Encompass: Managing Functionality

Oleg Goldshmidt, Benny Rochwerger, Alex Glikson, Inbar Shapira, Tamar Domany

{olegg,rochwer,glikson,inbar_shapira,tamar}@il.ibm.com.

IBM Haifa Research Lab

Agenda

- Motivation
- Managing functionality
  - Image management model
  - Image lifecycle
  - Some useful scenarios
- Image customization (if there is time left)
- Present and future

Motivation

Functionality and Resources

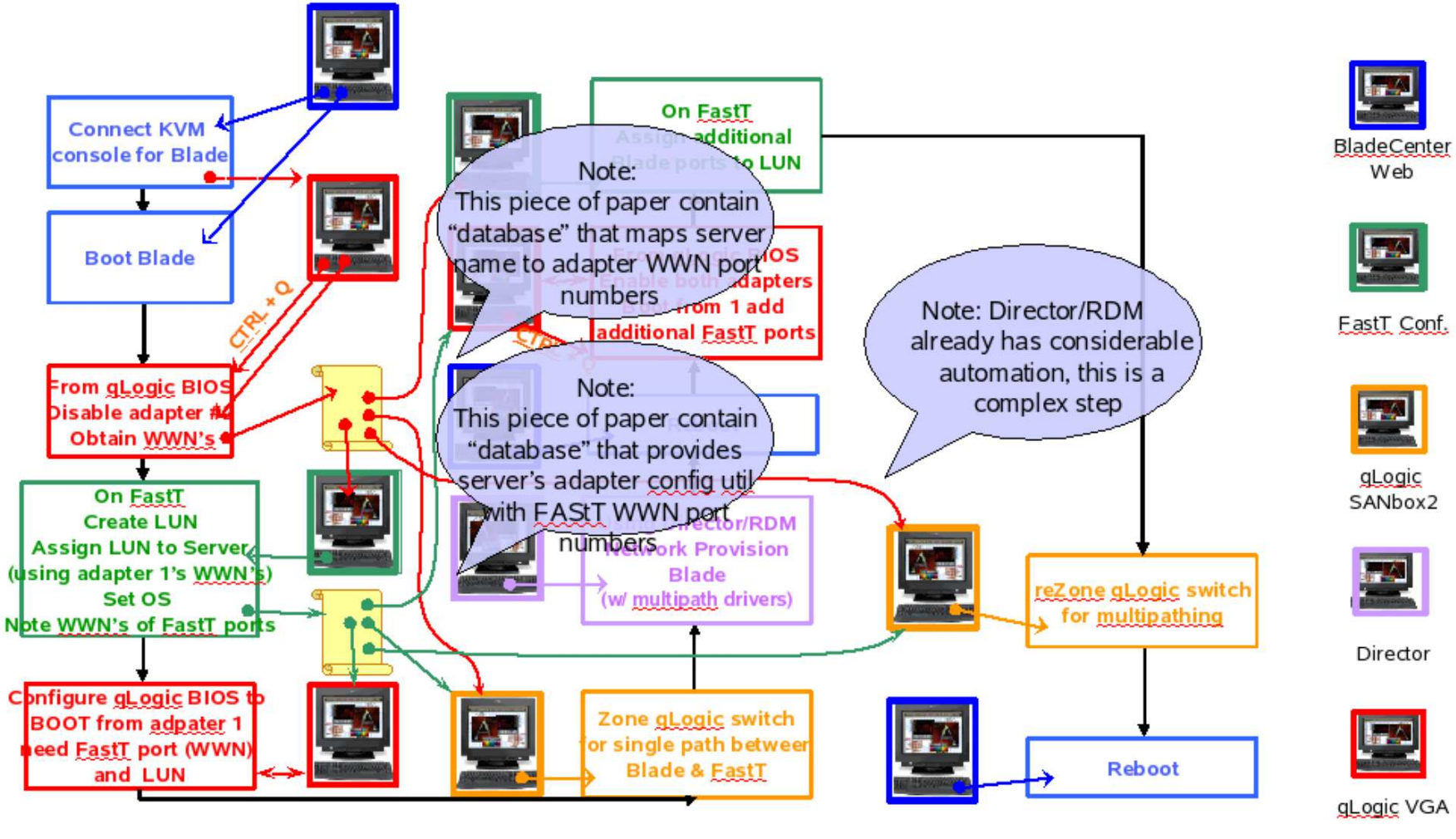
- customers are interested in functionality
  - determined by software and data residing on a disk (or disks)
- IT are managing resources
  - can your sysadmin identify your organization web server(s)? database server(s)?
- a machine is a familiar physical embodiment of
  - resources (CPU, memory) that determine performance
  - functionality (programs and data on disks)
    - disks are usually local (still)

Trends

- technologies that can potentially cause a paradigm shift
  - remote (shared) storage — SAN or NAS
    - separates functionality from computing resources
  - virtualization
    - creates a single resource pool
    - allows migration of functionality between physical resources
    - enables turn-key provisioning of complete SW stacks
- traditional management paradigm is still prevalent
  - pretend that everything works as before:
    - remote disks are just like local
    - virtual machines are just like physical

System Management Headache

Deploy image on remote storage and boot



Managing Functionality

Image Management Model

- image: a bootable software stack residing on one or more volumes and providing specific functionality (e.g., web server) when deployed; may be stored on block storage devices (e.g., SAN), files, etc..
  - all operations are performed on images
- machine: a set of computing resources (CPU, memory, etc.) that together form a deployment platform for an image; can be physical or virtual.
  - physical and virtual machines are treated similarly — both are just collections of resources
- masters and clones: there is a repository of master images; each encapsulates a specific of functionality; masters are never used directly, but are cloned; clones may be customized and deployed on machines.

Encompass Operation

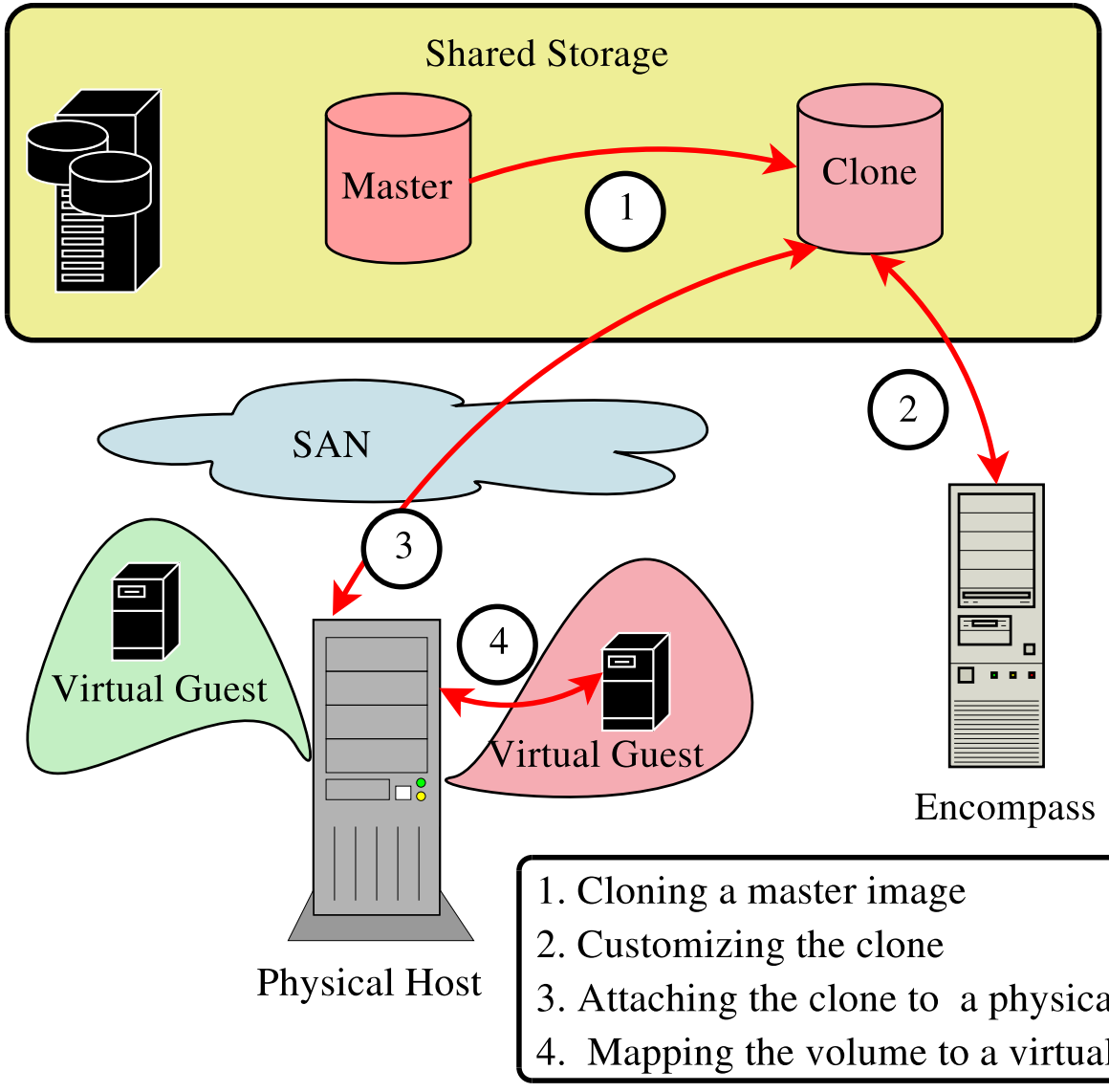
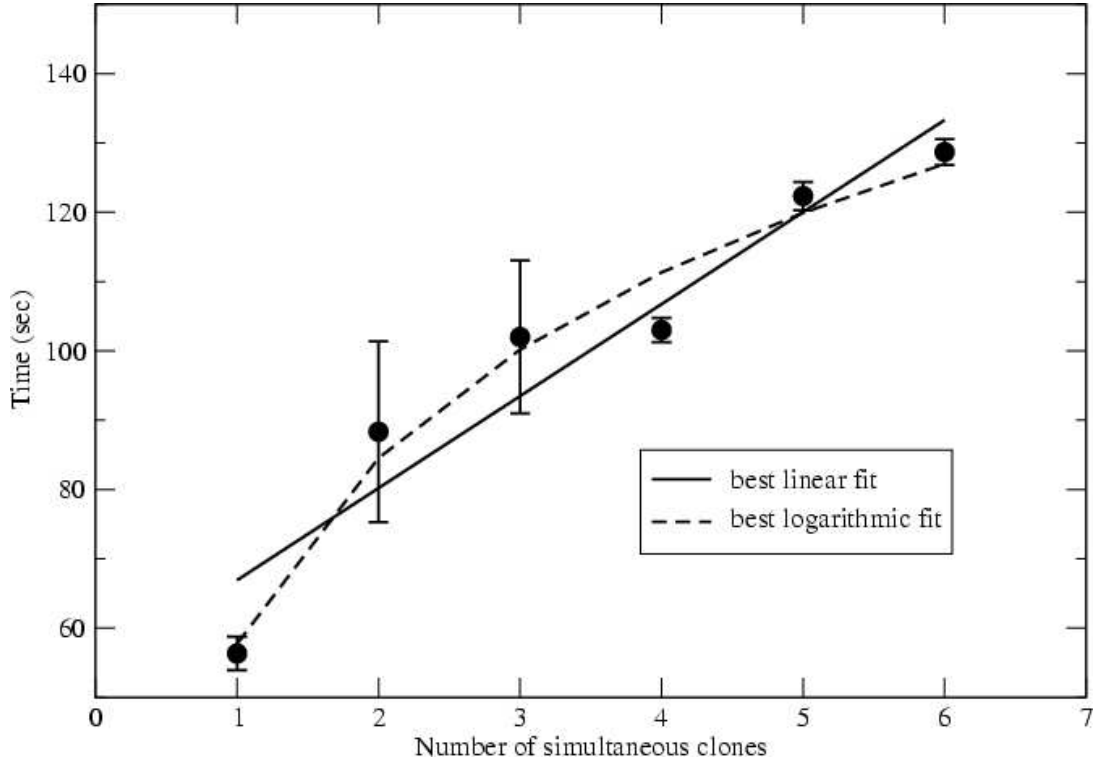


Image Metadata

- storage volumes
- HW profile — machine requirements
- SW profile — server functionality, system utilities
- state (discussed in detail below)
- related images
  - reference to master (for clones)
  - list of clones (for masters)
  - version control???
- machine the image is assigned to
- customizations (discussed in detail below)
- image bookkeeping is a central function

Image Cloning

- issues:
  - COW
  - RO/RW
  - parallelism
  - scalability
  - “cascading”
  - topology



- parallel cloning performance (see figure):
  - 3GB image from GPFS/DS4100 to local disks (`ext3`), copied by Xen

Some Useful Scenarios

- basic deployment
- migration of functionality
- resource reassignment
- updates and “baselining”
- export/import

Present and Future

Present

- research prototype — small, portable library, CLI
  - machine and storage discovery, persistent image metadata, merging discovered resources and metadata into a consistent picture, full image lifecycle implementation
  - storage: SAN (DS4xxx), NAS
  - physical machines: IBM BladeCenter
    - mostly for MM-assisted discovery
  - virtual machines: Xen, VMware ESX
- integrating with other IBM research projects as provisioning engine

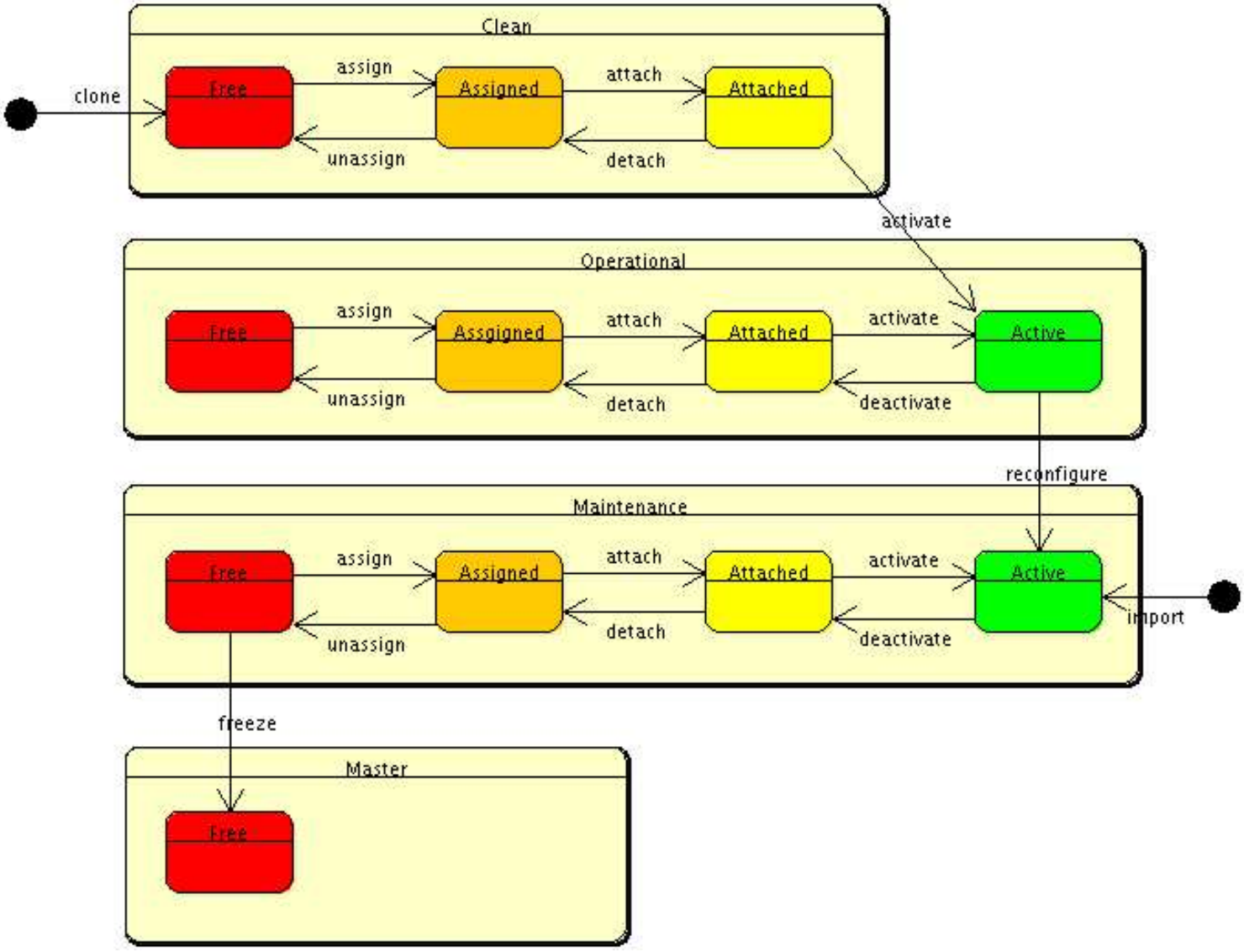
Future

- productizing: will be available as a part of IBM system management software stack
  - an independent implementation of the same set of ideas
- research
  - many largely unexplored areas: networking, security, cloning technologies
- other ideas: a “smart” storage controller that is aware of the stored functionality and helps provision and manage it?
- separation of concerns — a technology for a “flat” world?

Thank you!

BACKUP

Image Lifecycle



Resource Discovery

- discover physical resources — storage volumes and machines
- requires that physical machines be manageable even when no SW is running
  - IBM BladeCenter
  - Intel AMT
- either seed with minimal information about storage controllers, BladeCenters, or use SLP
- discovered resources must be matched with persistent image metadata
  - what if something changes in the physical world since the last metadata update? use heuristics to reconcile...

Image Customization

Image Customization

- customization: modification of the image contents and metadata to prepare it to run under particular circumstances
  - parameter configuration (hostname? static IP?)
  - set of services to start on boot
  - choose suitable OS kernel, set of drivers (initrd)
  - possibly p2v/v2p/v2v
  - application-specific
- target: zero or minimal customization

File-Based Customization

- UNIX philosophy: everything is a file
- procedure: mount a volume, add, replace, or modify the specified, files, unmount
  - target machine needs not be involved, can be done on a “utility server”, parallelized for many clones
  - e.g., for clones to be deployed in virtual machines the host OS, VMM, or a privileged domain may perform customization

File-Based Customization (Cont.)

- what if not everything is a file?
  - example from Microsoft world: `sysprep`
  - we do require unattended customization, so any proprietary tool should be able to read input non-interactively
    - `sysprep` can be given an input file
    - boils down to file replacement anyway
- special case: adding an executable/scrip to boot/init
- strive to achieve zero or minimal customization!

Customization In Image Metadata

- what to customize — in master metadata
  - ordered list of “customization items”
  - item: volume, partition, path, customization type (e.g., file replacement, regexp replacement), description, prompt
  - possibly defaults (to uncustomize)
- customization input — in clone metadata
  - input for each customization item (may be given interactively)
  - customization state (what has been customized already?)