Postprint: Practical Experience in Automated Batch Deployment Management

Authors: Wei Haitao

Date: 2023-10-08T00:00:00+00:00

Abstract

Continuously improving the efficiency of automated deployment and controllable management capabilities in physical server environments has always been a persistent goal for operations and development personnel. With the continuous advancement of Xinhua News Agency's "Media Convergence Development Project", the construction of the foundational platform environment under the unified operations and maintenance system model is practically tasked with providing basic infrastructure support for upper-layer business application systems and managing the deployment and maintenance of nearly a thousand physical server nodes. This urgently necessitates solving the challenges of automated batch deployment and management of operating systems, as well as promptly responding to the deployment requirements of business systems.

Full Text

Practice of Automated Batch Deployment Management

Abstract: Improving the efficiency of automated deployment and controllable management capabilities in physical server environments has been a continuous pursuit for operations and development personnel. As Xinhua News Agency's "Media Convergence Development Project" has advanced, the construction of foundational platform environments under a unified operations system model has faced the task of providing infrastructure support for upper-layer business application systems. This has involved deployment and maintenance of nearly a thousand physical server nodes, creating an urgent need to solve the challenges of automated batch deployment and management of operating systems, while responding promptly to business system deployment requirements.

Keywords: Servers; Controllable Management; Automation; Batch

1. Challenges in Physical Server Deployment

Compared with traditional "siloed business" small-scale operations, large-scale operating system deployment under a unified operations system presents several key challenges:

(1) **Multiple coexisting system versions:** Data and infrastructure software require lower system versions, while containerized applications demand higher versions. (2) **Elevated and refined delivery interfaces:** Tasks extend beyond simple OS installation to include disk partitioning and mounting, software package installation, network configuration, NTP time synchronization, user settings, and personalization under unified design requirements. (3) **Large-scale deployment demands:** Physical server deployment typically involves hundreds of nodes across complex network environments with different security domains and multiple network interfaces per server. (4) **Deployment monitoring:** The ability to monitor and view detailed installation progress for each server throughout the deployment process is essential. (5) **Result verification:** Installation outcomes for network connectivity, user accounts, and software packages must be verifiable.

Addressing these challenges through manual deployment would inevitably result in low efficiency and high risk of configuration errors due to human factors.

2. Automated Deployment Solution

To address the weaknesses of large-scale OS deployment and operations, we designed an automated OS deployment and management solution tailored to Xinhua News Agency's actual environment.

[Figure 1: see original paper] System Architecture

The system components and their functions are described below:

(1) **PXE-based network automated installation environment:** Custom Kickstart (abbreviated as ks) automatic installation configuration files and shell scripts are established. The DHCP/BOOTP service is provided by Layer 3 switches in the management network, with DHCP services configured on the corresponding VlanIf interfaces during installation. This approach is more stable than traditional server-based DHCP configurations and better suited for cross-subnet batch server installations. The TFTP server, corresponding to the next-server configuration on the DHCP server, provides minimal boot image downloads and runs on the central control server. The operating system relies on xinetd services to manage tftp. Minimal boot image files for different OS versions (CentOS 6 and CentOS 7) are stored separately to enable simultaneous automated installation of multiple operating systems.

(2) **Configuration management service:** Developed to use server serial

numbers as indices, this service abstracts configuration data from ks files into parameters for encapsulated management, supporting server configuration management and configuration API queries.

(3) **Monitoring service:** Also using server serial numbers as indices, this service handles monitoring data upload, progress tracking, and installation verification.

The technical architecture is shown in Figure 1. The core components are the configuration management service and monitoring service. Server engineers compile the "Server Infrastructure Planning Table" and upload it to the configuration management server. Target hosts retrieve their configurations from this server during automated installation based on their serial numbers, with progress uploaded to the monitoring server throughout the process for verification by server engineers.

HTTP Web Server: Corresponding to the ks configuration file URL on the TFTP server, this provides resource downloads for ks configuration files and OS rpm packages. Running on the central control server, it stores ks files and rpm packages separately for CentOS 6 and CentOS 7 versions. Compared with traditional approaches, we added downloads for configuration files such as static routing and disk partitioning based on automated design requirements.

Configuration Management Server: Provides server configuration management operations for installation personnel, offering Excel-format configuration file upload and browsing functionality. This component was developed separately and runs on the central control server.

Monitoring Server: Provides real-time server installation and deployment monitoring services for installation personnel. This component was developed separately and runs on the central control server.

CMDB Server: The database for recording final server configuration states.

3. Automation Principles and Execution Flow

The OS installation automation is based on PXE and Kickstart network automated installation. Building upon this, we use server serial numbers as indices to extract personalized configuration data, encapsulating it as a configuration management service. Combined with custom shell scripts in the pre and post sections of ks configuration files, this enables automated deployment of personalized configurations.

Target host installation execution is divided into two phases:

Phase 1: Minimal boot image loading for target hosts, shown in [Figure 2: see original paper]. The target host obtains network card IP, gateway, TFTP server address, and minimal installation boot image path from the DHCP server via PXE-enabled network cards. It then downloads the boot image and installation startup configuration file from the TFTP server, retrieving the ks file URL

path from the configuration file. The pxelinux.0 file serves as the bootfile, while pxelinux.cfg/default contains the installation startup configuration with the ks file URL. The vmlinuz file is the boot kernel, and initrd.img is the mount file.

Phase 2: KickStart automated installation, shown in [Figure 3: see original paper]. The KickStart installation method provides two custom execution stages: %pre (pre-installation) and %post (post-installation), allowing us to write shell scripts for configuration customization. Common components are written directly into the ks file, including software package lists, root and public account configurations, SELinux and firewall settings, NTP configuration, and DNS configuration. For personalized configurations, variable shell scripts call the configuration management service, using the pre stage to import disk partitioning and hostname configurations, and the post stage to implement IP, static routing, and other system initialization configurations. All configuration operations in both stages upload data to the monitoring server.

4. Configuration Management and Monitoring Service Design

The configuration management and monitoring services are written in Node.js, providing configuration upload and browsing, configuration query APIs, and installation progress monitoring. The technical architecture is shown in [Figure 4: see original paper].

[Figure 4: see original paper] Configuration Management Service Technical Architecture

The configuration management service has the following characteristics:

(1) **Tool-oriented design:** Uses an in-memory database for data storage. Configuration metadata includes: server serial number, physical rack location, OS version, hostname, business network and in-band management network IP (corresponding to hardware NIC positions, including bonding), remote out-of-band management card IP, RAID partitioning, and disk partition mount points, as shown in [Figure 5: see original paper].

(2) **Excel integration:** To accommodate server engineers' work habits, an Excel import configuration conversion and adaptation module was added.

(3) **RESTful API:** Configuration query interfaces adopt RESTful API format, with detailed design for NIC configuration queries including IP address, netmask, subnet, gateway, and NIC name to address IP-related queries during ks automated installation and reduce text parsing workload in shell scripts. Example endpoints:

- http://[ip]:[port]/api/confs/[serialid]/operateIP
- http://[ip]:[port]/api/confs/[serialid]/operateIPNetmask
- http://[ip]:[port]/api/confs/[serialid]/operateIPGateway
- http://[ip]:[port]/api/confs/[serialid]/operateNic

(4) **Installation management:** After installation completion, the process and results are saved through the installation management interface.

4.1 Implementation Steps

OS installation automation liberates server engineers from repetitive labor, allowing them to focus on configuration planning and design with only minimal manual operations required:

(1) Download the "Server Infrastructure Planning Table" template from the configuration management server and use Excel to plan and design server configurations.

(2) Upload the Excel document to the configuration management server and add the listed servers to the pending installation list.

(3) Configure DHCP services on the corresponding VlanIf of the Layer 3 switch in the target host's current management network segment to begin system installation.

(4) Monitor installation progress through the monitoring service page ([Figure 6: see original paper]) and access each server's out-of-band management address to view system installation status via KVM remote console ([Figure 7: see original paper]).

4.2 Implementation Experience

Content not provided in original text.

5. Conclusion

This solution optimizes and improves upon traditional automated deployment approaches based on the characteristics of Xinhua News Agency's Media Convergence Development Project. It enhances large-scale server deployment configuration management capabilities and strengthens visual verification of deployment management. In practice, this solution has liberated server engineers from repetitive and labor-intensive tasks, achieving automated batch OS installation, significantly improving deployment efficiency, enhancing system operations and management capabilities, and solving the challenge of on-demand, on-schedule system delivery.

Minimizing personalized requirements from business systems on the infrastructure environment and enforcing standardization for system naming, user accounts, and OS versions are prerequisites for achieving automated deployment and improving controllable management capabilities. We imposed restrictions on business application systems from two key aspects:

(1) **Deployment standardization:** We established the "Host Domain Name and Internal Service Domain Name Naming Specification," "User Account

and Directory Usage Specification," "Operating System Initialization Specification," and "Multi-NIC IP Usage Specification" to ensure consistent system deployment.

(2) **Baseline OS versions:** Considering baseline software compatibility, OS stability, and official support timelines, we selected CentOS 6.8 and CentOS 7.3 as baseline versions. Database and big data software were deployed on CentOS 6.8 due to compatibility and stability requirements, while Java/J2EE applications suitable for containerization were deployed on CentOS 7.3. Official end-of-support dates for CentOS versions are shown in .

Official End-of-Support Dates for Operating Systems

References: [1] Kickstart documentation. http://pykickstart.readthedocs.io/en/latest/ [2] BOOTP. https://en.wikipedia.org/wiki/Bootstrap_{Protocol} [3] DHCP. https://en.wikipedia.org/wiki/Dynamic_{{{Host}}*{{Configuration}}}*{Protocol} [4] DHCP RFC. https://tools.ietf.org/html/rfc2132 [5] PXE. https://en.wikipedia.org/wiki/Preboot_{{{Execu
(Translated from: https://wiki.centos.org/About/Product)

Note: Figure translations are in progress. See original paper for figures.