EPICO - eLab Procedure for Installation and Configuration

Moreno Baricevic
CNR-IOM DEMOCRITOS
Trieste, Italy
baro@democritos.it

Abstract

*The need to support new typologies of nodes in a HPC clustering environment extremely heterogeneous and rapidly growing, quickly integrating new technologies and computational solutions, dictated the development of a software flexible enough to handle such complexity. At eLab, in Italy, we developed EPICO, the **e**Lab **P**rocedure for **I**nstallation and **CO**nfiguration, a framework that aims to satisfy this requirement.*

Keywords EPICO, LINUX Clusters, HPC, High-Performance Computing, Cluster Deployment, Distributed Installation Procedures, Unattended Installation.

1. Introduction

In the continuously evolving world of HPC, a challenge that often arises is the integration of new technologies and hardware solutions in a production environment. Sometimes, this involves just the addition of new computing nodes identical to the ones already installed, but in a research context characterized by occasional and limited funds, this problem worsen when purchases are spread on a long period and the new hardware is often one or more generations advanced in respect of the one already in production, or that might carry different interconnection technologies on-board or incompatible solutions with the previous hardware.

250 nodes, 1300 cores, 20 different profiles based on logical typology and hardware requirements. This was the most important challenge we had to face at eLab, in Italy. Several small funds spread on many years led to a cluster that integrated together more than 20 different hardware and software profiles. Compute nodes with ide, scsi, sata, sas hard drives; disk-less nodes that rely on external storage; single and multi-core machines, AMD and INTEL processors; Gigabit, Myrinet, Infiniband interconnections; sometimes bleeding-edge hardware not yet supported by the kernel or LINUX distribution used on the other nodes.

The flexibility required to handle such heterogeneous hardware, is not always satisfied by already available installation and management software. Facing this problem, we decided to develop our own software solution. In this short paper, I will describe the evolution of EPICO, a software that aims to satisfy this requirement.

2. What is EPICO

EPICO, **e**Lab **P**rocedure for **I**nstallation and **CO**nfiguration, is a framework for unattended and distributed deployments of LINUX, focused on the post-installation and post-configuration for heterogeneous HPC clusters. It's made up of a collection of procedures, scripts and strategies, built brick-by-brick, whenever new hardware was introduced, in more than 10 years of on-the-field experience.

Fruit of the experience and requirements on extremely heterogeneous clusters ($>$250 nodes and $\sim$20 HW/SW profiles at eLab), it has been designed to be extremely flexible and customizable, able to handle different and independent installation profiles based on logical typologies (master node, I/O server, computing node) or by single hosts, and suitable for unattended deployment of a single ad-hoc machine as well as large heterogeneous clusters, making a node ready to enter the production environment and be operative at the first reboot after the installation, whatever its purpose (master/storage/compute node).

EPICO is based on open standards, well-known protocols, widely used open/free tools and standard procedures. As the tools and standards it depends on, EPICO is Open and Free (as in free beer and as in freedom), unconditionally modifiable to expand the customization possibilities by adding new scripts and integrating new procedures, allowing to rapidly integrate new resources into the production environment without much effort.

Flexible and customizable, as well as complex, this software is aimed at experienced LINUX system administrators, or skilled users with scripting experience and some knowledge about the services involved (PXE, DHCP, DNS,

NFS, RPM-based package repositories, queue systems). It was based and tested on RPM-based LINUX distributions using the Anaconda/Kickstart installer[1] (Red Hat[2], CentOS[3], Fedora[4], White Box[5], ...), even though scripts and procedures should work with other distributed installers too (e.g. FAI[6] for Debian[7] / Ubuntu[8]).

3. Evolution

Initially, we decided just to set up a network booting environment based on standard features and services like PXE/DHCP/TFTP and providing a repository for the packages via NFS. Different Kickstart configuration files were just supposed to setup a basic installation of Red Hat or any derivative distribution like White Box, Fedora and CentOS. Each different kind of machine was handled providing different Kickstart files.

The post-configuration issue, a set of procedures needed to integrate a freshly installed machine into the cluster in order to make it immediately available for production, was solved using the post-configuration section of the Kickstart file (`%post`). As soon as the number and the entity of the customizations grown up, the problem of maintaining and keeping consistent many copies of, in principle, the same file, had to be solved. We decided to move the `%pre` and `%post` sections outside the Kickstart file, in order to keep the customization script (initially a single bash script) as a separate file to be included by the Kickstart files. The Kickstart files were still providing the instructions for the hard-disk partitioning, not yet handled externally, and the configuration differences that defined the software/logical profiles (master node, storage nodes, computing nodes, ...).

Still, some nodes needed different instructions, depending on the available hardware, so a single script was not yet the solution. We made the scripts aware of the hardware, using `lspci`, for instance, and thus providing conditional statements for the execution of portions of the script. This was not yet enough, though, as maintaining a single script with everything and lots of conditional statements was getting troublesome. The next obvious step was to split this huge script in several tasks, basically one for each service, feature, piece of hardware or even just debug and test. A task-list, specific for each profiles was defining and loading the task scripts required for the specific configuration of a node, a subset or a pool of nodes, or a profile.

At this point we had the need to centrally separate and manage profiles, pools, subset and single nodes. Some of the solutions adopted involved the subnetting of the network and the definition of meaningful host names, in order to define subsets of nodes based on common characteristics identified by the IP or the name. The TXT entry of the DNS was exploited to supply some information too. At some point we also adopted the kernel cmdline as a vector for information parsed out and used during the installation.

All this led to a collection of scripts and procedures, built brick-by-brick in 10 years of experience in an extremely heterogeneous cluster environment, that we called EPICO, **e**Lab **P**rocedure for **I**nstallation and **CO**nfiguration.

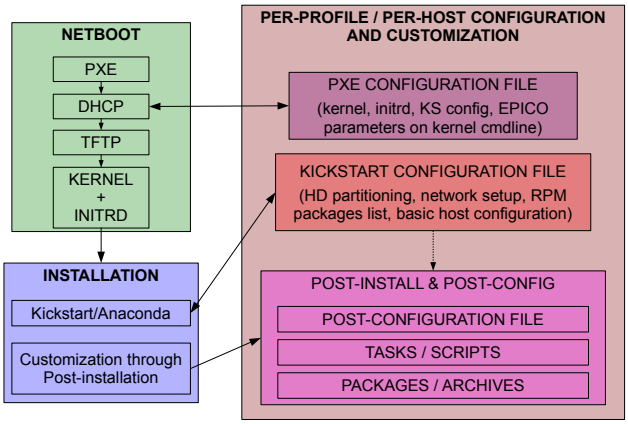

Figure 1. EPICO: installation process

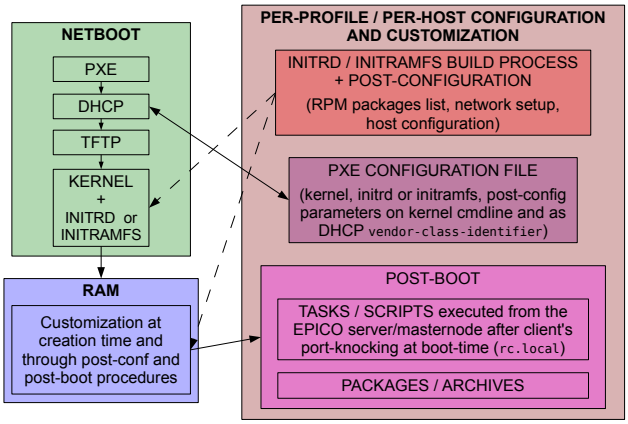

Figure 2. EPICO: ramdisk/ramfs for disk-less nodes, rescue and HW test

4. Deployment and usage

EPICO has been used for the deployment of 15 HPC and GRID clusters (production level), for a total of about 900 nodes, as well as for "hands-on" laboratories, HPC/GRID schools and training courses aimed at technical and scientific personnel.

Among the organizations that are using EPICO in production environments:

- >250 nodes with 20 HW/SW profiles at SISSA / CNR-IOM DEMOCRITOS, Trieste - ITALY
- >100 nodes with 6 HW/SW profiles at ICMS, TEMPLE University, Philadelphia - USA
- >50 nodes with 5 HW/SW profiles on a GRID/HPC cluster at MERCURIO, Udine - ITALY
- 2 HPC clusters at ADDIS ABABA UNIVERSITY, Addis Ababa - ETHIOPIA
- 1 HPC cluster at MS2, SPIN S.r.l., Trieste - ITALY
- >50 nodes on 2 GRID clusters at eLab, Trieste - ITALY

Currently, an EPICO server can be deployed by mean of a USB key, on which EPICO must be downloaded (see section 6) and configured, and then providing a boot loader and a repository for the packages. The more convenient way to supply the latter, is to use a bootable installation CD/DVD of a Red Hat based distribution, which provides both the boot loader and the repository. An external repository can be used as well (network-based: NFS/HTTP/FTP; or on local storage as a an HD or USB key), but in this case the boot loader must be provided through an installation media (CD/DVD-ROM) or PXE boot. Of course, if an EPICO server is already available, it provides the PXE boot, the EPICO framework itself over NFS, and an NFS-based repository, out-of-the-box. Moreover, EPICO can be deployed on any LINUX machine, if configured properly, whatever the distribution in use (in principle, it could be any LINUX or UNIX machine, either physical or virtual). To summarize, these are some of the supported combinations:

- USB key + RH/CentOS DVD (boot+RPM repo)
- USB key + RH/CentOS CD/DVD (boot) + external RPM repo
- USB key + PXE (boot) + RPM repo
- from an already available EPICO SERVER (PXE boot + EPICO over NFS + RPM repo)
- direct deployment on any LINUX machine (independently from the distribution)

Some promising preliminary tests have been conducted to create a self-consistent 8GB bootable USB key, with EPICO and the package repository as well.

4.1. How to handle complexity

The nodes that compose a cluster, can be divided by logical profiles, depending on the main task that the nodes are supposed to accomplish or the features they are going to provide; for instance, a master node or a front-end need different software than computing nodes, and probably will have different hardware requirements than storage nodes, as well as an I/O server will need different software than management and monitoring nodes or a workstation. Furthermore, hardware/software profiles, pools and subsets can be identified by the differences in hardware configuration, for instance nodes with ide/scsi/sata/sas hard disks, with or without raid configuration, attached to NAS, SAN, or just disk-less, with Infiniband/Myrinet/Gigabit networks, with or without bonding, with AMD or Intel processors, CPU or GPU oriented, involved in a grid and therefore requiring grid-enabling software, and so forth.
EPICO handles the following categories:

PROFILES
: subset of machines identified and divided by typology, purpose or major differences: master, iosrv, nodes, diskless, wks, ...

SUBPROFILES
: ad-hoc installations of single machines with minor differences related to network settings, partitioning, ..., handled using different extensions for the Kickstart `%include` files as defined on kernel cmdline (if the file exists; fallback to default otherwise), and a post-installation script executed at the end, if available (master@ICTP, master@TEMPLE, master@AAU, ...).

HOSTS
: single machine dedicated to a specific task or with peculiar characteristics (iosrv, node01, storage03, ...).

POOLS
: subset of machines, identified by hostname / IP / subnet or DNS TXT entry, with common characteristics or similar/identical hardware or special purpose (GPU, gpu01 $\Rightarrow$ GPU, gpu02 $\Rightarrow$ GPU)

4.1.1 Customization layers

From the first stage of the network boot-up, the EPICO server supplies:

- DHCP information
- PXE configuration file, EPICO keywords provided as kernel cmdline arguments in order to define profiles/subprofiles or force specific hosts

- kernel/initrd + kernel cmdline options
- Kickstart file
- DNS configuration (hostnames and TXT) to define pools, subset or customize by hostname/IP
- Kickstart `%include` files based on IP / hostname / profile
- pre-installation (`%pre`) and post-installation (`%post`) procedures externalized in a logical tree based on profiles/hosts or common defaults
- tasks/scripts that will be executed, or not, depending on a task list specific for each profile or host
- routines that check for hardware availability (e.g. presence of Infiniband card)
- Packages repository (base + extras)
- Post-boot procedure (startup script executed at each boot)
- RAMDISK integration (disk-less nodes)

4.1.2 Fallback procedure

Kickstart `%include` files, list of RPM packages, task-lists and scripts, are searched for in the following order:

1. ad-hoc (subprofile matching the extension of the files to be included)
2. host/pool specific (<PROFILE>/hosts.d/<HOST|POOL>)
3. default by profile (<PROFILE>/default/)
4. common (common/)

4.1.3 Main services involved

- PXE: network booting
- DHCP: IP binding + NBP (pxelinux.0)
- TFTP: PXE configuration file, alternative boot-up images (memtest[9], UBCD[10], ...)
- NFS: Kickstart + RPM repository (with little modification can be adapted to FTP/HTTP(S) based repos)
- POST-BOOT: uses port-knocking, ssh, c3-tools[11] (distributed shell)
- Configuration/Package Update: uses ssh, c3-tools

5. Future Developments and Open issues

In order to keep an high level of customization, we decided to develop the whole package as a collection of plain text scripts and configuration files. The complex structure we obtained allows great flexibility, but some knowledge about the services involved is a requirement, as well as scripting experience. Unfortunately, this limits its usage to experienced sysadmins and skilled users, but we aim to make it more user-friendly, providing configurators, installer and wrappers that will allow to easily modify configuration settings and parameters that currently have to be manually altered. Furthermore, we plan to support more distributions (e.g. Debian/Ubuntu) and newer versions of RPM based distros already supported, like Red Hat/CentOS 6 and Fedora 14/15.
Among the important issues that we have to face up:

- Redistribution of 3rd party and contribs (non-open licenses)
- EPICO as RPM package(s)
- BOOT from usbkey (self-consistent installer + repo)
- Improve HD/USB disk-overy and auto-partitioning
- Terminal and Graphical User Interface (TUI/GUI)

The positive feedback we have received so far, has encouraged us to continue the development and make it available as a free and open "product", in the hope it can be useful to the scientific community.

6. Web resources and download

- http://epico.escience-lab.org
- http://eforge.escience-lab.org/gf/project/epico/
- `svn co --username anonymous --password anonymous \`
  `https://eforge.escience-ab.org/svn/epico/trunk/distro`

References

[1] Anaconda/Kickstart – http://fedoraproject.org/wiki/Anaconda
[2] Red Hat Enterprise Linux – http://www.redhat.com
[3] CentOS – http://www.centos.com
[4] Fedora Project – http://fedoraproject.org
[5] White Box – http://whiteboxlinux.org
[6] FAI – http://fai-project.org
[7] Debian – http://www.debian.org
[8] Ubuntu – http://www.ubuntu.com
[9] memtest86+ – http://www.memtest.org
[10] UBCD – http://www.ultimatebootcd.com
[11] C3 tools – http://www.csm.ornl.gov/torc/C3