The ATLAS Software Installation System v2: a highly available system to install and validate Grid and Cloud sites via Panda

A De Salvo[1], M Kataoka[2], A Sanchez Pineda[3], Y Smirnov[4]
On behalf of the ATLAS Collaboration
[1] INFN sezione di Roma1, Italy
[2] University of Texas at Arlington, US
[3] Università di Napoli "Federico II", Italy
[4] CERN, Switzerland

Abstract. The ATLAS Installation System v2 is the evolution of the original system, used since 2003. The original tool has been completely re-designed in terms of database backend and components, adding support for submission to multiple backends, including the original Workload Management Service (WMS) and the new PanDA modules. The database engine has been changed from plain MySQL to Galera/Percona and the table structure has been optimized to allow a full High-Availability (HA) solution over Wide Area Network. The servlets, running on each frontend, have been also decoupled from local settings, to allow an easy scalability of the system, including the possibility of an HA system with multiple sites. The clients can also be run in multiple copies and in different geographical locations, and take care of sending the installation and validation jobs to the target Grid or Cloud sites. Moreover, the Installation Database is used as source of parameters by the automatic agents running in CVMFS, in order to install the software and distribute it to the sites. The system is in production for ATLAS since 2013, having as main sites in HA the INFN Roma Tier 2 and the CERN Agile Infrastructure. The Light Job Submission Framework for Installation (LJSFi) v2 engine is directly interfacing with PanDA for the Job Management, the Atlas Grid Information System (AGIS) for the site parameter configurations, and CVMFS for both core components and the installation of the software itself. LJSFi2 is also able to use other plugins, and is essentially Virtual Organization (VO) agnostic, so can be directly used and extended to cope with the requirements of any Grid or Cloud enabled VO. In this work we will present the architecture, performance, status and possible evolutions to the system for the LHC Run2 and beyond.

ATL-SOFT-PROC-2015-013
11 May 2015

1. Introduction

The ATLAS [1] Software Installation System v2 is the evolution of the first version of the original system developed for ATLAS starting in 2003. The original infrastructure was based on the *Light Job Submission Framework for Installation* (LJSFi) v1 [2], which was the first production-level system in ATLAS to install, validate and manage the full lifecycle of the software releases in the Grid sites. LJSFi v1 already had the possibility to extend to multiple backends, but it was not multi-threaded nor highly available by design.

The new system, based on a full rewrite of the original framework, LJSFi v2, added both the multi threading features, needed to speed up most of the operations by parallelizing tasks, and the High Availability (HA) properties, needed to build a distributed and robust system. Any bottleneck of the

old system was removed, and the new architecture is able to run both concurrently in the same site, by using multiple instances of the components, and in different sites, by adding classes of services in HA mode. In this way the failure of a single component in a location is not fatal for the whole infrastructure. All these characteristics guarantee both the vertical and horizontal scalability of the infrastructure.

In this paper we will describe the architecture of the new system, the different components with their functionalities and the performance of the overall system when operating on the ATLAS software releases in the sites.

2. Overview of LJSFi v2

LJSFi v2 is a complete rewrite of the original system, developed in ATLAS starting in 2003 as a job submission framework for the validation of software releases and other software installation related tasks.

The framework evolved with time to cope with the increased load, essentially determined by the bigger number of sites and resources available to the ATLAS Virtual Organization (VO).

The original system used the gLite WMS [3] backend to send installation jobs, but its scalability and support was no longer adequate for the needs of ATLAS. The new system introduced the support for Panda [4], the ATLAS Production and Analysis infrastructure, and the HA features, in order to make it usable with the new VO scenario and configuration.

LJSFi uses a plugin architecture since its first version: this guarantees the possibility of adding any other backend in the future, in order to expand the capabilities and target of the system.
At the moment the active backend is only Panda, but the interface with the LCG WMS is still available and operational.

The framework is multi-VO enabled. The same set of servers and services can handle multiple VOs at the same time; alternatively a set of federated services can be added.
The users can interact with the system via a Web Interface. Each user is assigned to a specific role and is identified by means of his personal X509 certificate. Anonymous users have limited access to the information contained in the web pages, while registered users can have the full access. The level of information accessible by each registered user depends on both his role and on the specific privileges assigned. Some of the services of the framework are accessible via X509 authentication only: the authentication layer uses the standard apache and PHP APIs, so that the code is lightweight and portable to multiple platforms without any big intervention.

LJSFi is able to cope with hundreds of resources and thousands of releases, with turnaround of the order of minutes in the submission phase. The turnaround of the full validation cycle of course depends on the sites' availability and readiness, but it is generally of the order of a few hours, where most of the time is spent with the actual testing of the releases at the destination. The fast turnaround is possible due to the high priority of the jobs sent by the Installation System via Panda.

The overall system has good horizontal and vertical scalability. The horizontal scalability is guaranteed by adding classes of components to the system, while the vertical scalability is achieved by adding more powerful machines and increasing the number of threads of the single instances. The HA functionality is ensured by the DB infrastructure and the embedded facilities of the LJSFi components.

3. The LJSFi v2 architecture

The LJSFi v2 framework has 3 main classes of components:

- The LJSFi server
- The Request Agents
- The Installation Agents

All these macro systems have independent subsystems that may be configured in multiple instances for load balancing, high availability or simply to have a better segmentation of the tasks.

In Figure 1 the current architecture and configuration of the infrastructure is shown.

The HA DB servers are distributed between Roma and CERN, while the server, the Request Agents and the Installation Agents are currently hosted only in Roma. The setup of a second server at CERN is in progress, while for the other services it is trivial to install them even in other sites.

The Task Request Agents and the Installation Agents always use the closest servers, from which they are redirected to the appropriate DB machines to be used for the specific instances.

The HA for the server may be granted by adding an HA alias that may redirect to any available server.

The software used to access the Panda and the ATLAS Distributed Data Management (DDM) services is obtained via CVMFS [5], so basically any machine able to use CVMFS can be used as a platform for the LJSFi components.

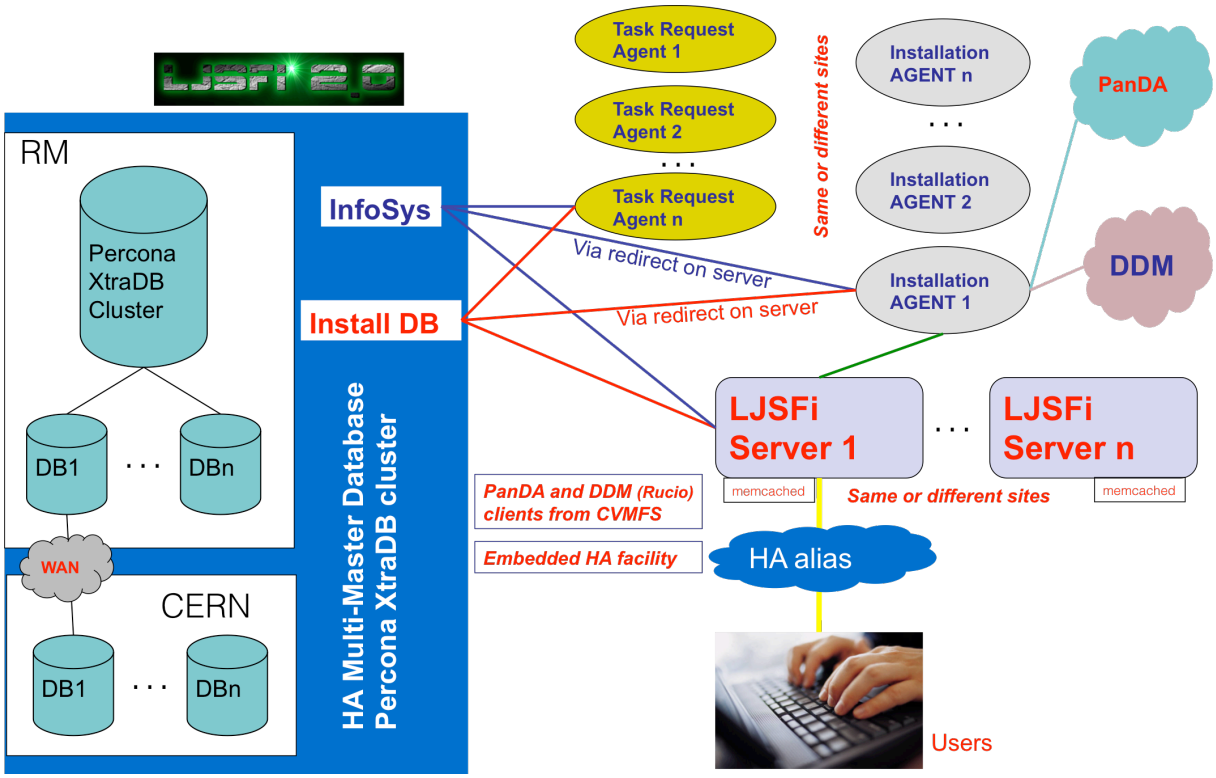

Figure 1: The LJSFi architecture and current configuration. The DB instances are partly hosted in Roma and partly at CERN. The system is ready for the installation of a second server at CERN, while the primary server is kept in Roma.

3.1. The LJSFi server

The server is responsible for the global orchestration of the services, and it is built out of the following components:

- The HA DB
- The Information System (InfoSys)
- The Web Interface
- The monitoring facilities
- The APIs

The servers can be used in single copy mode or in multiple copies for HA purposes. Every server is equal to the others for what concerns the functionality, and the only difference is the presence of

different log files coming from the associated jobs. All the other information is stored in the HA DB facility, so is therefore available coherently at the same time to all the server instances. In this scenario the failure of a server instance just means the unavailability of the log files stored in the failed machine, while all the other functionalities can be taken over by a different server.

3.1.1. The HA DB facility

The LJSFi DB is based on Percona XtraDB cluster [6]. The system is an extension of the MySQL engine, with WSREP (Write-Set Replication) patches. By using the Galera WSREP patches the LJSFi DB is a true multi-master, WAN-enabled engine.

All the commands, transient status and configurations of the Installation System are handled via the DB, in order to avoid single points of failure.

The minimum number of machines to be dedicated to the DB cluster is 3, but to cope with a high load the current configuration provides 9 machines in Roma and 2 at CERN. More machines may be added, even in other sites, for better redundancy.

No powerful machine is needed for the DB, but at least 4GB of RAM, 100GB of HD and standard network connectivity are required. The replication efficiency of WSREP is anyway performing much better on low latency networks, like the one between Roma and CERN.

The DB nodes at CERN are VMs running on the Agile Infrastructure and no performance issue was seen so far, including the WAN latency.

The LJSFi DB is hosting both the Installation DB, and the InfoSys.

The Installation DB is used as the source of release definition for CVMFS and the full driver for the ATLAS installations, while the InfoSys is used for resource discovery and matchmaking.

3.1.2. InfoSys

The InfoSys DB is used by the agents for resource discovery and matchmaking and is connected to AGIS, the ATLAS Grid Information System, and Panda. The site data available from AGIS are currently mirrored every 2 hours: this parameter can be adjusted to balance data matchmaking freshness and performance. If the data contained in the InfoSys are older than twice the refresh interval time window, the system refuses any interaction.

The internal matchmaking may use more parameters than the ones just obtained by AGIS, like for example the OS type/release/version, that are filled by the installation agents via callbacks directly in the InfoSys. These parameters can be sent to AGIS if needed, as is also done for the CVMFS attributes. Sites can also be disabled from the internal matchmaking, if needed: this is extremely important for HPC and opportunistic resources, where an automatic validation is not desirable.

3.1.3. Web Interface

The LJSFi Web Interface has been designed for simplicity and clearness. In order to facilitate the user input, most of the input boxes use hints rather than combo boxes, via JQuery [7] scripts. All pages are written in HTML 5, and have an online help for users.

Links to AGIS and Panda for the output resources are also provided for immediate access to all the debugging activities.

Each server has a separate Web Interface, but the interactions with the system are consistent, whatever server one is using, by means of the LJSFi DB.

3.1.4. Monitoring facilities

The framework offers several options for monitoring the status of the tasks. The main monitoring is displayed via the Web Interface, and gives the realtime status of the installation tasks in the sites.

Statistics on site configurations, facilities and services at the site level are also available through the web pages. Users can also retrieve the monitoring data via the APIs to build external monitoring systems. An example of this is given by the ATLAS SSB, which is able to retrieve information about the CVMFS setup status in the sites from the Installation System servers.

3.1.5. APIs

LJSFi v2 provides two ways to interact with the servers:

- The Python APIs
- The REST APIs

The Python APIs are typically used by the end users via the LJSFi CLI commands. The Installation Agents and Request Agents are also mainly using the Python interface.

The REST interface, more modern and fast, is used in a broader spectrum of activities, ranging from the callback for the jobs to external monitoring agents, CLI commands for the Installation Agents and internal server activities like periodic scheduled tasks.

3.2. The Request Agents

The LJSFi Request Agents are responsible for discovering new software releases and inserting validation requests into the DB. The installation requests are then handled by the Installation Agents.

The Request Agents use the data stored in the InfoSys to feed the matchmaker, discovering all the resources currently set to any status other than offline.

The matchmaker also considers task pre-requirements, for example pre-required releases, OS type, architecture, etc. An important parameter is also the maximum number of allowed concurrent jobs. This is especially important for multicore resources, where a single job uses more job slots than in normal resources.

The Request Agents periodically run on all the releases set in auto deployment mode in the DB.

The Request Agents are not yet multi-threaded, and currently the loop is set every 2 hours, but this value will be shortened as soon as the Request Agents have been ported to multi-threaded mode.

3.3. The Installation Agents

The Installation Agents are used to follow the whole job lifecycle.

They process the Task Requests coming from the database, sending the jobs to the configured backend and storing the status in real-time.

Collision control among multiple agents is handled by central locks on tasks, thus avoiding both multiple requests operated by multiple agents running in parallel and multiple concurrent jobs.

In case the site is tagged as using CVMFS, the given software releases are pre-tagged before sending the jobs, in order to speed up the validation procedures. Since almost all the sites in ATLAS are now instrumented with CVMFS, the pre-tagging technique is used in most of the jobs, except for special resources like the HPCs.

The Installation Agents are also responsible for checking the status of the jobs and retrieving the output, for debugging purposes.

If the target resource is CVMFS equipped, the job payload will detect this automatically and just perform the test of the software in the site. Otherwise, a full installation and validation is performed.

For jobs finishing successfully, the Installation Agents confirm the tags in AGIS. However, in case of failure, the tags are removed.

The installation agents are fully multi-threaded and able to send several jobs in parallel and follow the other operations. To protect against race conditions, timeouts to the operations are provided either from the embedded commands or by the generic timeout facility in the agents themselves.

Several installation agents can run in the same site or in different sites. Each agent is linked to an LJSFi server, but when using an HA alias it can be fully delocalized. Each server then redirects the DB calls via haproxy to the closest DB machines, taking advantage of the WAN Multi-Master HA properties of the DB cluster.

Currently the Installation Agents are serving all the ATLAS grids (LCG/EGI, NorduGrid, OSG), the Cloud resources, the HPCs and opportunistic facilities via Panda.

The logfile of every job is kept for about a month in the servers, for debugging purposes: each server knows where the logfiles are and can redirect every local request to the appropriate machine.

4. Performance

The system can scale up to more than several thousand jobs per day. In Figure 2 a typical example of the daily installation activities is shown. In the current configuration, with 5 parallel Installation Agents and 2 Request Agents with a single server, the system can reach up to 30000 activities per day. The submission rate is more than 12000 jobs/day (Figure 3). Adding more agents would raise this limit. The server and the DB backend are in fact not saturated by this rate. In case of saturation on the server or the DB infrastructure, it is also possible to add more hosts of that class.

To improve the performance, a limit on the number of jobs handled by the currently running agents has been set to 4000. This limit actually depends on the machine type the agents are running on, thus it must be adjusted on an ad-hoc basis.

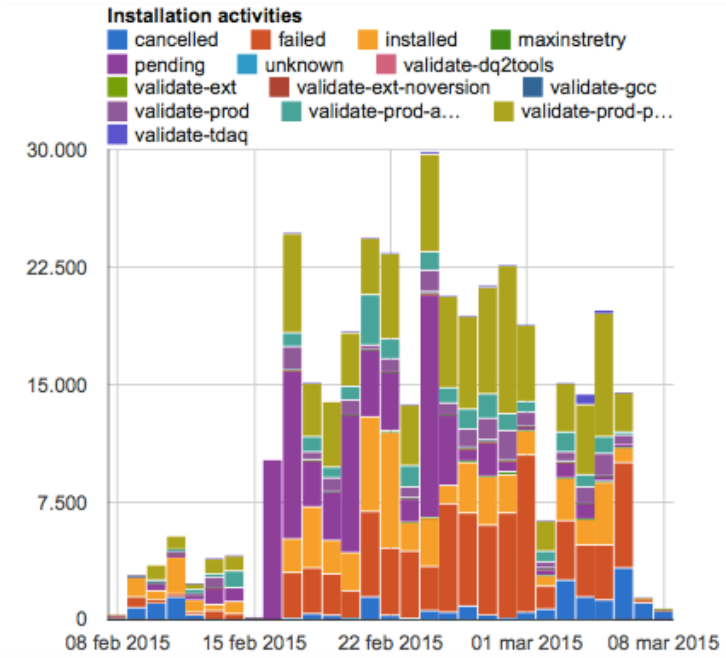

Figure 2: The installation activities of LJSFi. The systems scales up to 30000 activities per day, in the current configuration. Adding more agents would raise this limit.

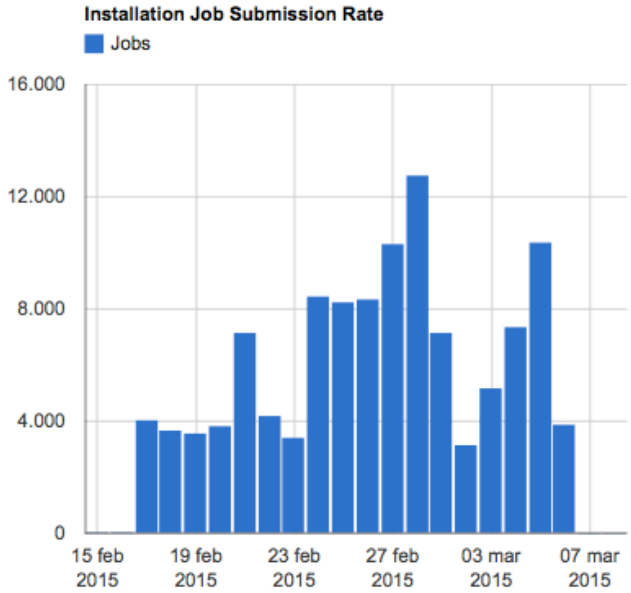

Figure 3: the Installation Job submission rate. In the current configuration the system is able to send more than 12000 jobs per day.

The system processes new requests before the others, to allow a fast turnaround of urgent tasks: generally only a few minutes are needed between the task requests and the actual job submission by the agents. In Figure 4 the number of requests and the number of running jobs in a peak period are displayed, showing that LJSFi is able to cope very quickly with more than 20000 requests and track the lifecycle of the corresponding validation jobs.

The ATLAS Installation System v2 is able to handle a large number of releases and sites: we currently have more than 500 different resources and more than 1600 software releases or patches handled by the system, but the infrastructure is not saturated and it is also easily expandable.

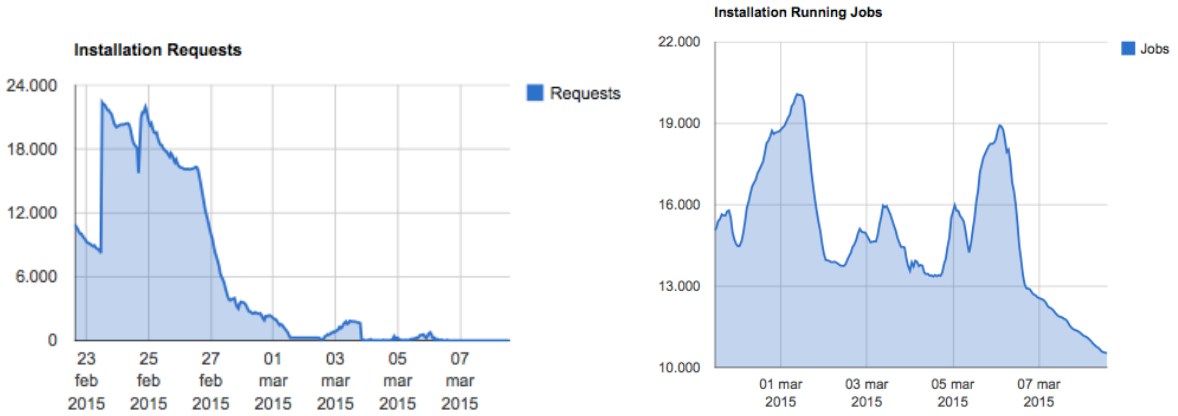

Figure 4: Number of installation requests (left plot) and running jobs (right plot) handled by the system. LJSFi is able to cope very quickly big more than 20000 requests and track the lifecycle of the corresponding validation jobs.

5. Conclusions

The ATLAS Installation System v2, based on LJSFi v2, has been shown to be a powerful and robust solution for the installation of the experiment software and validation in the Grid, Cloud and HPC sites via Panda. The HA properties of the infrastructure, the robustness of the solution and the scalability are properly fulfilling the requirements of a big Virtual Organization like ATLAS. The system is also multi-VO enabled, so that other communities may adopt it.

6. References

[1] ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider, *JINST 3 S0800 3doi:10.1088/1748-0221/3/08/S08003*

[2] De Salvo A, Barchiesi A, Gnanvo K, Gwilliam C, Kennedy J, Krobath G, Olszewski A, Rybkine G 2008 The ATLAS Software Installation System for LCG/EGEE *Journal of Physics: Conference Series* **119** (2008) 052013

[3] Cecchi M et al. 2010 The gLite Workload Management System *Journal of Physics: Conference Series* **219** (2010) 062039

[4] Maeno T 2008 PanDA: distributed production and distributed analysis system for ATLAS *Journal of Physics: Conference Series* **119** (2008) 062036

[5] De Salvo A 2012 Software installation and condition data distribution via CernVM File System in ATLAS *Journal of Physics: Conference Series* **396** (2012) 032030

[6] http://www.percona.com/software/percona-xtradb-cluster

[7] https://jquery.com/