Rebasable File Systems for Enhanced Management of Virtual Machines

Jinglei Ren*, Bo Wang, Weichao Guo, Yongwei Wu, Kang Chen, and Weimin Zheng
Department of Computer Science and Technology, Tsinghua University
*renjl10@mails.tsinghua.edu.cn

Management of large collections of virtual machines (VMs) becomes difficult and costly, as the number of VMs held by an enterprise proliferates. Leading VDI vendors have described numerous active deployments of over 10,000 users. Meanwhile, management has strong impact on the total cost of ownership (TCO).

Traditional provisioning and management of VMs are cloning-centric. A clone disk is created by linking to a read-only base image, and performs copy-on-write with modifications. This mechanism reduces space consumption and speeds up VM creation, but has two inherent drawbacks: (1) while administrators have to frequently update the base image to improve software and guarantee security, the block-level linking prevents these changes from being merged to previous diverged images; (2) while many misconfigurations manifest in the middle of the system's lifetime, there is no easy way for updated VMs to rollback to a previous base with users' latest changes reserved.

Current solutions (e.g. VMware View, Citrix XenDesktop) mitigate these problem by reinstallation of users' software and configuration each time users are forced to adopt a new base. But this remedy is hardly cost efficient as the number of VMs becomes excessively large. Traditional snapshot-based rollback is also problematic in that users' latest changes since the last update are not reservable. Meanwhile, versioning file systems neither lend much help as they currently only provide cloning-style semantics.

To overcome the limitations, we propose Cinquain, a file-based storage that enables a rebase-centric management model. Cinquain provides isolated file-system views, instead of virtual disks, for VMs. Most importantly, the linking to a base is changeable via rebase. Due to knowledge of file-level semantics, Cinquain is capable of merging the differential files of the child with the new base to form a functional root file system.

Cinquain and its rebasable file systems have the following advantages: (1) **Seamless propagation of updates**. *Forward rebase* of VMs to a new base breaks the inconvenience brought to users and burdens to the infrastructure in updating clones. (2) **Reservable rollback**. *Backward rebase* of VMs to a previous base rollbacks the system state but reserves users' latest changes (not reserved if using snapshots). This lightweight operation also enables on-site update directly in the target environment without costly whole system replica. (3) **Divide-and-conquer management** that enables a cooperative way for administrators and users to separately maintain their managed software in final VM views.

We make several contributions to achieve these advantages. (1) **The rebase-centric management model.** Our model provides flexibility in partial update or rollback of a VM. It also encourages a hierarchical organization of administrator roles. (2) **The table-walking metadata algorithms**. Rebase brings complexity to metadata. We overcome the challenge by elaborately designing a tagged tree to confine main metadata operations within linear time $O(n+m)$. (3) **Merging strategies based on our premier investigation of configuration files.** Merge of configuration is another challenge in rebase. We extensively study over 1,000 configuration files to summarize patterns and improve the three-way merge algorithm to generate functional configurations. (4) **Shadow uid/gid and micro file virtualization**. As uid/gid of different file systems may conflict, runtime rewriting of file attributes is employed.

Rebasable File Systems for Enhanced Management of Virtual Machines

Jinglei Ren Bo Wang Weichao Guo Yongwei Wu Kang Chen Weimin Zheng

Department of Computer Science and Technology, **Tsinghua University**

Motivation: Two Drawbacks of Fast VM Cloning for VDI and Cloud

Background: fast cloning creates a VM by linking it to a base. (Block-level mapping)

(1) Updates to the base cannot propagate to derived images.

Current Approach: coercively refresh users' images

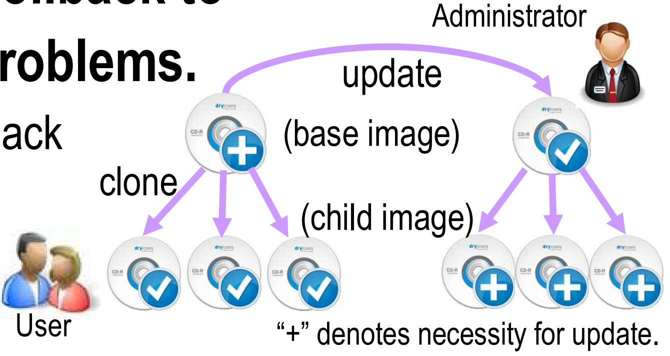
➡ Users have not long-lived software. ➡ Reinstallation in every VM is costly.

(2) Derived images cannot seamlessly rollback to a previous base when meeting config problems.

Current Approach: snapshot and whole-system rollback

➡ 16.7% - 32.4% misconfigurations happen in the middle of system lifetime.

➡ Changes since the last snapshot are lost.

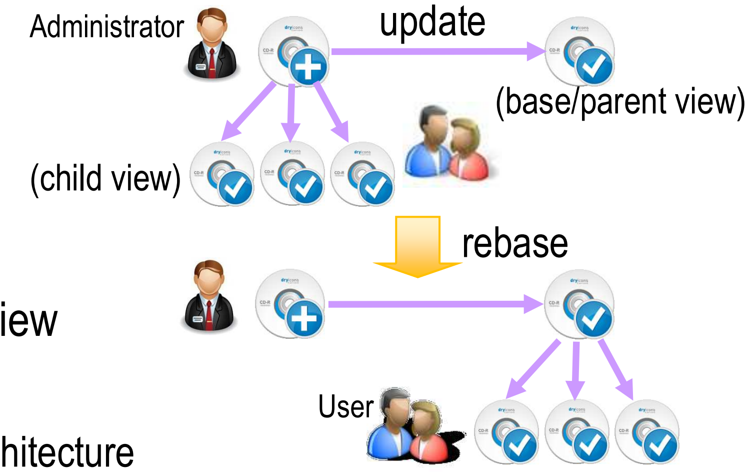


"+" denotes necessity for update.

Solution: Cinquain

- A file-based storage providing a file system *view* for each VM
- **Rebase operation for VMs** ➡

  Seamlessly changing the parent of a child view



Advantages

- **Seamless propagation of updates**

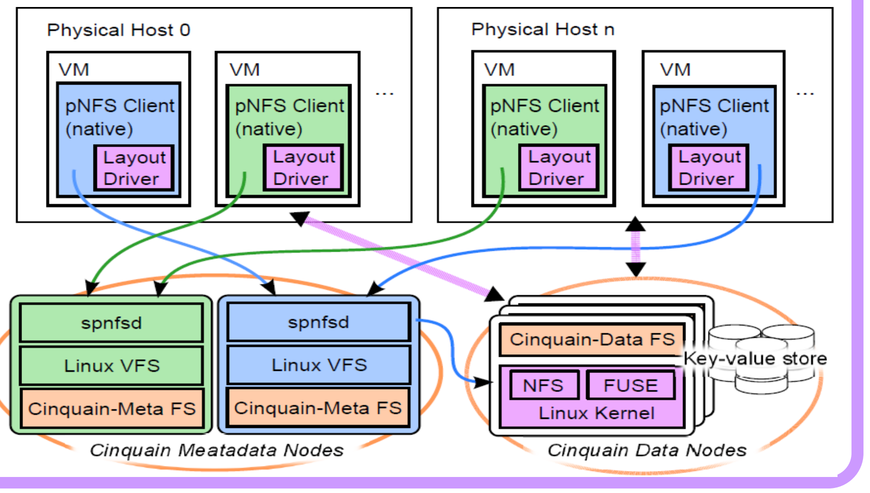
  by forward rebase
- **Support of reservable rollback**

  by backward rebase

  Changes betw. snapshots are reserved.
- **Divide-and-conquer management**

  Administrators and users separately maintain their own set of software.

Architecture



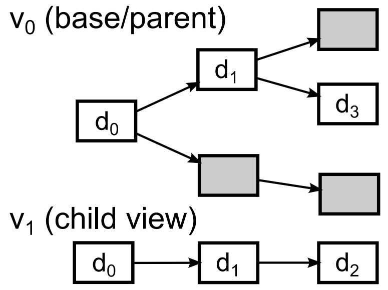
Challenge One: Metadata Algorithms

- Limitations of existing data structures

Views make a tree by derivation.

Each view only contains the dir/files that are different with its parent.

O(mn) to locate a file ($/d_1/d_2/.../d_n$, $v_0/v_1/.../v_m$), where d_i denotes dir/file, v_i denotes VM



- Table walking on a tagged dir tree within linear time *O(n+m)*

Only one dir tree with tags.

A dir/file is tagged ("+") with a view if the view has its own version.

Walk on a conceptual table to locate a dir/file: move right and down, from upper left to lower right; stop at the border or "-" (that means "removed").

	/	d_1	d_2	...	d_{n-1}	d_n
v_m	+	+	+	...		
v_{m-1}	+	+		...		
v_{m-2}	+	+	+	...	+	+
...						
v_2	+	+		...		
v_1	+	-	-	...	-	-
v_0	+	+	+	...		

Challenge Two: Merge of OS Configurations in Rebase

- An extensive investigation on over 1,000 software config files

  Patterns in config files: 1) single 2) list 3) key-value 4) group 5) mixed
- Extended three-way merge algorithm

  Apply the algorithm recursively until to a single line

  Design pre/post-merge extensions for each pattern to adjust Diff3 output

Original Base Version	Local Modified Version	Updated Base Version	Bare Output of Diff3

DIR_MODE=0755
SETGID_HOME=no
	DIR_MODE=0755
	SETGID_HOME=no
	EXTRA_GROUPS="dialout plugdev"
	ADD_EXTRA_GROUPS=1
		DIR_MODE=0755
		SETGID_HOME=no
		EXTRA_GROUPS="dialout plugdev extra"
		ADD_EXTRA_GROUPS=1
			<<<<<<< kv-m.conf
			EXTRA_GROUPS="dialout plugdev users"
			ADD_EXTRA_GROUPS=1
			||||||| kv-o.conf
			=======
			EXTRA_GROUPS="dialout plugdev extra"
			ADD_EXTRA_GROUPS=1
			>>>>>>> kv-y.conf

Example of Two Typical Merge Problems