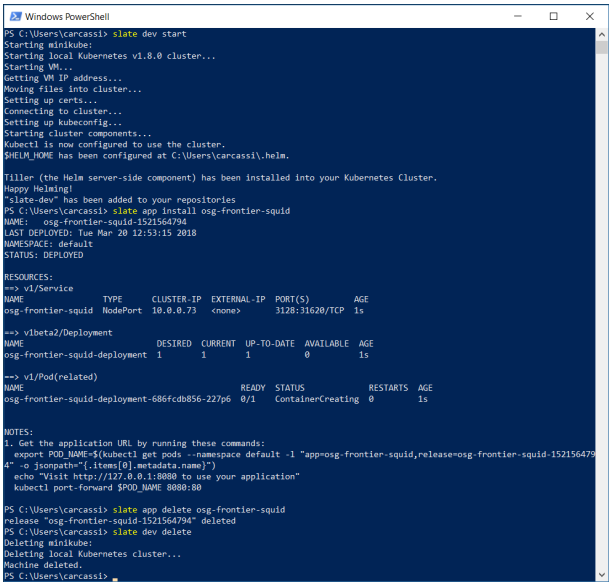

Figure 4: Example of command line installation lifecycle

This command starts the development environment. This environment consists of minikube[18] (which is assumed to be available and properly installed) properly configured with Helm and the SLATE application catalog[22].

```
slate app install
```

This command installs an application from the application catalog.

```
slate app delete
```

This command removes already installed applications.

```
slate dev delete
```

This command tears down the development environment. Figure 4 shows an example of the working tool.

4.5 Applications

The SLATE team has deployed several applications on its alpha platform.: JupyterHub, perfSONAR, Elasticsearch, and Kibana. The team has also created an XCache [25] prototype in order to demonstrate caching, an essential feature of future distributed data delivery networks.

The team is working to prototype a process for packaging these using a combination of SLATE developed tools, Helm and other tools. The SLATE deployment process will comprise the testing and integration of the application container, as well as the ability to deploy the application container across the federation.

The SLATE team has encountered a number of challenges which the team is attempting to mitigate in its packaging and deployment process. Some of those challenges are:

- Library support
- Helm support
- Package builds focused on virtual machine installations as opposed to containers
- Location of static config files for packages so that containers can be ephemeral

4.5.1 XCache deployment. XCache is a caching service that can reduce latency and reduce bandwidth needs of the wide area network accesses based on the XRootD [34] protocol. Reliable operation of the XCache service, without local systems administrator input, is very important for the successful utilization of network resources by the experiments at the CERN Large Hadron Collider which must deliver data to hundreds of HPC centers. Resources needed by the application depend on the scale of the computing resources served by the cache and range from a single caching node with 10 Gbps network interface card and a few terabytes of disk, to a cluster of caching nodes and hundreds of terabytes of disk. The single-node use case is very simple: a single pod with the xrootd server is run, with only one service exposing port 1094. Cluster installation is more complex requiring two services per node (xrootd and cmsd), and a "master" service that unifies them. Client authentication is based on X509[32] certificates, while the XCache service itself authenticates against data origins (XRootD services at different storage elements) using so-called “robot" certificates. Figure 5 shows a production implementation which uses SLATE as the management layer in the edge network. A production deployment will have three more services:

- Service registration. A health-check probe will automatically register and enable an XCache endpoint in an information service, and blacklist it when necessary.
- Reporting of cache state. In addition to caching blocks of data accessed, XCache stores important metadata per file. This metadata includes: number of accesses, times of first and last access, number of blocks accessed, etc. All this information is periodically aggregated and reported to a central monitoring service.
- Summary stream reporting. Every minute, the service reports important operational parameters of the XCache server, including number of connections, number of errors, memory used, etc.

XCache is one of the first science applications supported on the SLATE platform, and has driven requirements for the initial alpha platform. The team has packaged XCache for a Kubernetes deployment, including a Dockerfile, Kubernetes YAML files, scripts to create and retrieve shared secrets, etc. All code and configuration lives on Github, and hooks are enabled to automatically build XCache via DockerHub.

4.6 Logging and Monitoring

Logging and monitoring of a distributed infrastructure requires both a central and distributed approach. The University of Chicago has deployed a central Elastic Stack[10] and Prometheus[20] for gathering data from the remote sites. For visualization, Grafana allows for different dashboards and perspectives. Each site is running perfSONAR and the Elastic Stack to support active measurement and passive collection.

5 SUMMARY & OUTLOOK

The SLATE team has completed a first iteration of its alpha version platform distributed and federated over three sites. The next iteration in the project will be to harden the packaging of the image, package example applications for deployment, test the loading and

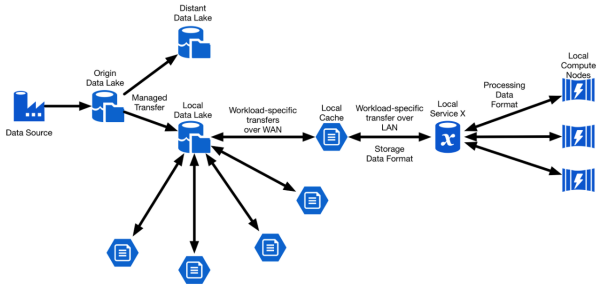

Figure 5: Example distributed caching and data delivery service using SLATE

A data lake service can deliver experimental data to processing facilities using a network of caching servers and purpose-built delivery services that transform the data into the needed format.

scaling of the platform, finalize the hardware configuration based on loading and scaling, and start to explore the network aspects. The project welcomes contributions and participation from groups and individuals focused on building distributed research platforms and gateway systems. Interested readers may follow developments at http://slateci.io/.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation Office of Advanced Cyberinfrastructure (OAC), grant number 1724821.

REFERENCES

[1] 2005-2009. perfSONAR-PS Publications. (2005-2009). http://www.perfsonar.net/publications.html
[2] 2016. GENI Network Stitching Architecture. (2016). http://groups.geni.net/geni/wiki/GeniNetworkStitching
[3] 2018. AlienVault OSSIM. (2018). https://www.alienvault.com/products/ossim
[4] 2018. Center for Trusted Scientific Cyberinfrastructure. (2018). https://trustedci.org/
[5] 2018. Check_Mk. (2018). Retrieved 2018/03/26 from https://mathias-kettner.de/check_mk.html
[6] 2018. Cloud-Init: multi-distribution cloud initialization. (2018). http://cloudinit.readthedocs.io/en/latest/
[7] 2018. CoreOS Container Linux. (2018). https://coreos.com/os/docs/latest
[8] 2018. Cyverse life sciences platform. (2018). http://www.cyverse.org/about
[9] 2018. Docker Website. (2018). https://www.docker.com/
[10] 2018. Elastic Stack. (2018). https://www.elastic.co/guide/index.html
[11] 2018. etcd Documentation. (2018). https://coreos.com/etcd/
[12] 2018. GENI (Global Environment for Network Innovations), a virtual laboratory for networking and distributed systems research and education. (2018). https://www.geni.net/
[13] 2018. GitHub. (2018). https://github.com/
[14] 2018. The Go Programming Language. (2018). https://golang.org/
[15] 2018. Kubernetes | Production-Grade Container Orchestration. (2018). https://kubernetes.io/
[16] 2018. Kubernetes Federation Project. (2018). https://kubernetes.io/docs/concepts/cluster-administration/federation/
[17] 2018. The Kubernetes Package Manager: Kubernetes Helm. (2018). https://github.com/kubernetes/helm
[18] 2018. Minikube GitHub Repository. (2018). https://github.com/kubernetes/minikube
[19] 2018. The Pacific Research Platform. (2018). http://prp.ucsd.edu/
[20] 2018. Prometheus Monitoring and Alerting Toolkit. (2018). https://prometheus.io/
[21] 2018. RancherOS Website. (2018). https://rancher.com/rancher-os/
[22] 2018. SLATE Application Catalog. (2018). https://github.com/slateci/slate-catalog
[23] 2018. SLATE Architecture Doc. (2018). https://docs.google.com/document/d/18F1qV3WGmcz7fxr95aCUAgR-_qqz1iQEAMF-AFTsuoE/edit?usp=sharing
[24] 2018. Squid. (2018). Retrieved 2018/03/26 from http://www.squid-cache.org
[25] 2018. XCache Documentation. (2018). http://slateci.io/XCache/
[26] 2018. YAML Ain't Markup Language. (2018). http://yaml.org/
[27] E. Aprile et al. 2016. Physics reach of the XENON1T dark matter experiment. *JCAP* 1604, 04 (2016), 027. https://doi.org/10.1088/1475-7516/2016/04/027 arXiv:physics.ins-det/1512.07501
[28] E. Boyd A. Brown M. Grigoriev J. Metzger M. Swany M. Zekauskas B. Tierney, J. Boote and J. Zurawski. 2009. Instantiating a global network measurement framework. In *SOSP Workshop on Real Overlays and Distributed Systems (ROADS'09)*. Big Sky, MT, USA.
[29] Ilia Baldine, Yufeng Xin, Anirban Mandal, Paul Ruth, Chris Heerman, and Jeff Chase. 2012. ExoGENI: A Multi-domain Infrastructure-as-a-Service Testbed. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer Berlin Heidelberg, 97–113. https://doi.org/10.1007/978-3-642-35576-9_12
[30] Mark Berman, Jeffrey S. Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. 2014. GENI: A federated testbed for innovative network experiments. *Computer Networks* 61 (mar 2014), 5–23. https://doi.org/10.1016/j.bjp.2013.12.037
[31] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. 2016. Borg, Omega, and Kubernetes. *ACM Queue* 14 (2016), 70–93. http://queue.acm.org/detail.cfm?id=2898444
[32] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. 2008. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. RFC Editor. http://www.rfc-editor.org/rfc/rfc5280.txt http://www.rfc-editor.org/rfc/rfc5280.txt.
[33] Eli Dart, Lauren Rotman, Brian Tierney, Mary Hester, and Jason Zurawski. 2013. The Science DMZ. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*. ACM Press. https://doi.org/10.1145/2503210.2503245
[34] Alvise Dorigo, P Elmer, Fabrizio Furano, and A Hanushevsky. 2005. XROOTD - A highly scalable architecture for data access. 4 (04 2005), 348–353.
[35] V Garonne, R Vigne, G Stewart, M Barisits, T B eermann, M Lassnig, C Serfon, L Goossens, and A Nairz and. 2014. Rucio – The next generation of large scale distributed system for ATLAS Data Management. *Journal of Physics: Conference Series* 513, 4 (jun 2014), 042021. https://doi.org/10.1088/1742-6596/513/4/042021
[36] J. Mambretti, J. Chen, and F. Yeh. 2015. Next Generation Clouds, the Chameleon Cloud Testbed, and Software Defined Networking (SDN). In *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*. 73–79. https://doi.org/10.1109/ICCCRI.2015.10
[37] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow. *ACM SIGCOMM Computer Communication Review* 38, 2 (mar 2008), 69. https://doi.org/10.1145/1355734.1355746
[38] P M Mell and T Grance. 2011. *The NIST definition of cloud computing*. Technical Report. https://doi.org/10.6028/nist.sp.800-145
[39] Robert Ricci, Eric Eide, and The CloudLab Team. 2014. Introducing CloudLab: Scientific Infrastructure for Advancing Cloud Architectures and Applications. *USENIX ;login:* 39, 6 (Dec. 2014). https://www.usenix.org/publications/login/dec14/ricci
[40] S. Tuecke, R. Ananthakrishnan, K. Chard, M. Lidman, B. McCollam, S. Rosen, and I. Foster. 2016. Globus auth: A research identity and access management platform. In *2016 IEEE 12th International Conference on e-Science (e-Science)*. 203–212. https://doi.org/10.1109/eScience.2016.7870901
[41] Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the European Conference on Computer Systems (EuroSys)*. Bordeaux, France.