Current ESGF Search Architecture

- Enables local administration of metadata catalogs, yet federation wide searches
- Based on Apache Solr, leverages functionality for distributed searches and replication
- Each node replicates the catalogs of all the other nodes to resolve searches locally
- A client can query any of the nodes in the federation and obtain the same results
Current Shortcomings

- Each node administrator must manually configure a replica shard for all other nodes in the federation.
- High potential for inconsistencies across nodes (for example, if one replica breaks at one node).
- All nodes must scale concurrently when the federation grows (number of index nodes or metadata holdings at each node).
- The current Solr installation is becoming obsolete and insecure, yet it is difficult to upgrade:
  - All sites must upgrade simultaneously for replication to keep working in both directions.
  - Data must be re-indexed to upgrade the underlying Lucene version.
Proposal for Next Generation ESGF Search Architecture

- Let each institution maintain only one index node where they publish their data (i.e. no replica shards)
- Establish a few “super-indexes” that aggregate metadata from all institutions
- Point all client applications to the super-indexes
Technical Implementation

- Adopt Solr Cloud
- Deploy as Docker and Kubernetes
  - On the cloud, or in-premise
- Harvest and sync
Solr Cloud

- Solr Cloud is a more advanced and scalable Solr architecture, designed to be deployed on a multiple hosts.
- The full metadata index is partitioned into logical shards.
- Each shard is physically instantiated as one or more replicas.
- Replicas are automatically deployed onto Solr instances on separate hosts (if possible) for resiliency.
Solr Cloud

- Metadata can be published to any Solr instance and it will directed to the proper shard leader, then replicated (“distributed indexing”)
- Clients can query any Solr instance, and the query will be load balanced and resolved versus a complete set of shard replicas (“distributed querying”)
- A set of Zookeeper servers provides centralized configuration management
Prototype Deployment on AWS

- Small cluster of 3 EC2 instances of type t2.medium (2 CPUs, 4GB memory)
- Solr configuration: 3 shards per collection, 3 replicas per shard
- Tracking ESGF global archive for over 2 months
Docker and Kubernetes

All software components were deployed as Docker containers onto an AWS Kubernetes cluster

- Zookeeper = K8s Deployment
- Solr instances = K8s Stateful Sets
- Harvest/Sync clients = K8s Cron Jobs
Harvesting and Syncing

- Harvesting clients are run initially to read all records from each existing master node into the super-index
  - May take up to several days for large indexes
- Syncing clients are run every hour to sync a remote index to the super-index
  - Algorithm uses timestamp stats to compare the indexes by time interval - year/month/day/hour
The ESGF super-index on AWS
Advantages

- Automatic distributed indexing, querying and load balancing
- Resiliency and automatic failover
- Horizontal and vertical scalability
  - Add more servers and/or increase the memory of each server
  - The system can be scaled by increasing the resources at one location, not at all sites through the federation
- Upgrades can be executed by bootstrapping a new system in the background, and switching over the proxy when ready
Benchmarking: Datasets

- Using “super-index” deployed on small AWS K8s cluster
- 3 EC2 instances of type “t2.medium” - 2 CPUs, 4GiB memory

Solr Benchmarking

Query for Datasets

Query Time (ms)

~1M Datasets
BENCHMARKING: FILES

~18M Files
Conclusions

- Proof of concept successfully executed
- Software stack is ready for operational deployment as beta service
- Need to find resources - on the cloud or in-premise
- A timely deployment is recommended to enable smoother upgrades during CMIP6 operations