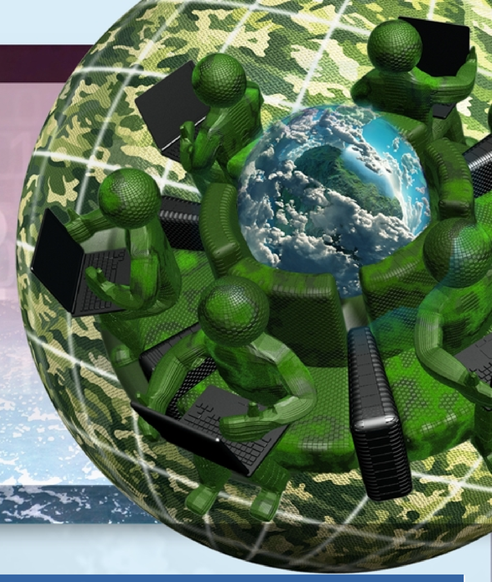
Enhancing the ESGF data node to load balance a distributed cluster of THREDDS instances

Ezequiel Cimadevilla, Pablo Celaya, Antonio S. Cofiño

Santander Meteorology Group, Dep. of Applied Mathematics and Computer Science (Univ. Cantabria), 39005 Santander, Spain

http://meteo.unican.es

Contact: antonio.cofino@unican.es



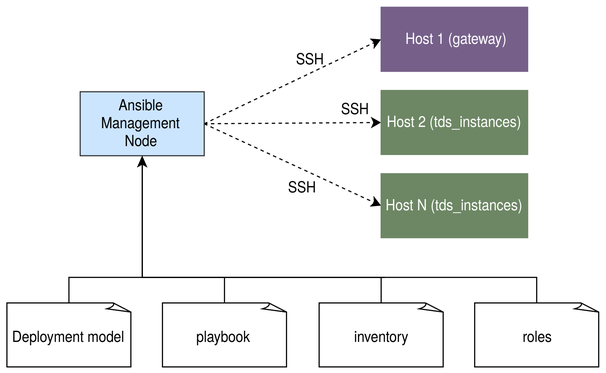
Introduction

Currently, the ESGF-node includes a gateway to the services running in the ESGF-node. One of these services is the TDS web application which runs sharing existing resources on the host. The ESGF-node design only considers one TDS instance running besides to the rest of ESGF-node services. Also, this TDS instance deploys the complete catalog hierarchy automatically generated by the *esg-publisher*, which can become difficult to maintain and to scale if lots of catalogs are generated.

In this contribution, we show a way of deploying a load balanced and automatic provisioned cluster of TDS instances. The definition of the desired infrastructure is declared in a YAML file, which uses Ansible roles and playbooks, that will automatically deploy the cluster of TDS instances and catalogs. The TDS Cluster that conforms this infrastructure is composed of a gateway, the ESGF Data node and the TDS instances.

Ansible - Automatic configuration management and deployment

Ansible variables, roles and playbooks are used for automatic deployment and configuration of the TDS Cluster. The desired infrastructure is defined through a set of Ansible variables called the "**Deployment Model**". Ansible roles are divided into "installation" and "configuration" roles.



Installation roles are in charge of deploying the software dependencies required for the infrastructure to run. They make use of variables to allow administrators to customize the installation and also to provide information to the configuration roles.

Configuration roles gather information from the deployment model and perform the configuration in the software dependencies installed by the installation roles, such as registering the TDS instances in the gateway, which corresponds to the ESGF Data Node in this particular deployment.

The Deployment Model

The definition of the desired infrastructure is declared in a YAML file for Ansible (**Infrastructure as Code, IaC**), which uses roles and playbooks, that will automatically deploy the cluster of TDS instances and catalogs. This definition of the deployment infrastructure follows the TDS Deployment Model, which is composed by **Collections**, **Replicas** and **Instances** deployed in Hosts conforming Clusters.

Collections are hierarchies of THREDDS catalogs that can be deployed to a regular TDS instance on its own. TDS instances are Apache Tomcat server instances, accessed from the outside through a gateway (i.e. reverse proxy), running the TDS web application in a load balanced way. We refer to every publication of the collection in the TDS instances as a **replica**. When a **replica** is copied into an instance, a *catalogRef* tag is added to the instance's root catalog, which references the root catalog of the collection. Through this tag, the TDS builds the catalog tree and it makes the datasets accesible to the gateway.

For the ESGF deployment, collections would correspond to subsets of the catalogs generated by the ESGF publication workflow (esgpublish). Through the use of replicas, groups of hosts will pick up the collections they want to replicate, allowing large hierarchies of catalogs to be **partitioned** and **distributed** between multiple clusters.

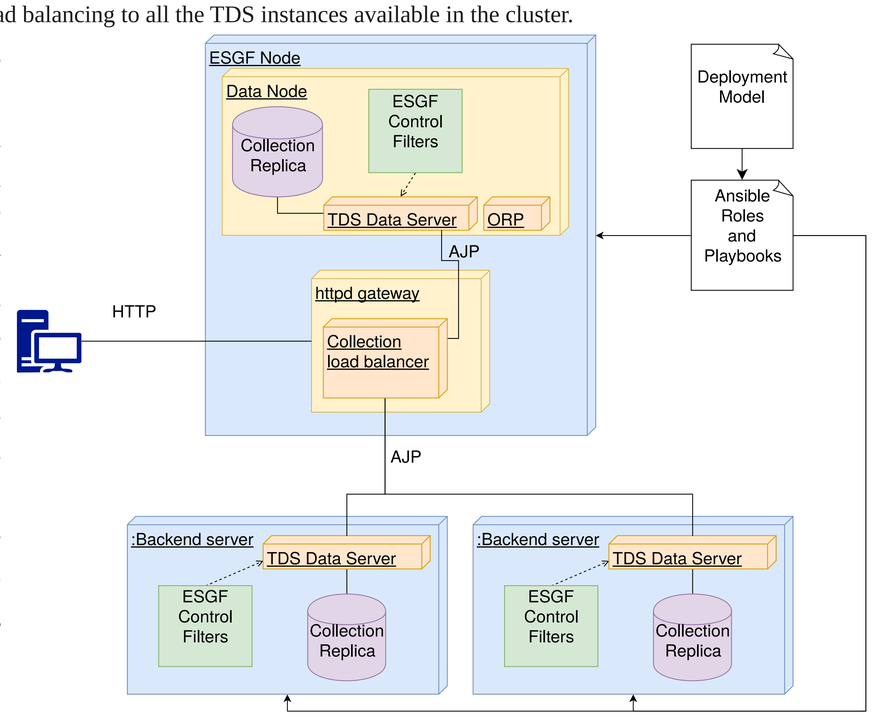
```
collections:
  - &esgcet
    path: esgcet
    catalogs: /esg/content/thredds/esgcet
    services: [catalog, fileServer, dodsC]

instances:
  - &instance1
    name: instance1
    connectors:
      - protocol: HTTP/1.1
        port: 8080
      - protocol: AJP/1.3
        port: 18009

replicas:
  - gateway: esgf-node
    tds_instance: *instance1
    collection: *esgcet
    port: 18009
```

The extended ESGF Data Node

The deployment of this infrastructure requires to change some configuration directives in the default configuration of a ESGF Data Node. We must override the ProxyPass directives in the httpd configuration, in order to embed the directives that allow mod_proxy to perform the load balancing to all the TDS instances available in the cluster.

We also have to configure the ESGF Control Filters in the TDS running in the backend servers, in order to redirect the authentication to the ORP running in the ESGF Data Node, working as the gateway of the TDS cluster. User session persistence is achieved through the use of sticky sessions, which means that once a user gets a session, he or she will be routed to the same TDS instance in future requests.

Requests for datasets will be distributed into the TDS instances running in the backend, avoiding disruptions of the service.



HTTP session clustering

The TDS keeps user sessions through HTTP cookies and the TDS instances that conform the cluster must be aware of them. We have considered the following strategies to maintain user sessions: **sticky sessions**, **tomcat cluster** and **memcached manager**.

- **Sticky sessions** ensure that requests for an existing user session are forwarded always to the same backend instance. However, a TDS instance that goes down losses the sessions of its users. It is recommended to replicate sessions using the tomcat cluster or the memcached manager.
- **Tomcat clustering** enables user replication although it is limited in terms of scalability.
- The **memcached manager** stores user sessions in a separated service, allowing TDS instances to retrieve sessions from this store. This service allow user sessions to persist even when TDS instances become unavailable, since another TDS instance of the cluster can take over it and recover the session from memcached.

Conclusions and future work

- THREDDS **catalog partitioning** is a desirable feature since it allows to split catalogs into smaller and semantic groups.
- Integration with **containers** (ansible-container).
- The gateway is a bottleneck in scenarios with heavy workloads. Alternatives may be evaluated to resolve this problem e.g. **Direct Server Return**.
- Required integration with the **ESGF publication** process. Collections require standalone catalog trees and the publisher actually produces one large tree.
- The gateway is now the single point of failure of the infrastructure, although the TDS service becomes more resilient to failure.

References

- ansible-thredds-cluster, https://github.com/SantanderMetGroup/ansible-thredds-cluster
- JASMIN Conference 2017: Advanced Computing for Environmental Science, Fenández-Tejería, S. Cofiño, A.S. Kershaw, P. Petrie, R. Pryor, M. Stephens, A.
- THREDDS Data Server, https://doi.org/10.5065/D6N014KG
- ESGF, https://esgf.llnl.gov/
- Memcached-session-manager, https://github.com/magro/memcached-session-manager
- Linux Virtual Server, http://www.linuxvirtualserver.org/

This work is performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. IM release: LLNL-POST-XXXXXX