
Federated Varnish & XCache Deployment with SLATE

Throughput Computing 2023

Madison, July 12, 2023

Ilija Vukotic, University of Chicago



MANIAC LAB



ATLAS
EXPERIMENT

Service Deployment Models

There are services that we would like running at most of the ATLAS sites (eg. Perfsonar)

Standard way of doing it is to ask sys admins to run it:

Adding a service incurs a significant cost in sys-admin time:

- Learning about service
- Configuring
- Monitoring
- Keeping it up-to-date

People using a service need to communicate with sites, explain changes needed, debug things that break due to skipped updates, etc. sometime take months.

This makes for a unreliable service, security issues, slow rollout of new features, stifles innovation.



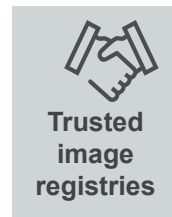
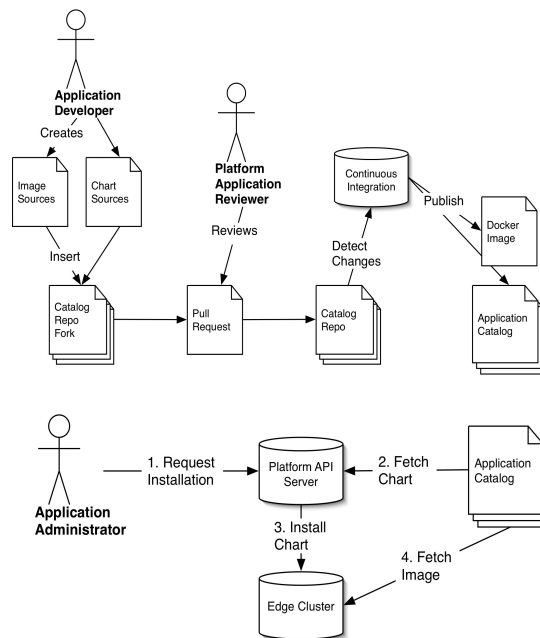
Service Deployment Models - cont'd

Federated way of doing it:

- Have one or (better) two people that know the service inside-out.
- Give them a secure way to deploy, (re)configure, monitor, start/stop/update it, with minimal/no involvement of site's personal.
- Site personal has only one-off things to do - NoOps.
- SLATE is one way to do Federated Ops.

SLATE: **S**ervices **L**ayer **A**t **T**he **E**dge

- **SLATE** - a value added K8s distribution
 - Support for CVMFS, ingress controller (multi-tenant, scoped privileges), Prometheus monitoring, **curated application catalog w/ Github Actions**
- Site security & policy conscious
 - SLATE works as an unprivileged user
 - Single endpoint via **institutional identity**
 - Site owner controls group whitelists & service apps; **retains full control**
- With OSG, WLCG, trustedci.org & others worked to establish a "CISO compliant" security posture and **new trust delegation model**



SLATE - Adding a service

Assuming an old-fashioned app...

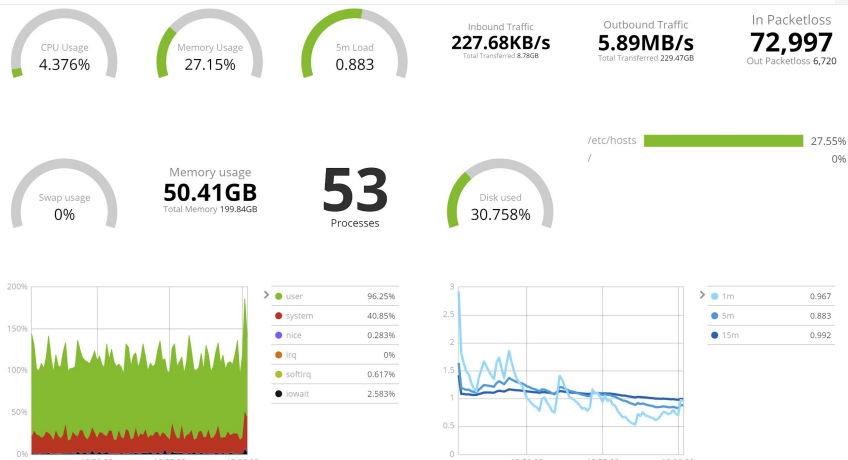
- Create docker image(s)
 - Directly from github using github actions, push them to a registry (DockerHub, OSG Harbor, CERN Harbor,...)
 - Check image scan reports.
- Create kubernetes deployments, services, ingress, etc. Test all works correctly (Docker Desktop).
- Create Helm chart
 - Basically decide what are the parameters that need to be configurable. Write instructions.
 - Add a few SLATE required lines.
- Add the chart to SLATE integration repository. Test.
- Add it to production repository.

SLATE - Managing a service

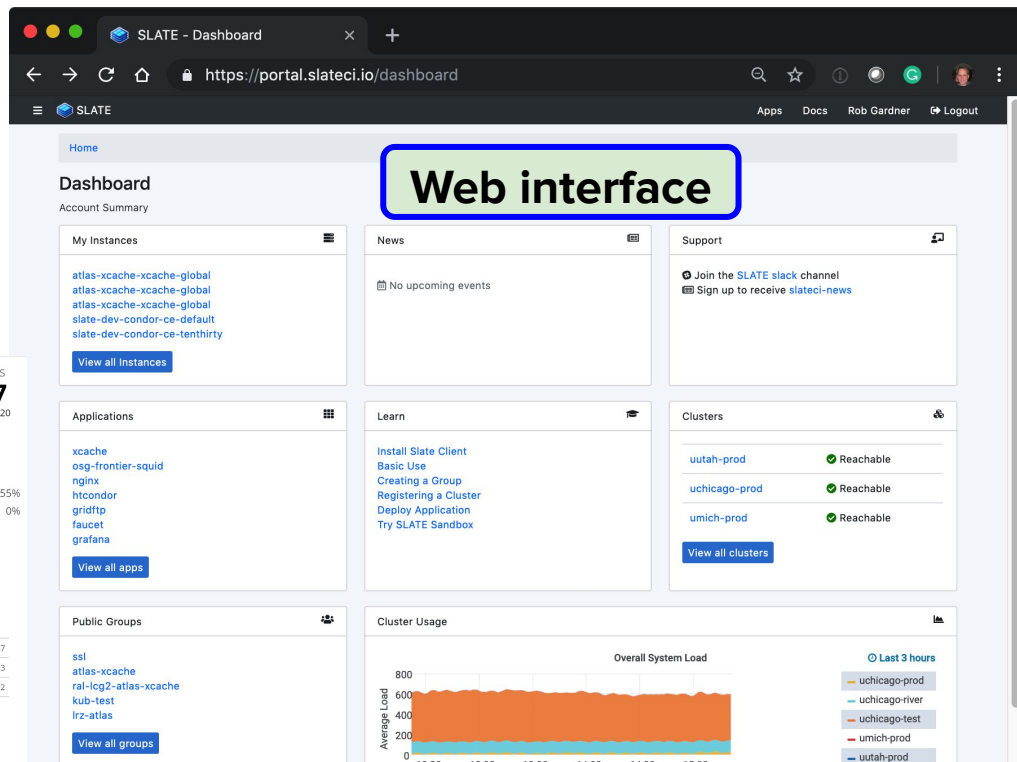
CLI

```
$ slate instance list
$ slate instance delete <instance name>
$ slate app install --group atlas-xcache --cluster uchicago-prod
--conf MWT2.yaml xcache
```

Kibana monitoring



Web interface



XCache

ATLAS has two situations where data is remotely accessed:

- **Virtual Placement** - jobs scheduled to sites that have no input data
- **ServiceX** - a service that quickly filters, enriches, delivers data in multiple formats for semi-interactive analysis.

Both practically require caching input data for faster subsequent accesses.

- The data is primarily accessed via xroot protocol.
- XCache is a specifically configured XRoot server that caches blocks accessed (also used by OSDF, CMS, etc.).

XCACHE in SLATE

A rather complex application with multiple containers:

- Server itself
- Proxy renewal
- Rucio heartbeats
- Monitoring stream udp2tcp proxy

Special requirements:

- NodePort service (for performance reasons)
- Dedicated node:
 - Special label `xcache-capable: "true"`
 - Tainted `effect: PreferNoSchedule.`
 - A lot of disks as JBODs
 - At least one NVMe for namespace
 - Good NIC (> 25Gbps)

Deploying XCache

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

- 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
```

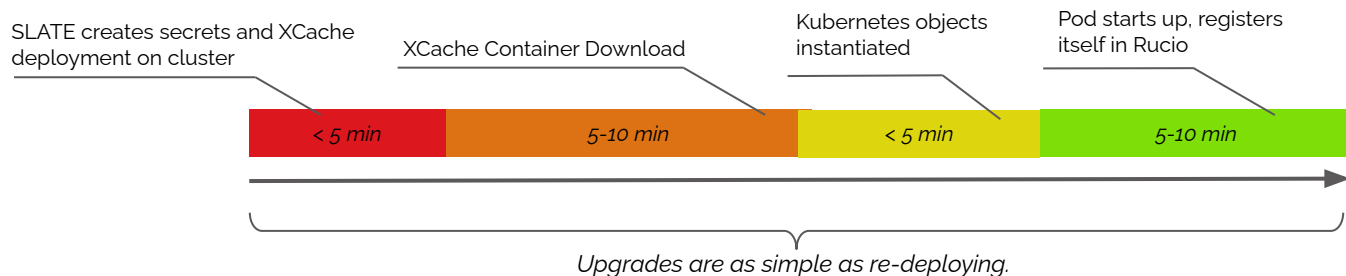
- Check it works

```
$ xrdcp -f  
root://198.129.248.94:1094//root://fax.mwt2.org:1094//pnfs/uchicago.edu/atlasdatadisk/rucio/  
data15_13TeV/3b/5d/A0D.11227489._001118.pool.root.1 /dev/null
```



XCACHE in SLATE

- We need it at all US Tier2s, Tier1, Analysis Facility, several UK and DE sites.
- Update of all SLATE instances takes <10 min, non-SLATE deployments take days to update.

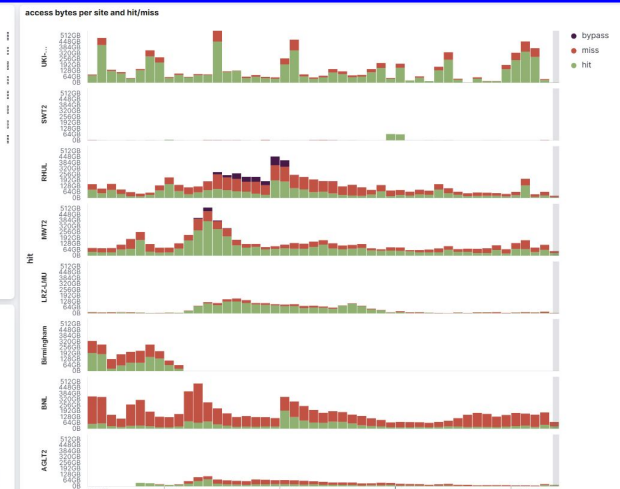
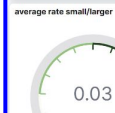
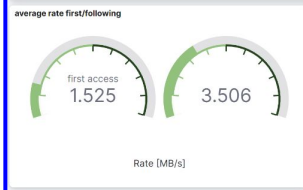
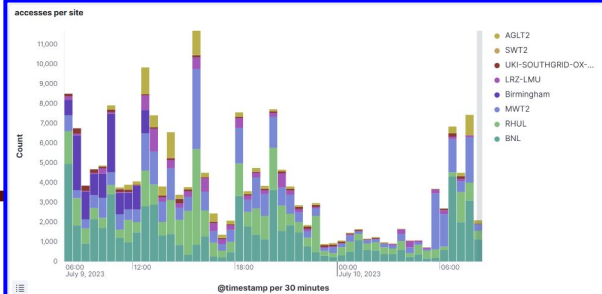


A data caching network deployed in less than 20 minutes.

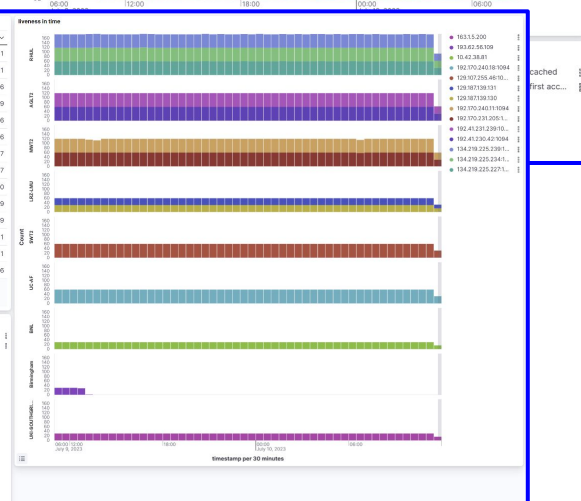
XCache monitoring

Several sources:

- SLATE
- gStream
- Panda
- Pilot
- Functional tests



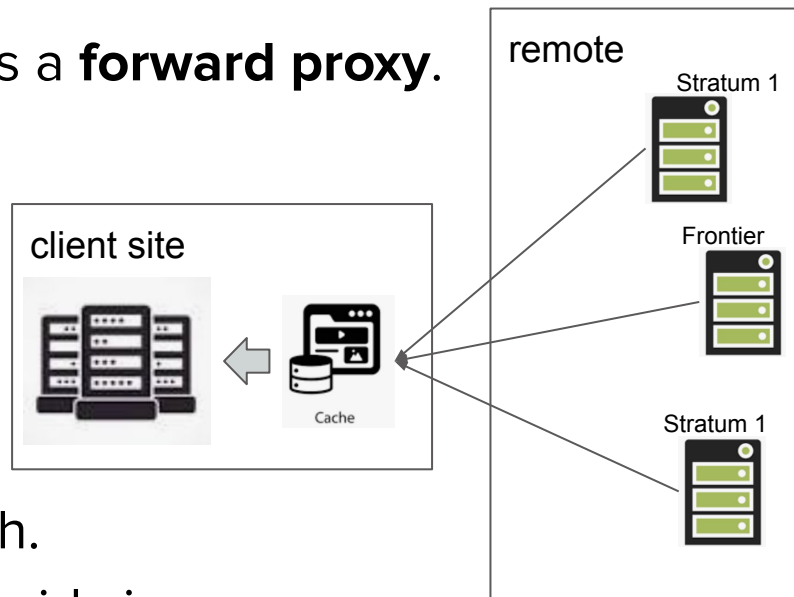
Site	Address	Count of records
AGLT2	192.41.230.42:1094	2,971
AGLT2	192.41.231.239:1094	2,971
BNL	10.42.38.81	1,486
Birmingham	193.82.56.109	119
LRZ-LMU	128.187.139.130	1,486
LRZ-LMU	128.187.139.131	1,486
MW2	192.170.231.205:1094	2,967
MW2	192.170.240.11:1094	2,957
RHUL	134.219.225.227:1094	2,930
RHUL	134.219.225.234:1094	2,929
RHUL	134.219.225.239:1094	2,929
SWT2	129.107.255.46:1094	2,971
UC-AF	192.170.240.18:1094	2,971
UKI-SOUTHGRID-OX-HEP	163.1.5.200	1,486



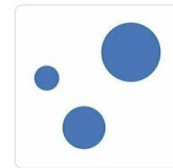
Squid



- We use Squids for http caching. Squid is a **forward proxy**.
- Single threaded, quite old technology.
- We use them for two different purposes:
 - to cache Frontier requests
 - to cache CVMFS accesses.
- Most sites have the same cache do both.
- Sites are recommended to have two Squids in a round robin configuration.
- Usually configured with 32GB RAM and a persistent disk cache.

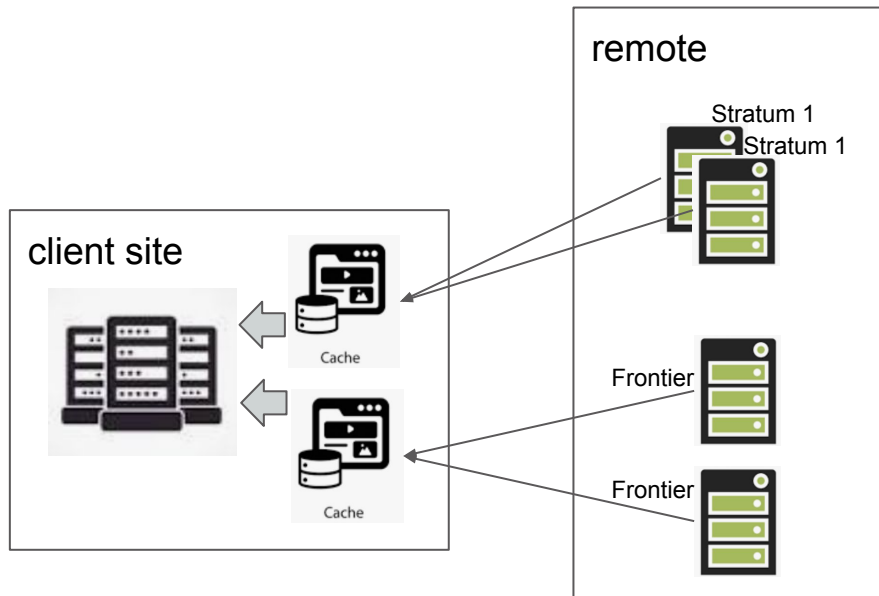


Varnish



Varnish

- “Made for modern hardware. Working with kernel not against it.”
- While it is a **reverse proxy**, in our case it doesn't matter as our origins are known.
- Very flexible - Varnish Cache Configuration Language (VCL) allows developers to specify request handling rules and set specific caching policies giving them a lot of control over what and how they cache.
- Nice modern monitoring.
- RAM only version is free, Disk persistence and federated versions are paid for.



Serving Frontier requests

- Varnish can be added to CRIC as a Squid and simply swapped in place.
- A VCL configuration.
 - ACL of WNs
 - List of backends
- Adding support for SNMP monitoring was 20x more effort.

```
vcl 4.1;
import dynamic;
import directors;

{{- range $nindex, $be := .Values.backends }}
backend {{ $be.name }} {
    .host = "{{ $be.host }}";
    .port = "{{ $be.port }}";
}
{{- end }}

acl local {
{{.Values.acl | nindent 4 }}
}

sub vcl_init {
    new vdir = directors.round_robin();
    {{- range $nindex, $be := .Values.backends }}
    vdir.add_backend({{ $be.name }});
    {{- end }}
}

sub vcl_recv {
    set req.backend_hint = vdir.backend();
    set req.http.X-frontier-id = "varnish";
    if (client.ip !~ local) {
        return (synth(405));
    }
    if (req.method != "GET" && req.method != "HEAD") {
        return (pipe);
    }
}
```

Serving CVMFS requests

- This is configured on WNs. In our case simple Puppet configuration change.
- A VCL configuration.
 - ACL of WNs
 - List of backends
 - Some complications:
 - correctly handling the fact that not all stratum 1 serve all repos.
 - Correct handling of requests for repos that don't exist anymore.

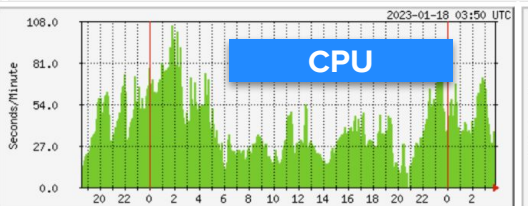
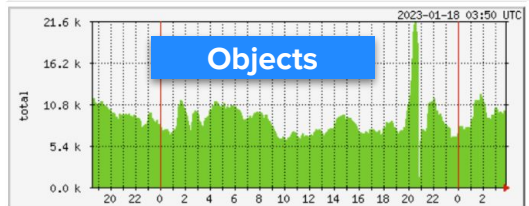
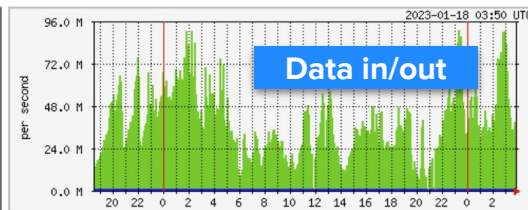
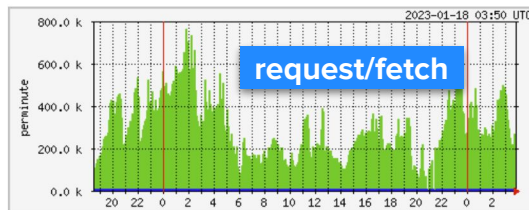
```
vcl 4.1;
import dynamic;
import directors;
{{- range $nindex, $be := .Values.backends }}
backend {{ $be.name }} {
    .host = "{{ $be.host }}";
    .port = "{{ $be.port }}";
}
{{- end }}
acl local {
    {{.Values.acl | nindent 4 }}
}
sub vcl recv {
    if (!(client.ip ~ local)) { return (synth(405));}
    if (req.method != "GET" && req.method != "HEAD") {
        return (pipe);
    }
    {{- range $nindex, $be := .Values.backends }}
    if (req.restarts == {{ $nindex }}) {
        set req.backend_hint = {{ $be.name }};
    }
    {{- end }}
}
sub vcl backend fetch { unset bereq.http.host; }
sub vcl backend response {
    if (beresp.status == 404 ) {
        {{ $lb := last .Values.backends }}
        if (bereq.backend != {{ get $lb "name" }}) {
            set beresp.uncacheable = true;
            return (deliver);
        } else {
            set beresp.ttl = 180s;
        }
    }
}
sub vcl deliver {
    if (resp.status == 404) {
        if (obj.uncacheable){ return(restart);}
    }
}
```

Varnish deployment

- Created two SLATE applications (<https://portal.slateci.io/applications>):
 - v4a - Varnish configured to serve Frontier requests
 - v4cvmfs - Varnish configured to serve CVFMS accesses
- Unlike XCache, straightforward Helm charts. Basically only one configmap, one deployment and one ingress.
- Both v4a and v4cvmfs currently **in production** at two US ATLAS Tier-2 centers: MWT2 (UC, IU, UIUC) and AGLT2
- For more than one year we saw **no issues of any kind**.
- Added to OSG Topology, configured in CRIC (v4a), configured on worker nodes (v4cvmfs).

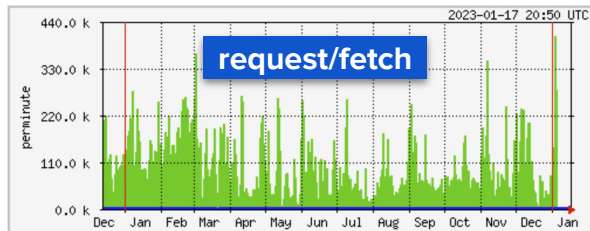
Performance Varnish - SNMP

- Both Varnish and Squid are monitored in both Elasticsearch and [ATLAS MRTG monitoring \(cern.ch\)](#).
- Reports request/fetch, I/O data rate, CPU usage, objects & file descriptors.
- Response times can't be compared as Squid rounds them to 0 seconds.

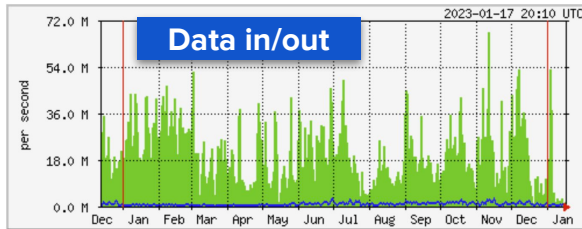


Varnish for Frontier node.
File descriptors is 0 since it doesn't use disk storage.

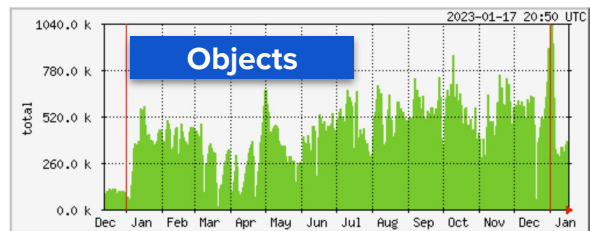
Performance Squid - SNMP



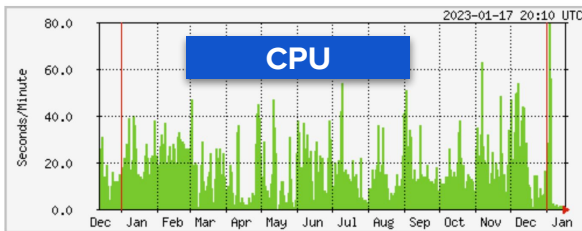
	Max	Average	Current
HTTP reqs	406.7 kreq/min	94.5 kreq/min	890.0 req/min
HTTP fetches	1715.0 req/min	700.0 req/min	172.0 req/min



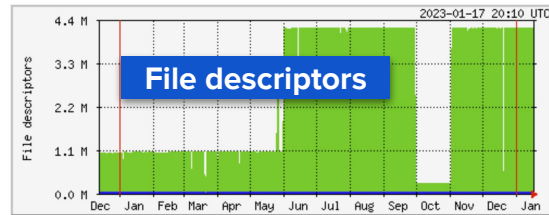
	Max	Average	Current
Total	67.1 MB/s	18.8 MB/s	1534.0 kB/s
Fetches	2866.0 kB/s	733.0 kB/s	368.0 kB/s



	Max	Average	Current
Obj num	1034.3 kobj	412.3 kobj	378.0 kobj



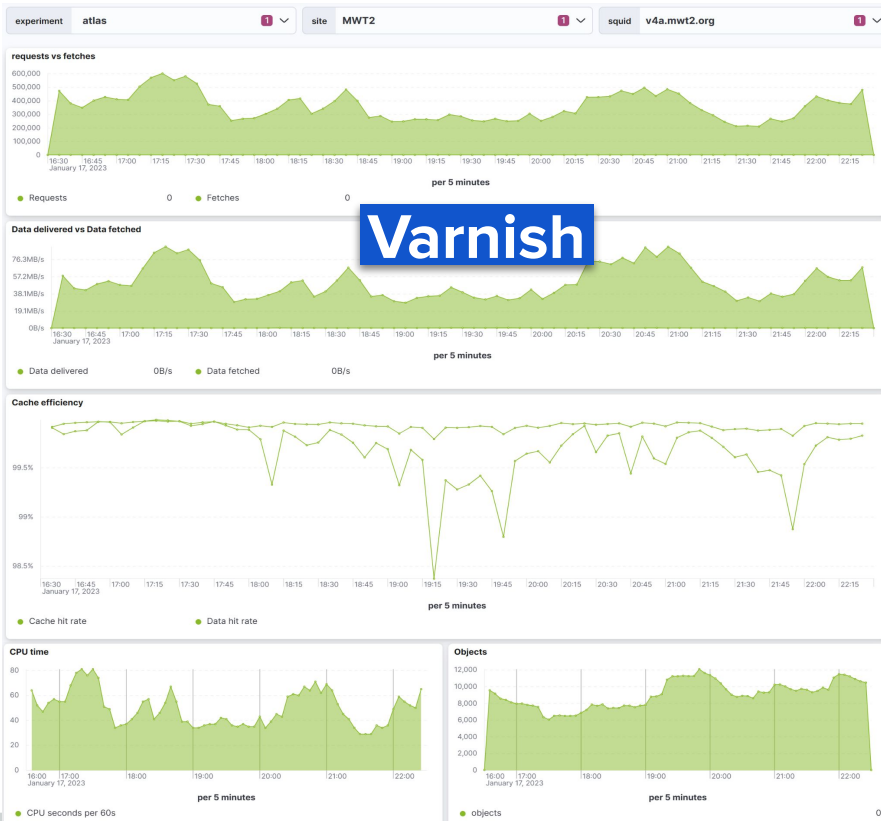
	Max	Average	Current
Cpu time	79 sec/min	17 sec/min	1 sec/min



	Max	Average	Current
Available	4193.6 kdescriptors	2485.2 kdescriptors	4193.5 kdescriptors
Used	7258.0 descriptors	2483.0 descriptors	767.0 descriptors

One of the Squid nodes.
Serving both Frontier and Squid.

Performance in Elasticsearch



Testing it - CVMFS

Squid was completely empty.

x6 faster!

Varnish was under regular production load.

```
Transactions:      101391 hits
Availability:      100.00 %
Elapsed time:      235.63 secs
Data transferred:  7059.98 MB
Response time:     0.05 secs
Transaction rate:  430.30 trans/sec
Throughput:        29.96 MB/sec
Concurrency:       22.02
Successful transactions: 93525
Failed transactions:      0
Longest transaction:  3.37
Shortest transaction: 0.03
```

```
Transactions:      101391 hits
Availability:      100.00 %
Elapsed time:      42.66 secs
Data transferred:  6894.09 MB
Response time:     0.01 secs
Transaction rate:  2376.72 trans/sec
Throughput:        161.61 MB/sec
Concurrency:       16.04
Successful transactions: 96796
Failed transactions:      0
Longest transaction:  4.01
Shortest transaction: 0.00
```


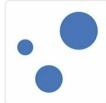
Conclusions

- Thanks to SLATE it is easy to test/deploy new caching servers in production and at scale.
 - Simple to prepare an application, very simple app deployment/management, monitoring
- XCache is more stable, performant and up-to-date when deployed in Federated way.
- Varnish is definitely faster than Squid, needs less resources, it is easier to monitor. **Now in production at two US ATLAS Tier2s: MWT2 & AGLT2**
- Will be adding more applications to test physics data HTTP proxy caching: Nginx, [Apache Traffic Server \(ATS\)](#), Nuster

Extras

Squid vs Varnish | What are the differences?

(stackshare.io)

					
Squid			Varnish		
+ Follow			+ Follow		
+ I use this			+ I use this		
Stacks	Followers	Votes	Stacks	Followers	Votes
96	192	17	12K	2.3K	370

Squid: *A caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. Squid reduces bandwidth and improves response times by caching and reusing frequently-requested web pages. Squid has extensive access controls and makes a great server accelerator. It runs on most available operating systems, including Windows and is licensed under the GNU GPL;*

Varnish: *High-performance HTTP accelerator. Varnish Cache is a web application accelerator also known as a caching HTTP reverse proxy. You install it in front of any server that speaks HTTP and configure it to cache the contents. Varnish Cache is really, really fast. It typically speeds up delivery with a factor of 300 - 1000x, depending on your architecture.*

Testing it - Frontier

Harder to test (but will be done).

Squid was completely empty.

**Huge
difference!**

Varnish was under regular production load.

```
Transactions:      177595 hits
Availability:      99.96 %
Elapsed time:      1612.89 secs
Data transferred:  1232.78 MB
Response time:     0.26 secs
Transaction rate:  110.11 trans/sec
Throughput:        0.76 MB/sec
Concurrency:       28.34
Successful transactions: 177595
Failed transactions: 75
Longest transaction: 24.38
Shortest transaction: 0.21
```

```
Transactions:      177602 hits
Availability:      99.96 %
Elapsed time:      37.07 secs
Data transferred:  1232.79 MB
Response time:     0.00 secs
Transaction rate:  4790.99 trans/sec
Throughput:        33.26 MB/sec
Concurrency:       16.55
Successful transactions: 177602
Failed transactions: 68
Longest transaction: 4.97
Shortest transaction: 0.00
```