ARGO

Availability and reliability monitoring for e-Infrastructures

ARGO Overview

ARGO Framework. Scalable Monitoring

Track Key Performance Indicators

ARGO can monitor a wide range of platforms and provide operational and business insight for a wide range of built-in and user defined key performance indicators.

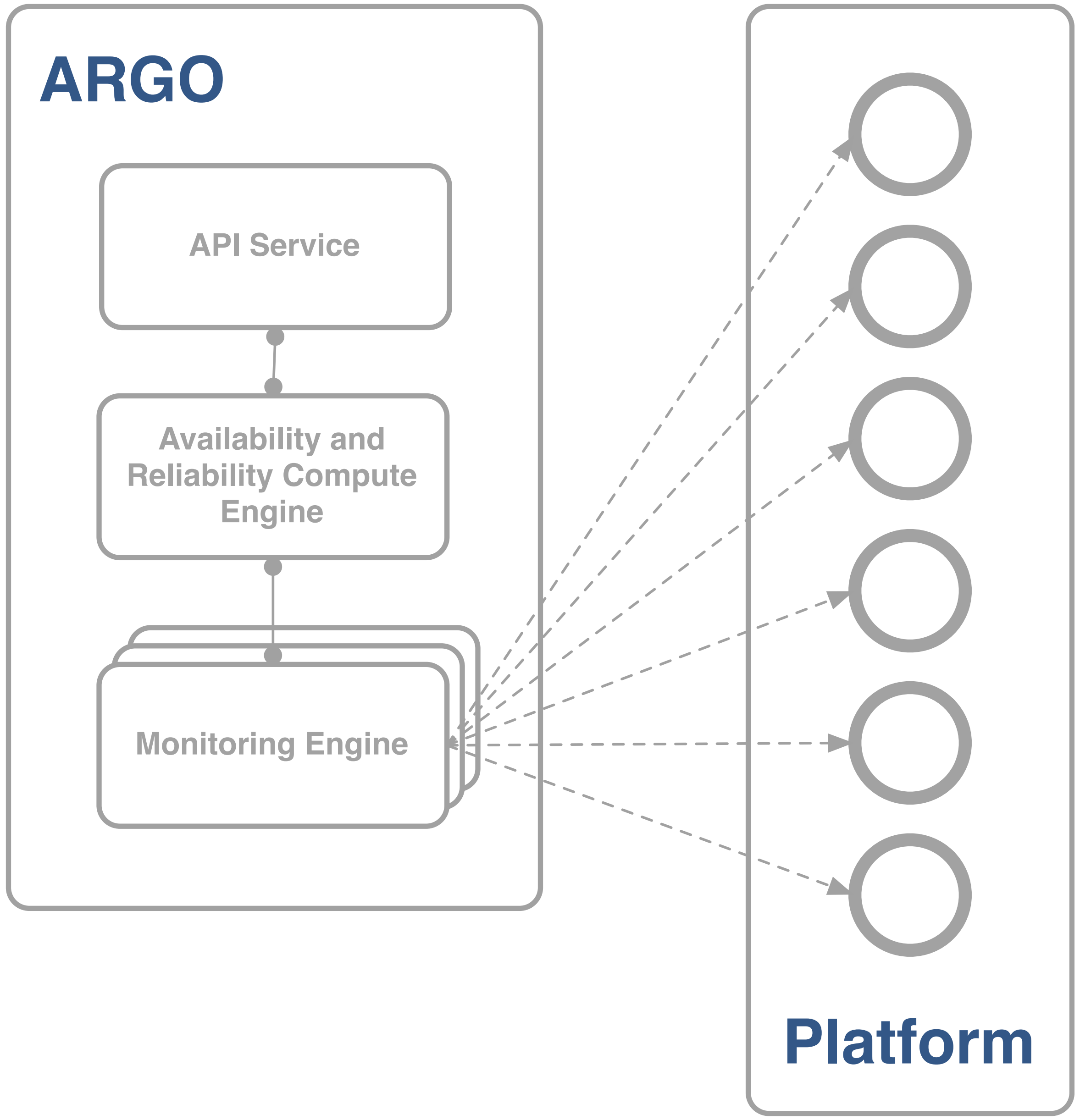
Notification and detailed status reports

Using ARGO operations teams can track the performance and status of each component in the infrastructure, receive notifications and detailed status reports.

Monitor SLAs

Management teams can monitor the availability and reliability of the services from a high level view down to individual system metrics and monitor the conformance of multiple SLAs

ARGO Overview

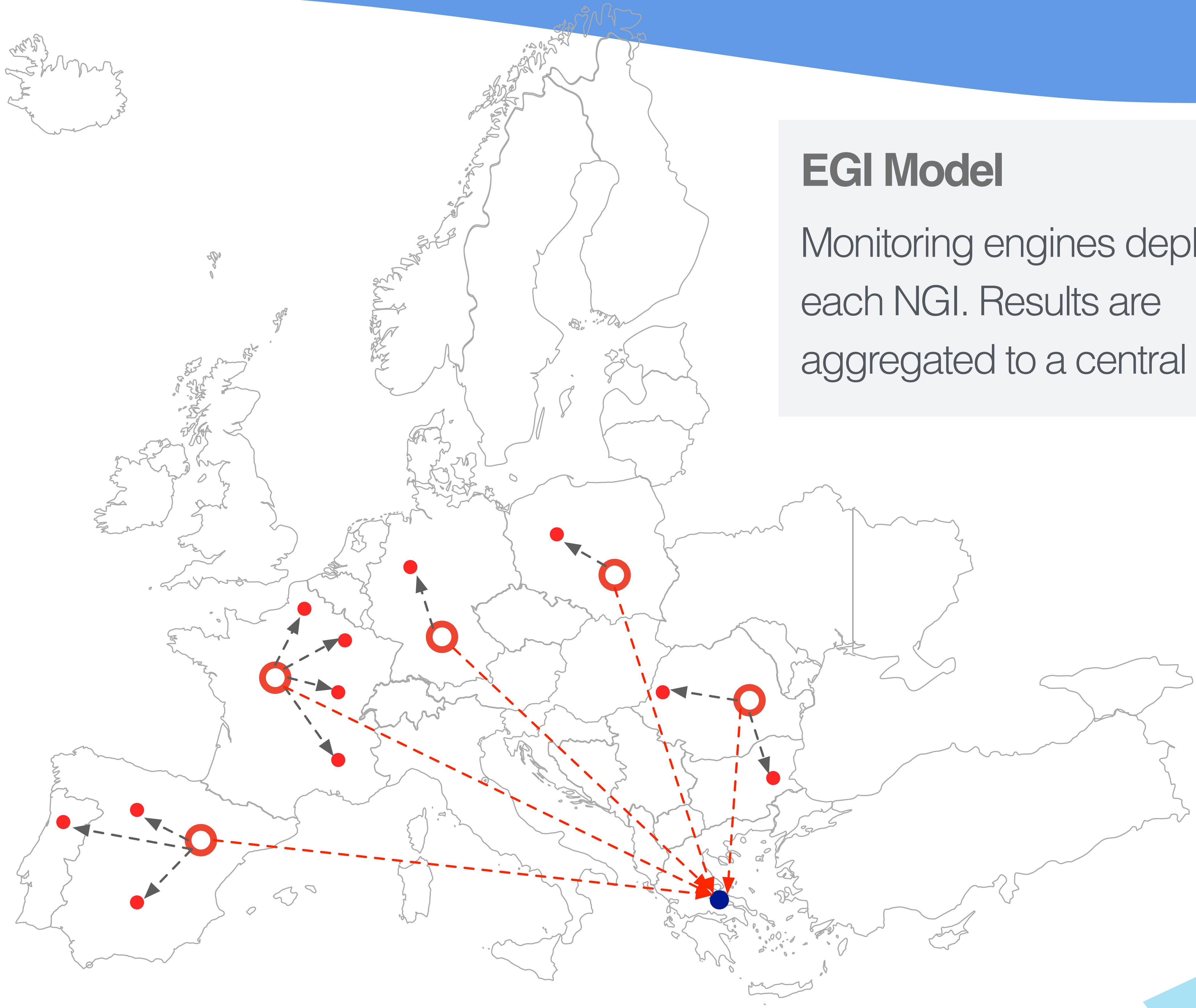


ARGO Components. Modular architecture

At its core, ARGO uses a flexible distributed monitoring engine built around Nagios, a powerful analytics engine and a high performance API service. Embracing a modular, pluggable architecture, ARGO can easily support a wide range of architectures.

Connectors for POEM, GOCDB, BDII and gstat

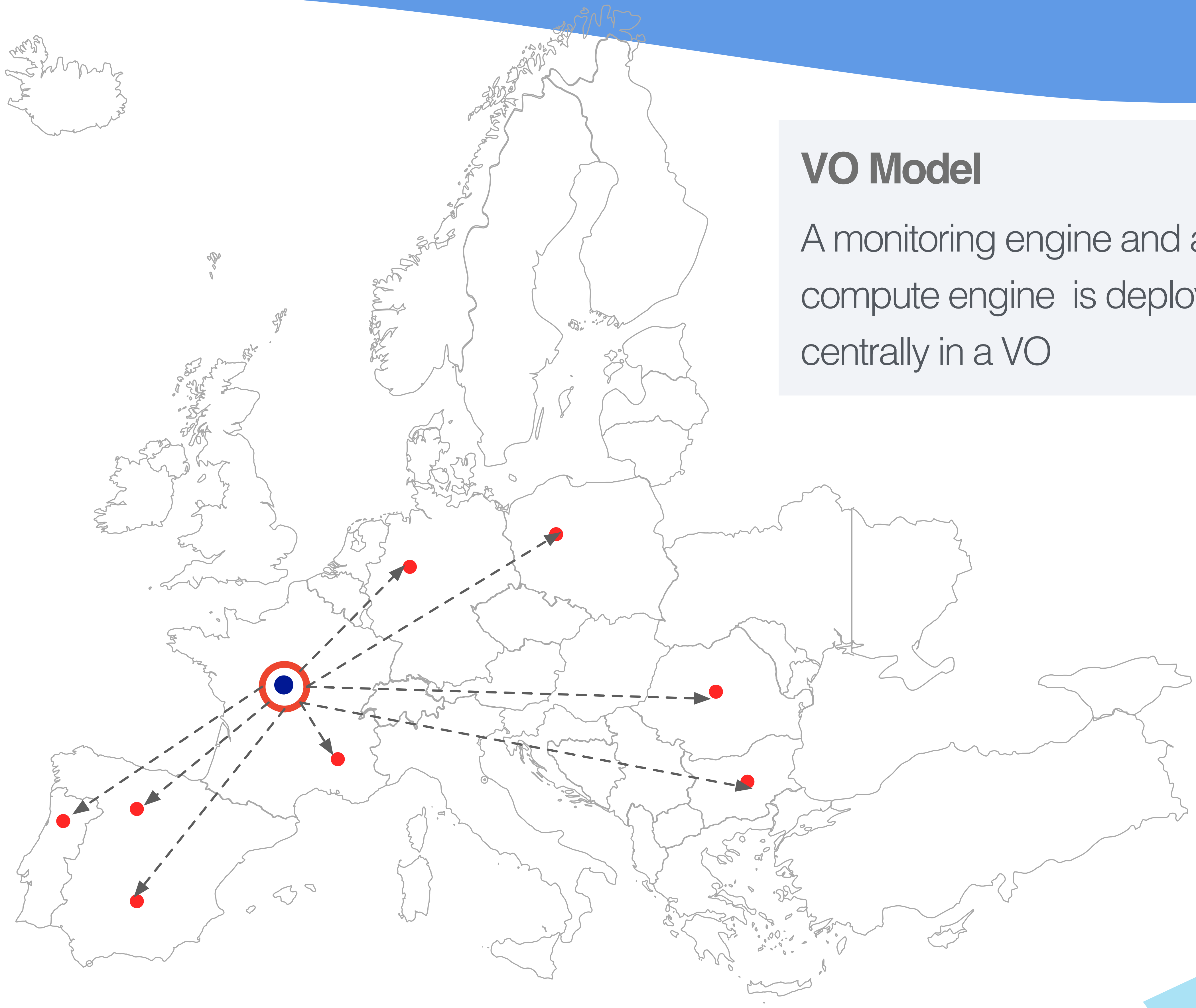
Example deployment models



EGI Model

Monitoring engines deployed at each NGI. Results are aggregated to a central location

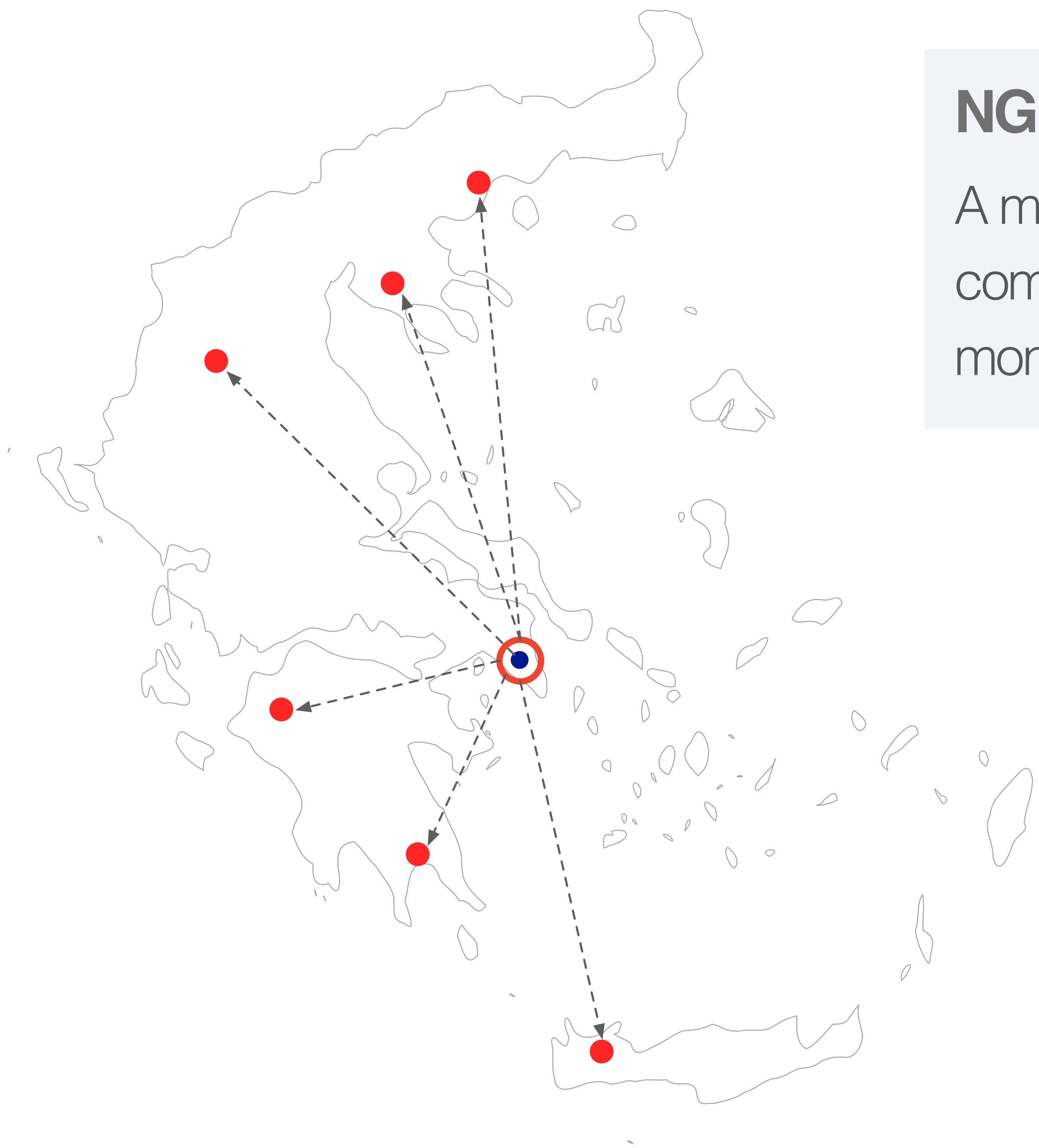
Example deployment models



VO Model

A monitoring engine and a/r compute engine is deployed centrally in a VO

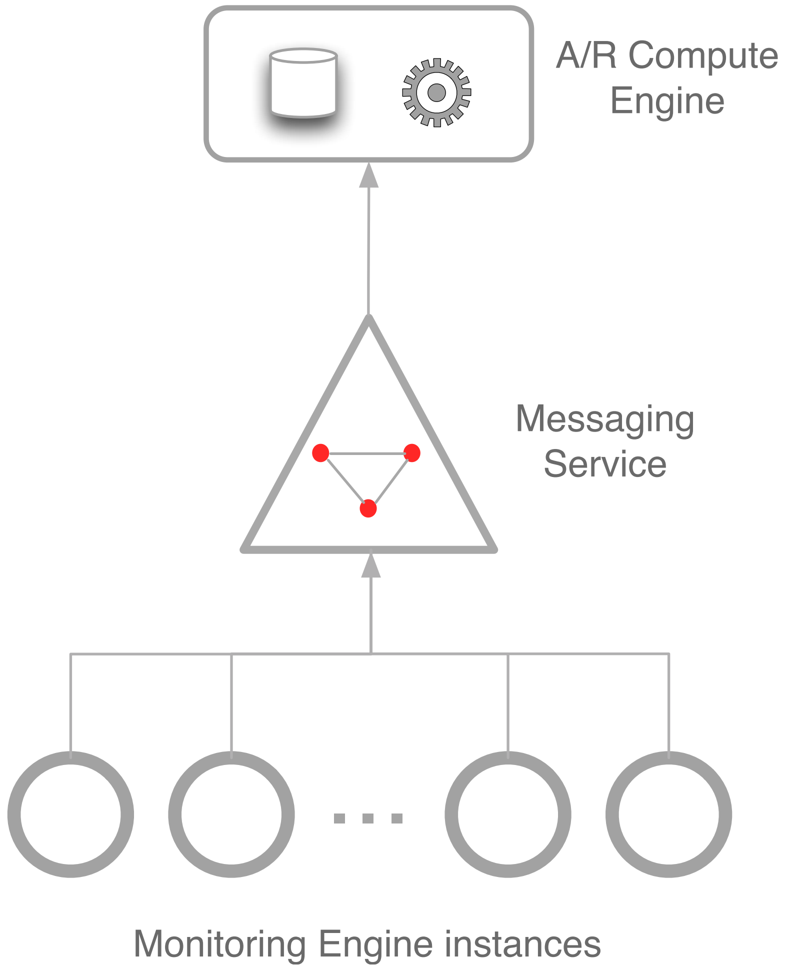
Example deployment models



NGI Model

A monitoring engine and a/r compute engine within an NGI and monitors the local infrastructure

ARGO Overview



ARGO Messaging. Horizontal scaling

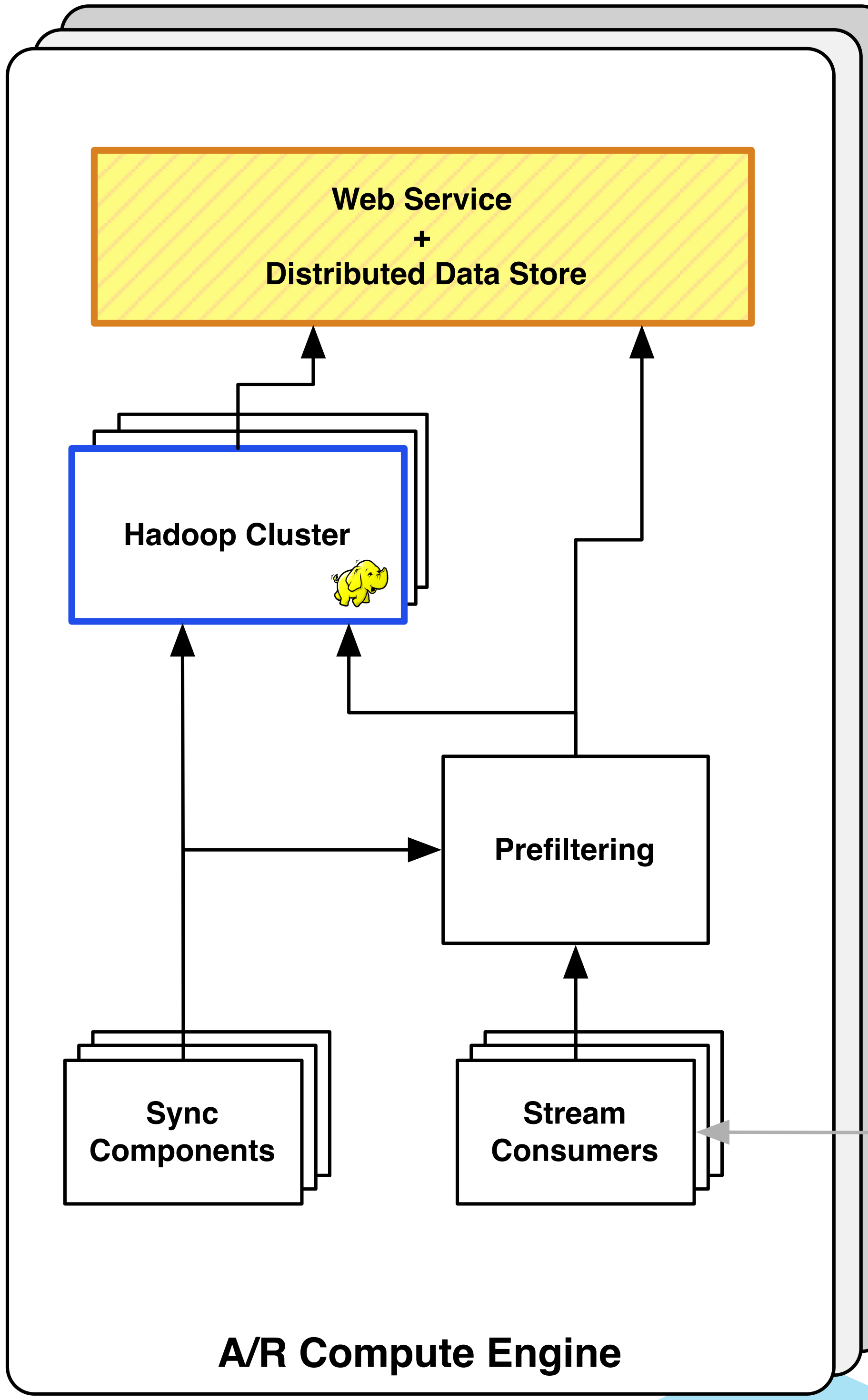
ARGO Monitoring Engine can publish monitoring results in real time to any message broker service, which support the STOMP or AMQP protocol.

Monitoring result collectors can listen on the message broker service, retrieve the results

A/R Compute Engine

Availability and Reliability for distributed services

- **Metric profiles** (which metrics are relevant for a specific type of service)
- **Multiple Topologies** (groups of service endpoints or groups of other groups)
- **Multiple Availability Profiles** (Logical grouping of service end point status results across services groups)
- Definition of **downtimes** (was a service endpoint supposed to be unavailable or not?)
- **Custom factors** (Service Group A can have different importance from SG B)
- **White listing** of monitoring sources
- **Re-computations** on demand



Advantages over ACE

- Based completely on open source components
  - https://github.com/argoeu/
- Can operate in cluster and standalone mode
- Performance scalability & Easiness in day to day operations and debugging
- Can perform batch re-computations in parallel
  - Supports request and approval of re-computation through the API & web interface.
- Removed the limitation of a hard coded availability profile
  - Supports configuration of custom availability profiles through the API & web interface

Advantages over ACE

- Pluggable architecture:
  - Created abstract interfaces for Topology, Downtimes, Metric profiles and factors
  - Sync components are external modules that implement this interfaces and they are responsible for retrieving the required data from the various sources
- Support historical data for topology, downtimes, metrics profiles and factors
  - Use data retrieved from the external sources at the time of the execution of the monitoring probes and not at the time the A/R computations are performed
- Reports are accessible through the web interface. No need to circulate a PDF, although this is still an option

Experience with SAM NGI Nagios

- SAM Monitoring system is deployed in a distributed manner (Over 50 instances registered in GOCDB)
    - Distributed setup was supposed to enable NGIs to add their own custom test and service types and compute A/R
        - …but ACE depended on Oracle and thus it was never added to the SAM Nagios distribution
- The vast majority of the NGIs deploy the stock version of SAM Nagios without any other tests custom configuration

Experience with SAM NGI Nagios

- SAM NGI Nagios maintenance and support requires significant effort
- SAM NGI Nagios upgrades take really long time (in the range of months). Not possible to have more than 1 or 2 upgrades per year
  - Even the addition of new tests or simple modifications of existing tests require focused, lengthy campaigns.
- only a small subset of NGIs deployed failover SAM Nagios instance and therefore failures on instances cause recalculation requests

Experience with Mid/Sec/Ops Mon Nagios

- Three centralized SAM Nagios instances were deployed for the needs of EGI operations
  - midmon - monitoring middleware versions of various services across the whole EGI infrastructure
  - opsmon - monitoring central and NGI operational tools
  - secmon - running various security tests across the whole EGI infrastructure.
- Practice has shown that these instances are performing very well and they resolve some of the drawbacks of distributed approach listed above.
- WLCG experiments are also using central SAM Nagios instances for monitoring all their sites

ARGO Monitoring Engine

- ARGO monitoring engine is refactored SAM Nagios
  - contains Nagios monitoring framework and components needed for communication via messaging infrastructure.
- Heavyweight databases for service status calculation and web interface were removed.
  - Such simplified engine is capable of running more tests and thus monitor larger infrastructure.

ARGO Monitoring Engine

Two possibilities for deployment in EGI-Engage:

1. Use the distributed monitoring model
    - All NGIs will have to upgrade their monitoring instances to the new ARGO Monitoring Engine
    - A roll-out campaign will start in Q1 of of EGI-Engage
2. Move to a centralized monitoring model

A Central ARGO Monitoring Service

- The Central ARGO Monitoring Engine:
  - will run in high availability mode centrally
  - if more capacity is required, new Nagios instances can be deployed in no time
- Will allow us to focus our effort on the operation and maintenance of the Monitoring Infrastructure
  - instead of spending the effort on the coordination of the upgrades
  - It will be much easier to debug, identify and fix problem when they occur.
  - No new effort is going to be required, as we are going to use the existing effort that we have for the coordination of the upgrades
- Will lighten the burden on the NGIs, which will be freed from having to operate the NGI Nagios instances.
- The same code base that will run on ARGO Central, will be available to all NGIs, VO to take an install locally

Thank you

egi