BMC Software Inc.

Technical Disclosure Publication Document

Authors

Amol Manohar Vaikar
Samir Vinayak Bayani

Image-Based Provisioning Using Multiple Data Streams by Capturing Split Images

Posted: September 2012

Overview

This document describes a solution for image-based provisioning of Servers using multiple data streams by capturing split images.

Background

Cloud and provisioning solutions usually rely on “full stack provisioning,” but it is no longer enough to provision a barebones physical or virtual server with just the base operating system (OS) in cloud-like environments where provisioning of servers is required to be at very high speed on-demand. Currently, the time taken to provision a full-stack server using images captured from another machine is dependent upon the size of a captured image because data written on a disk while provisioning is in single stream from the captured homogeneous image. As the number of software packages in the stack increases, it leads to an increase in the image size, and also to the time taken to provision a machine using the image. Owing to the existing nature of the image created from a source machine and the methods currently used during provisioning, any kind of parallelism is not possible.

Cloud datacenter administrators would like to have one or more gold machines installed with the set of required packages and replicate these machines whenever new server provisioning request is triggered.

Solution

Reduce overall provisioning time in cloud-based environments:

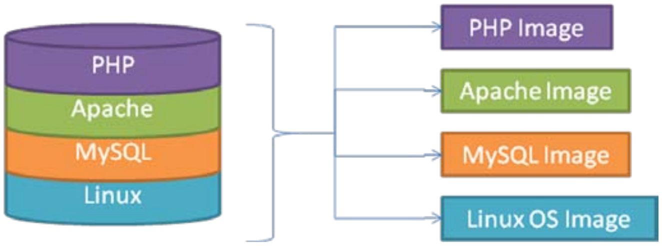
The goal of this invention is to reduce the provisioning time by:
1. Generating a set of sub-images to form a composite image (instead of single homogenous image) from the source machine.
2. Laying out only the basic sub-image (typically the core operating system) during provisioning and booting the new machine.
3. Once the new machine is up and running, laying out the other sub-images using parallel data streams.

As an additional advantage, generation of independent sub-images can also make it possible to carry out selective provisioning of software.

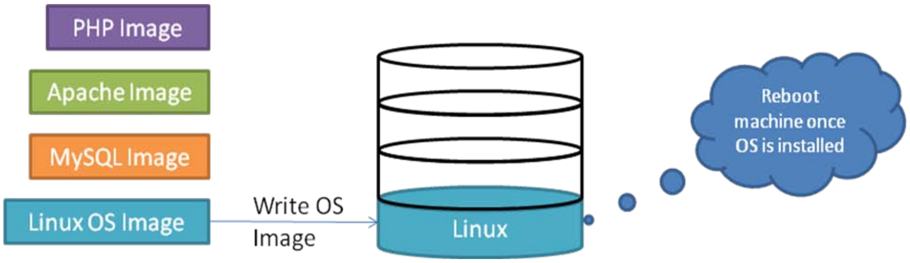
To illustrate, the source machine is running Linux OS for image creation and has Apache, MySQL, and PHP installed and configured.

While capturing an image from this machine a list of independent significant software and the components thereof is created. Files composing the software are then bundled up into distinct sub-images to form a single composite image.

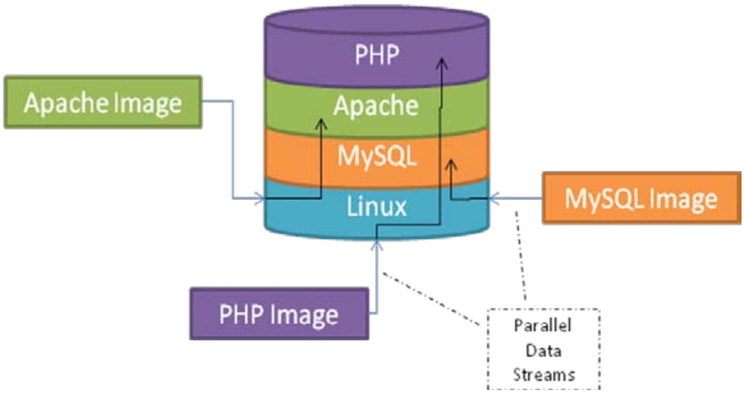
So for the above example, using the proposed solution, there would be a single image comprising of four sub-images, viz.:
Image comprising files of the Operating System
Image comprising files of Apache software
Image comprising files of MySQL software
Image comprising files of PHP software



While provisioning a target machine from this image bundle, first the OS will be laid out. This will enable booting the new machine into a proper multitasking/multiprocessing-capable operating system environment.



From this OS environment, there can be two or more parallel image layout processes running for the Apache, MySQL, and PHP images. This is possible because the software is not actually being installed (which usually involves locking software metadata); the contents of the images, as is depicted by the following diagram are merely being laid out as files:



While it is very unlikely that distinct software packages will have common files a common file could exist, and the conflict will have to be resolved to prevent data being created by one data stream from getting overwritten by a different parallel data stream.

To prevent this from happening, the data streams will have to ensure that they do not overwrite any existing files, but instead concatenate their data to such files. In the above example, this will ensure that configuration data for both Apache and MySQL can co-exist.

Further, owing to the split nature of the images, the solution can be extended to allow users to request only the provisioning of selected components instead of provisioning all the components from the source machine.

It is conceivable that distinct software packages have some common components that might cause a conflict during image creation or provisioning. This will result in the merging of two configuration files. For instances where conflicts exists while writing data from two parallel stream, this conflict situation could be presented to an Administrator or user to resolve.

Reduce overhead of multiple image management:

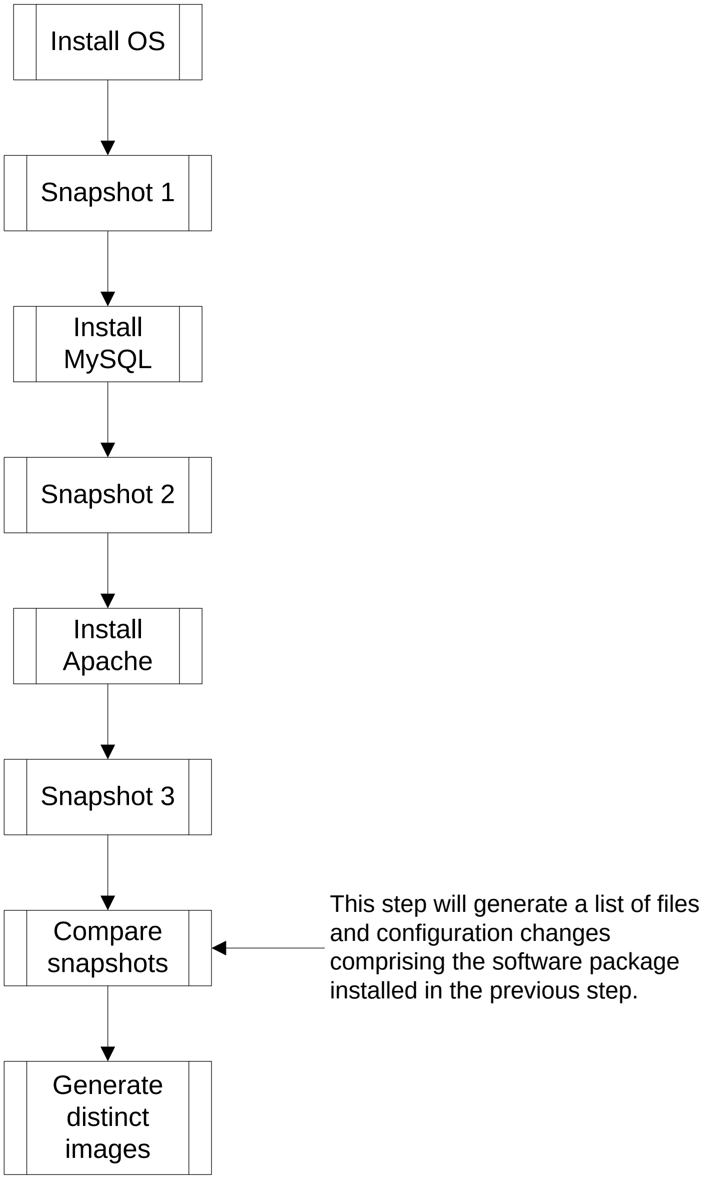
This method will capture a single composite image comprising of sub-images, including base operating system image and individual software images. For cloud/data-center administrators, sub-images will be abstracted as they will have to manage only the single composite image, with knowledge of all sub-images included in composite image.

The administrator has the liberty to choose the combination from the composite image and provision a server whenever a request comes up.

For example, a particular data-center may have a composite image containing MySQL, Apache and PHP. From this composite image any combination can be chosen, say OS + MySQL or OS + Apache + PHP to provision a server.

Drawings

Flow diagram below describes the process of capturing operating system for split images.



This step will generate a list of files and configuration changes comprising the software package installed in the previous step.