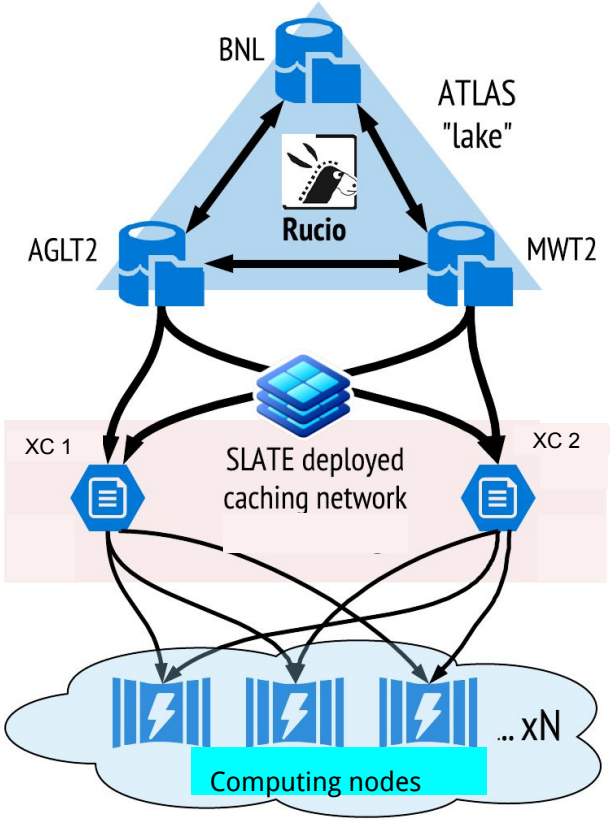
SLATE deployment example
XCache

Ilija Vukotic
University of Chicago

WLCG pre-GDB Dec 10th, 2019

What is XCache?

- ATLAS has an R&D effort for data delivery layer for HL-LHC (http://bit.ly/ddn-rd)
- Using software from the XrootD group called "XCache"
- XCache is a specially configured XrootD server. User accessing the data through the XCache server from an upstream XrootD server will see his data cached. Cache can be configured as a full file cache (with optional prefetch) or block level cache.



What is XCache? - technical view

Basically an xrootd proxy server that also stores data passing through it. On next access it delivers data from disk.

It needs:

- “Dedicated” node
- Local storage
- IP
- Secrets (to authenticate against origin servers)
- Integration with ATLAS workflows (RUCIO, AGIS, monitoring)

Lessons from FAX

- FAX (Federated ATLAS Xrootd) was a project to connect all ATLAS storage in a global network, enabling users to access all of the ATLAS data from anywhere.
- At a small scale, things worked, but **operations became much harder as we scaled**
  - (FAX at its peak had 65 sites all over the world)
- XrootD endpoints would inevitably fail
  - Made an alert system when it happens and we would contact sys admins at sites.
  - **Simple restart often took days and weeks.**
- Debugging issues could take months
  - sending back and forth tens of mails, logs, explaining operations, writing TWiki's with detailed instructions for all the different versions of storage (xroot, DPM, dCache, Lustre).
- Updating server version or changing a configuration parameter was slow and never complete: **half of sites would upgrade in a few days, one quarter in a month, the rest never.**

Toward a simpler operations model

- Wanted to be able to deploy new services to sites with less friction
- Adopted SLATE platform for deploying XCache
  - XCache container created, packaged as Helm chart, submitted into SLATE catalog
  - Sites who have Kubernetes with SLATE need only set up the node and allow access to the `atlas-xcache` group

XCache - structure

Three main parts:

- Server itself
- Proxy renewal
- Monitoring - this is application specific monitoring. In new version it will be replaced with Prometheus.

All in the same docker image auto-built in DockerHub from github repo.
K8s deployment has one pod with 3 containers.
Service is NodePort (for efficiency reasons).
There are liveness probe (simple http checking if anything is listening on the port) and a pre-stop that unregisters service from AGIS (service self-activates on startup).

Prerequisites

This being cache, it has a lot of prerequisites that “regular” application won’t have:

- A dedicated node with a lot of disks attached as JBODs. This node is labeled as `xcache-capable: "true"` and tainted with `effect: PreferNoSchedule.`
- Fixed IP.
- JBOD mount paths and IP have to be communicated to me.
- Extra large ephemeral storage for logs that tend to be large.

XCache HELM chart

Started from standard k8s yaml files, replaced everything that I wanted configurable.
Added SLATE required properties:

```
metadata:
 name: {{ template "xcache.fullname" . }}
 labels:
   app: {{ template "xcache.name" . }}
   chart: {{ template "xcache.chart" . }}
   release: {{ .Release.Name }}
   instance: {{ .Values.Instance }}
```

Created a pull request to Slate catalog github repo.

Deploying XCache

Once a site approved application, informed me of disk mounts, IP and labeled node I:

- Prepare configuration. Most of it are defaults.
- Create slate secret (xcache service certificate)

```
$ slate secret create --group atlas-xcache --cluster esnet-lbl
--from-file userkey=xcache.key.pem --from-file usercert=xcache.crt.pem xcache-cert-secret
```

- Deploy XCache

```
$ slate app install --group atlas-xcache --cluster esnet-lbl --conf ESnet.yaml xcache
```

- Check it works

```
$ xrdcp -f
root://198.129.248.94:1094//root://fax.mwt2.org:1094//pnfs/uchicago.edu/atlasdatadisk/rucio/
data15_13TeV/3b/5d/AOD.11227489._001118.pool.root.1 /dev/null
```

Supporting SLATE application

Now that there is one more person to support ATLAS XCache Slate deployments (Wenjing Wu), it becomes even nicer that SLATE keeps deployed configurations, so we don't overwrite each other.

Most of the things I normally do with kubectl I can do via SLATE. Takes a bit of time to get used to but definitely a lot of benefits:

- I don't bug sys admins anymore
- I can easily check application state
- Reconfigure
- Upgrade
- Tune parameters
- Check logs

Conclusion

For a production level applications that are stable and don't require too many "exceptions", SLATE makes deployment and management almost trivial.

Even for special applications like XCache, that are very picky and not rock-solid, but need to be deployed at more than a couple sites, it makes it worthwhile to do it in SLATE.

It spares a lot of nerves and effort to both sys admins and application support team.

Appendix

Docker container

Everything in a github repo and docker image built automatically in dockerhub, documentation in github too.

The image is rather basic:

- Based on centos
- Xrootd-server, xrootd-client, vomsxrd, fetch-crl, python,...
- xrootd user has fixed GID and UID
- Creates all directories needed, makes them owned by xrootd (but only if needed!)

Containers

3 containers run in each pod:

- xcache - server itself
- x509 - renews proxy
- reporter - collects info on cached files and sends to logstash

All server configuration done through environment variables.

XCache:

- Sets few default environment variables if not already defined.
- Sleeps 2 min for x509 container to finish first update of CA
- Starts server
- Activates itself in AGIS using REST API
- Sleeps indefinitely

X509:

- Updated x509 proxy
- Fetches crls
- sleeps 6 h

Reporter:

- Collects info from .cinfo files
- Reports to ES
- Sleeps 1h

Server - K8s deployment

Secrets: service certificate (2 files)

As k8s deployment (not a simple pod)

Since it requires special node it uses nodeSelector

You don't want anything else using this node so *

Volume to be used for caching is a hostPath

Liveness probe on server container

All configs done through environment variables. In hindsight it would be nicer to use ConfigMaps.

```
tolerations:
- key: "special"
  operator: "Exists"
  value: true
  effect: PreferNoSchedule
```

```
 - name: xcache-data
   hostPath:
     path: /scratch
```

```
livenessProbe:
    tcpSocket:
       port: 1094
     initialDelaySeconds: 180
     periodSeconds: 60
```

Service is a NodePort. IP is fixed.

```
type: NodePort
ports:
- protocol: TCP
  name: xrootd
  port: 1094
  targetPort: 1094
  nodePort: 31094
externalIPs:
- 192.170.227.151
```

Stress test - k8s deployment

Used to stress test any xcache instance and report about results.

Uses the same image, same secrets, just runs different code.

Helm chart

Maybe an overkill for app this simple, but makes config more readable.
Basically replaced values with placeholders like this:

```
containers:
  - name: {{ .Chart.Name }}
    image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
    imagePullPolicy: {{ .Values.image.pullPolicy }}
    env:
    - name: XC_SPACE_HIGH_WM
      value: "{{ .Values.XCacheConfig.HighWaterMark }}"
    - name: XC_SPACE_LOW_WM
      value: "{{ .Values.XCacheConfig.LowWaterMark }}"
```

Helm values

Clean and with a lot of comments (not shown here).

```
Instance: global
Service:
 Port: 1094
 ExternalIP: 192.170.227.151
SiteConfig:
 Name: MWT2
 AGISprotocolID: 433
Monitoring:
 Collector: http://uct2-collectd.mwt2.org:8080
XCacheConfig:
 CacheDirectory: /scratch
 HighWaterMark: 0.95
 LowWaterMark: 0.90
 RamSize: 16g
 BlockSize: 1M
 Prefetch: 0
 CertSecret: xcache-cert-secret
image:
 repository: slateci/xcache
 tag: latest
 pullPolicy: IfNotPresent
```