# The Rocks Avalanche Installer

## Background

The Rocks Clustering Toolkit (`www.rocksclusters.org`) has been in development for 5 years driven by the goal to *make clusters easy*. With more than 600 registered clusters and several clusters on the Top500 list, Rocks has become a staple of a large number of organizations. As clusters have grown in size, we have examined some of the scaling limits of the current implementation. A basic philosophy of Rocks is to use "fire and forget" re-installation as a basic management tool. The toolkit treats a complete software footprint on any machine as a set of software packages and configuration that together form a *Rocks Appliance*. Appliances can share packages and configuration and our approach takes advantage of the many similarities among login nodes, compute nodes, web servers, storage servers, grid-enabled nodes, and visualization walls. Instead of monolithic "golden" images for each appliance, Rocks defines a configuration graph that is "compiled" for each node at installation time. Using installation as the basic management tool only works on large-scale systems if this primitive operation is fast. Our new **Avalanche Installer** provides installation scaling into the 1000s of nodes with no special hardware. Like the rest of Rocks, Avalanche needs no special configuration by an administrator and the same installer supports clusters from the smallest (2 nodes) to the largest (1000s of nodes).

## The Avalanche Installer

A large number of cluster toolkits use "golden images" to define nodes and then use specialized software to push the image onto cluster nodes. Golden images do not easily handle heterogeneity in hardware or in node function. For example, a storage server node has a completely different image than a compute node. Scaling can also be problematic for images and others suggest reliable multicast as the image transport solution. Multicast itself also can be quite a challenge to configure correctly on large, multi-stage switched networks.

Avalanche does not rely on multicast. Because Rocks does not treat an OS as a monolithic image, we have significantly more flexibility in how to optimize without losing generality. Heterogeneous nodes (both hardware and function) are just as simple as homegeneous configurations. Avalanche attacks the two most time-consuming portions of a Rocks install: creation of the Red Hat-compatible Kickstart File (text-based configuration) and scalable serving of software packages. We could cheat and pre-generate configuration files, but Avalanche does not take this easy way out. Instead, we have split the generation of configuration files into two portions: the first reads the cluster configuration database (automatically created for you) and then feeds it to the second stage that formats it into a Red Hat Kickstart File. The latter part is time consuming (taking about 80% of the generation time) and is now performed on the installing node, not the frontend or head node. For scalable serving of packages, simultaneously-installing nodes form a peer-to-peer file sharing network using a BitTorrent tracker and a custom client. Nodes only share installation packages during the installation phase.

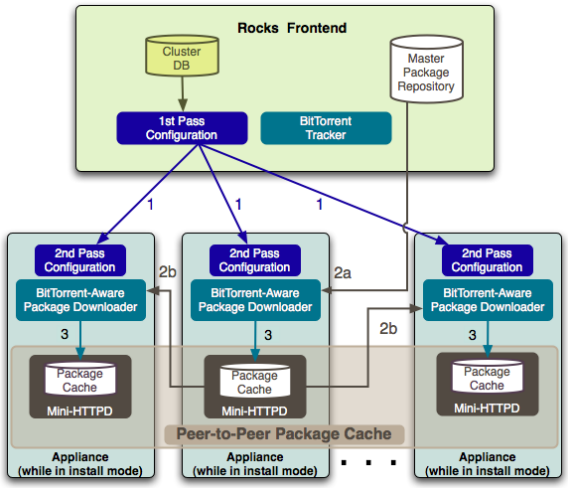


Figure 1: **Avalanche Ad-Hoc Peer-to-Peer Package Serving Network.** In step 1, the frontend sends the kickstart file in XML form to an installing node. Then, in the second pass, the node parses the XML and produces a Red Hat Kickstart File. In step 2a, a node downloads a package directly from the frontend. Step 2b shows how nodes can download packages from each other using the peer-to-peer package cache. After a compute node downloads a package, the BitTorrent-Aware Package Downloader places it into the peer-to-peer package cache (step 3).

Figure 1 depicts the splitting of configuration file generation and the ad-hoc peer-to-peer package serving network. The frontend is the definitive arbiter of package revisions and since the caching of packages is only valid while a node is installing, the cluster administrator does nothing to manage or keep caches in sync. Another advantage of this approach is that the number of installation nodes does not need to be known *a priori*. If several nodes are installing simultaneously, they will share packages. If only one node is installing, the tracker will point back to the frontend. Since modern networks are fully switched, the scaling of package serving scales linearly with the number of nodes. Avalanche does not affect other nodes that are busy computing because the tracker will only point to installing peers. Finally, because Rocks appliances share configurations and packages, different types of appliances still greatly benefit from the ad-hoc sharing. For example, a compute node can use the kernel package that has been cached by a web-server appliance.

# Performance

Avalanche scales very well. The effectiveness of Avalanche depends on the speed of the network, the total number of packages, and the time for configuration file generation on the frontend. We show two results below for old and new hardware using a trimmed compute node installation:

| Cluster | # Nodes | Node Type | Network | Packages/Node (MB) | Install Time (1 node/All) |
|---|---|---|---|---|---|
| Meteor | 44 | Pentium III | 100Mbit | 325 MB | 17 min / 20 min |
| Rockstar | 128 | Xeon 2.8 | 1000Mbit | 325 MB | 12 min / 15 min |

Table 1: Single and full-cluster installation timings. A full cluster installation (100s of nodes) take slightly longer than a single node. The extra time is attributed the component of Kickstart generation that runs on the frontend.

The following figure compares standard HTTP-only (prior to Rocks 4.1) and Avalanche (introduced in Rocks 4.1). A quick analysis shows that all the sum of all packages downloaded is about 14.3GB. A 100Mbit frontend can serve this amount of data in 1200 seconds (about 20 minutes). The total turn around time for this small cluster was less than 20 minutes with Avalanche and more than 30 minutes without.

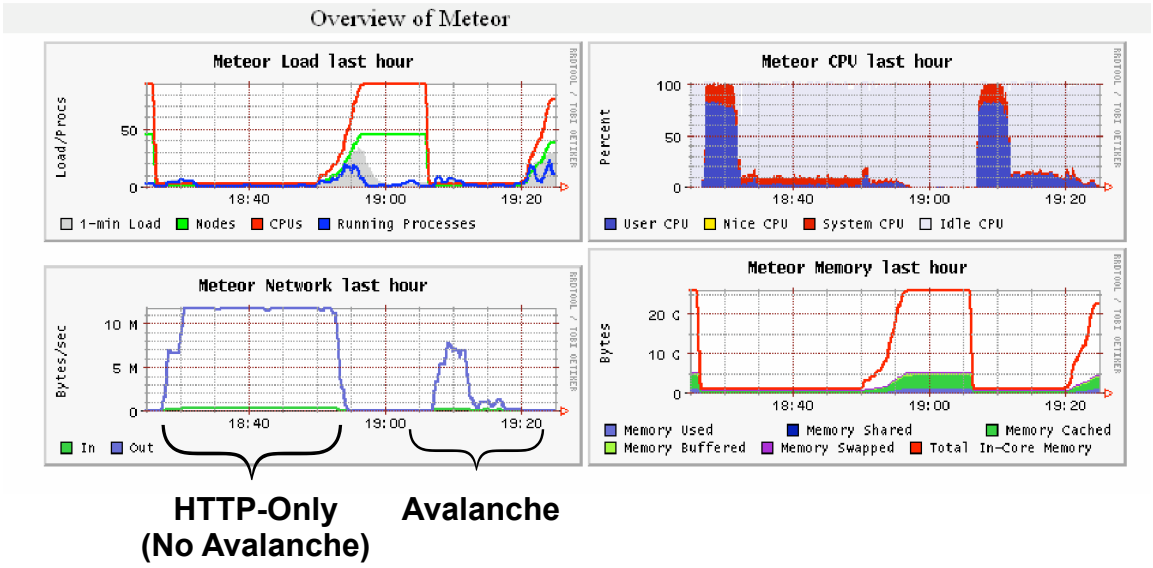


Figure 2: Without Avalanche (start time: 18:25), the full cluster re-installation takes more than 30 minutes. With Avalanche (start time: 19:05) takes less than 20 minutes for all nodes to complete. Note the reduction in network traffic directed to the Meteor Frontend when Avalanche is active. Meteor's frontend is an 800 MHz PIII with 100Mbit Ethernet.

While the nodes in the above figure are very slow by modern standards, a key observation is that the full-cluster install takes only 3 minutes longer than a single node install. Also, Rocks Avalanche handles full-system, heterogeneous, and trimmed installations with the same ease of "zero-administration".

# Acknowledgments

The Rocks Group at The San Diego Supercomputer Center is funded by the National Science Foundation as part of contract #OCI-438741. The Rockstar Cluster was made possible by a grant from Sun Microsystems. Meteor is the first Rocks Cluster built at SDSC.

We greatly appreciate the key interactions with several user groups who give us constant and constructive feedback.