XenSummit Asia

November 2-3, 2011

Seoul, Korea

Xen @ Google

Iustin Pop, <iustin@google.com>
Google Switzerland

Sponsored by:

SAMSUNG & KOREA UNIVERSITY (LIBERTAS JUSTITIA VERITAS) & kt & github SOCIAL CODING

Outline

Google

Overview

I will talk about...

- virtualization at Google:
  - in the corporate (internal) infrastructure
  - this is *not* used for user-facing products (search, gmail, ...)
- use cases, software used, tools and infrastructure

Terminology I might use (accidentally):

node physical machine (`dom0`)

instance virtual machine (`domU`)

Outline

Google

Corporate infrastructure

- comprises servers located in offices
  - support local office infrastructure
  - low-latency services (e.g. DNS, caches)
  - (very) small numbers of machines
  - spread across many offices
- and servers located in datacenters
  - various purposes
  - just a few datacenters
  - but many machines per datacenter
- note that we virtualise mostly Linux servers

Office deployments

- how to provide (redundant) services with as few machines as possible?
  - some offices are remote enough that timely replacement of parts is not a given
  - other offices are big enough to need multiple, redundant copies of services
- initial use of Xen (early 2006), and start of tool development
- this allowed shrinking footprint down to 2-4 machines
- and improved reliability against hardware failures
- however it somewhat decreased software reliability

Datacenter deployments

- in DCs we have multiple use cases:
  - again corporate infrastructure, e.g. DNS, LDAP, etc.
  - team servers/one-off applications
  - virtual workstations
- and the challenges are different:
  - scaling management software to many machines
  - capacity planning
  - redundancy across racks
  - intra and inter-DC VM moves

Use case: server consolidation

- the "standard" way to use virtualisation
- reduces hardware/footprint/power
- services owned by dedicated services team or by a small team
  - redundancy can be implemented at VM level or at service level
  - resource guarantees needs can vary from "very strict" to relaxed
- interesting corner cases
  - services which cannot afford the downtime of live migration
  - services which cannot afford the performance penalty of virtualisation

Use case: desktop virtualisation

- internal project named *Ubiquity*
- allows every engineer to have a virtual workstation in the "internal cloud" (a nearby datacenter)
- accessible over either SSH or NX
- advantages:
  - workstation state stored in the "cloud", not on (less managed) physical workstation
  - workstation closer to datacenter-based services
  - easier to provision more hardware in a datacenter than in a (possibly space-restricted) office
  - workstations can follow people as they travel
- potential issues:
  - depending on hardware refresh cycles, a dedicated physical workstation can be more powerful than a shared virtual one
  - latency to datacenter can sometimes be a problem

Google

Use case: machine management layer

- some workloads are too big for a shared environment
- but virtualisation has other advantages beside consolidation:
  - independence from hardware (well, storage...)
  - the hypervisor layer can abstract/unify hardware monitoring
  - much easier to move to new platforms
- hence the use of virtualisation in single-VM-per-machine model, aka "dedicated" model
- still in testing
- what do to when size of VM smaller than size of HW?
- currently investigating a "hard-partitioned" model:
  - share machines, but do not oversubscribe any resource
  - try to isolate CPU cores, disk spindles, network, RAM

Outline

Google

Overview

- we deploy Xen...
  - on standard (off-the-shelf) x86 hardware (amd64)
  - on top of standard operating systems (Debian and Ubuntu)
  - in paravirtualised mode
- no SAN/NAS: compute nodes are storage nodes as well
- layered software model: machine $\Rightarrow$ cluster $\Rightarrow$ fleet
- machine level handles hardware and hypervisor management
- cluster level abstracts machines:
  - all resources are internal to, and managed by the cluster
  - software scales from one to a few hundred physical machines
  - upper level deals with clusters, not machines
- fleet level abstracts clusters:
  - end-users do not care about specific clusters (maybe geographic location)

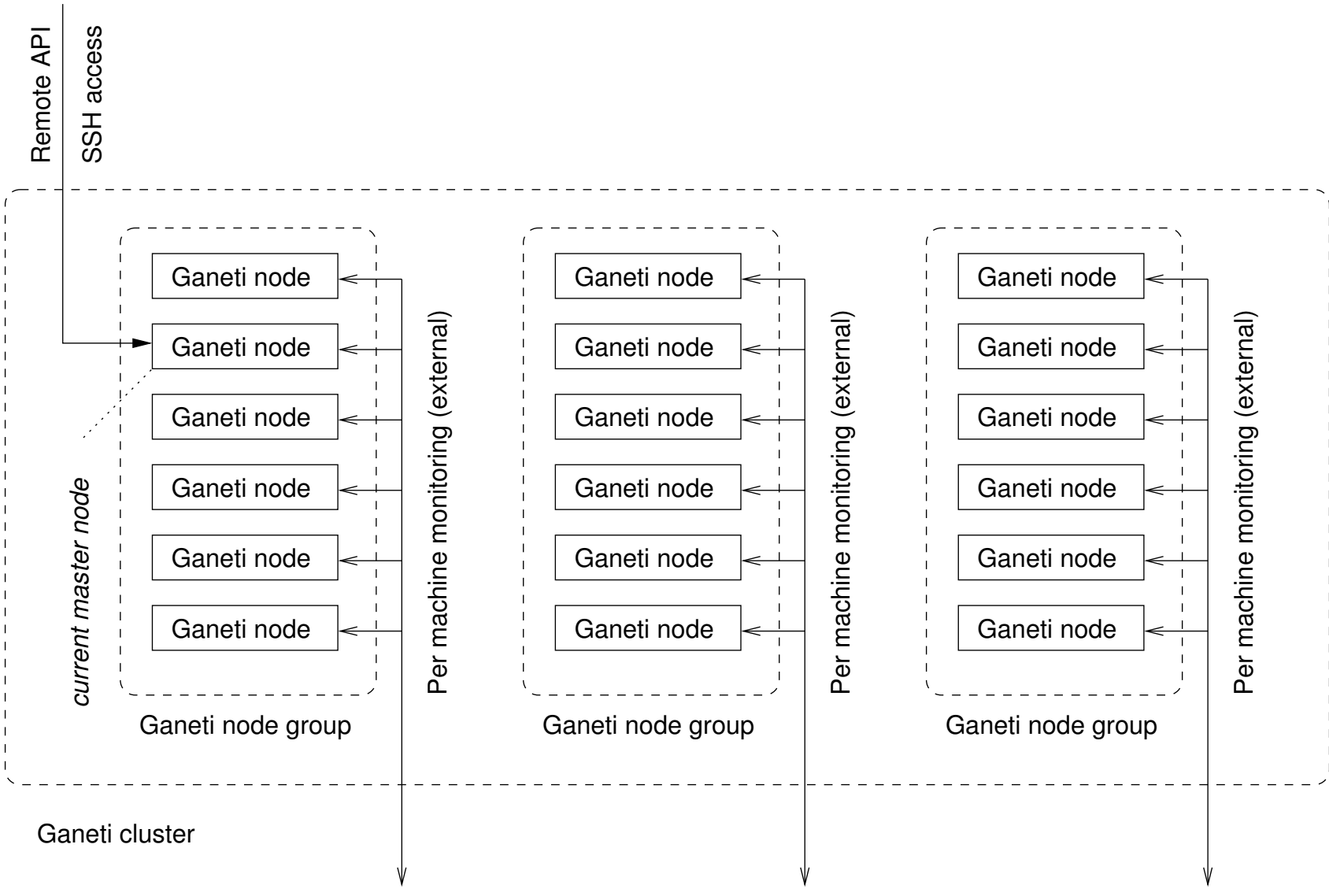
Machine level: OS/hypervisor

- we use Xen as KVM still has some disadvantages for us:
  - mostly performance-related, but note that also Xen pvops is not as good as Xen "native" (2.6.18 patch)
  - but the field is still evolving
  - and we could convert easily from one to the other
- we use Debian stable/Ubuntu LTS as base OS:
  - choice of OS is due to many Debian developers in the team
  - standard OS install, just trimmed down
  - standard tools for base OS configuration (cfengine/puppet)
  - all machine installs are fully automated
- at this level, it's mostly what you would get from a plain Debian + Xen install

Cluster level

- we use Ganeti as virtualisation manager
  - it supports other hypervisors but we only use Xen
  - for storage, we use mostly DRBD (network-level RAID 1), and also simple LVM storage
- cluster layout
  - physical machines ("nodes") are organised in "node groups"
  - multiple node groups constitute a cluster
  - the node group is the default mobility domain for the VMs
- no single point of failure for the cluster
  - one machine acts as "cluster master", but this role can be moved
  - no external resource dependencies (especially storage)
  - no network-level services required for the cluster operation
- all software at cluster level is open-sourced

Cluster diagram



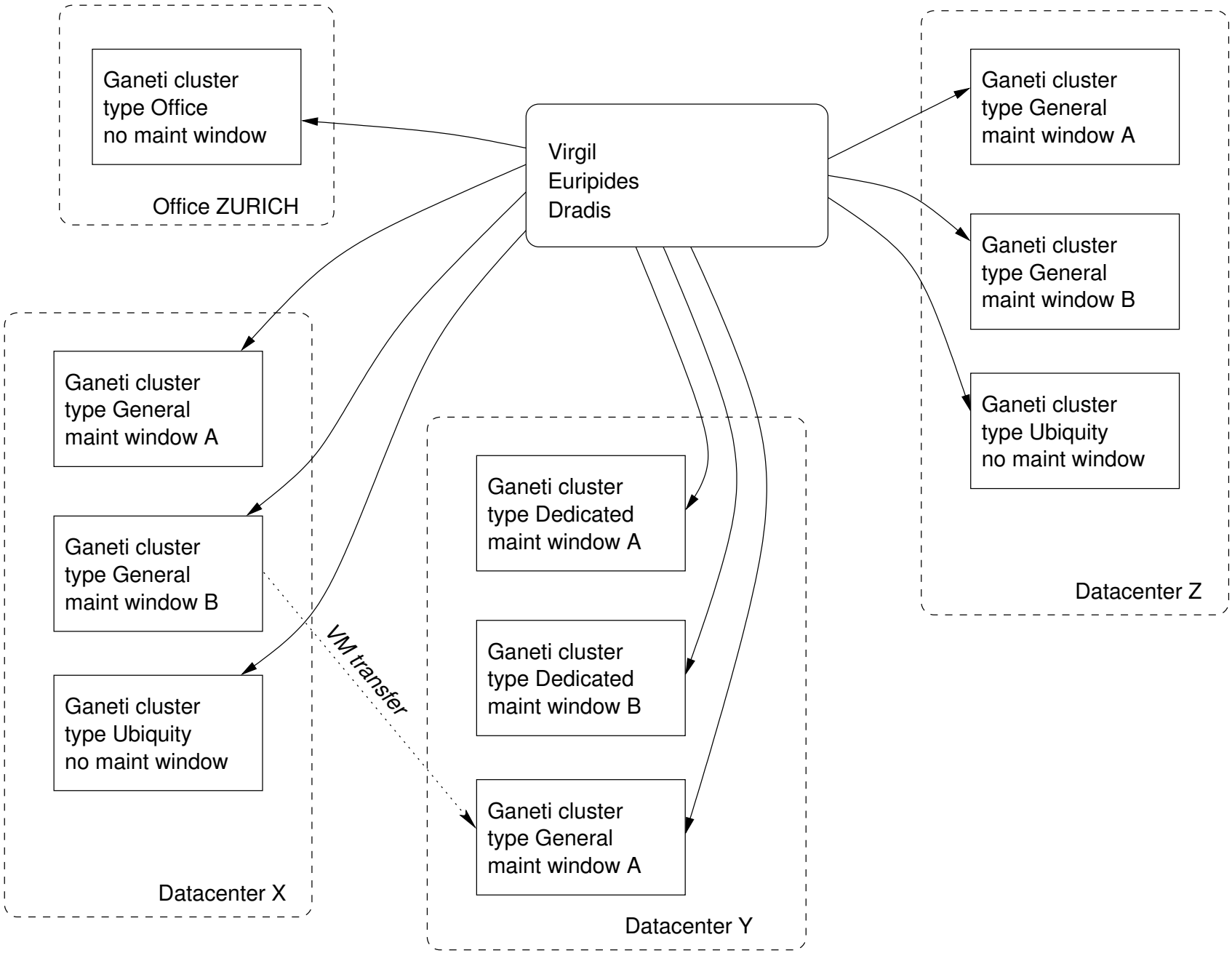
Fleet level

- we integrate with internal systems at this level
- this is done via internal software (not open source)
  - both generic (to Google):
    - monitoring
    - machine database
  - and specific to Ganeti-in-Google:
    - web interface to the clusters (code name *Virgil*)
    - cluster-level configuration management *(Dradis)*
    - machine (repair) workflow manager *(Euripides)*
  - these are related to hardware work-flows, not virtualisation
- the generic components have open-source alternatives
- "Ganeti Web Manager" is an open source web console
- no known equivalents for Dradis and Euripides
- large deployments of Ganeti will most likely need to reimplement them

Fleet organisation

- clusters are split (categorised) according to customer type
- clusters of the same type and in the same region are split in two "maintenance windows"
  - allow for maintenance work on only half of the clusters in a region
  - compensates for the fact that the cluster is a single point of failure for a given VM
- Virgil talks to all the clusters and provides fleet overview
- such meta-level organisation is implemented at Ganeti level via *cluster tags*
- tags are used for many other tasks that cannot be expressed directly at Ganeti level

Fleet diagram



Other internal tools

- machine history console
  - displays physical machine history
  - ties into monitoring, hardware repairs process, life-cycle, etc.
- `rolling-reboot` tool
  - allows rebooting an entire cluster without VM impact
  - uses live migration and sequential reboots
- `ganeti-capacity`: a capacity planning tool
  - computes simulated cluster capacity
  - VM specs versus physical resources, space, power
  - soon to be open sourced, not related to internal systems
- and many other small tools
  - notification of owners per cluster/physical machine
  - monitoring and resource dashboards
  - etc.

Known issues

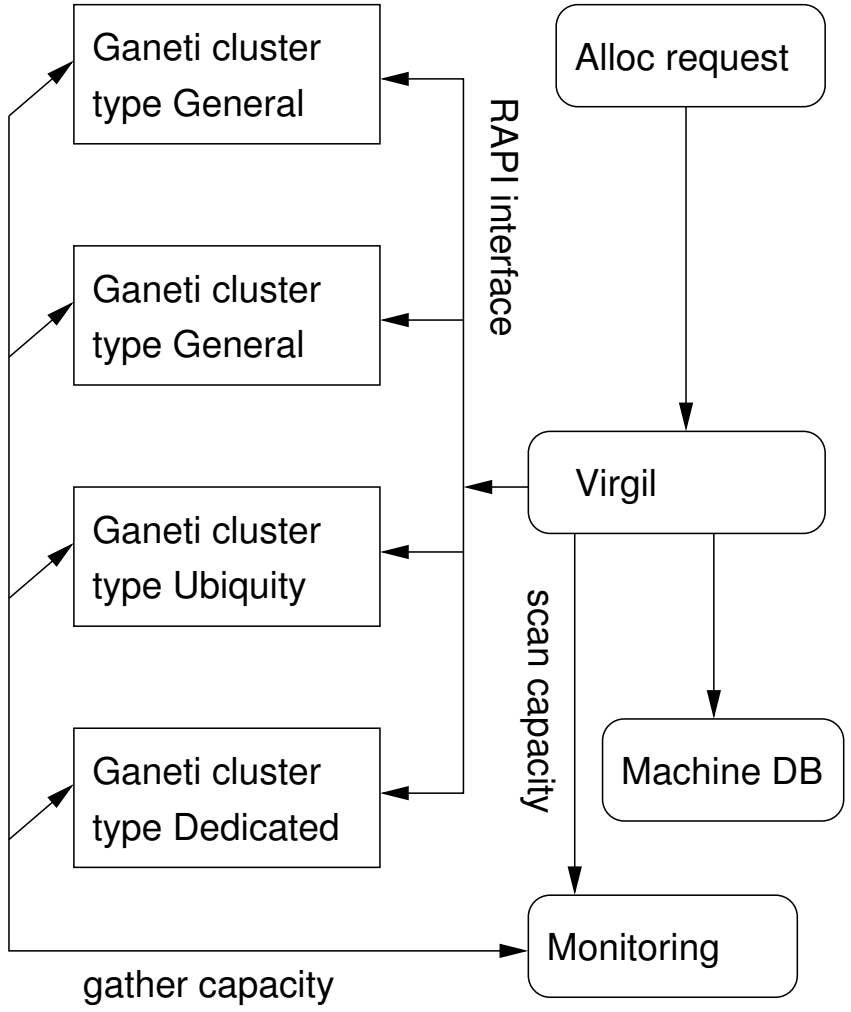
- VM clock issues:
  - a long-standing problem
  - we still see cases where VM clocks are rolled back 3000s due to machine clock-source problems
- IO issues
  - DRBD + Xen much lower performance than just DRBD or just Xen
  - In general, hard to model I/O performance

Outline

Google

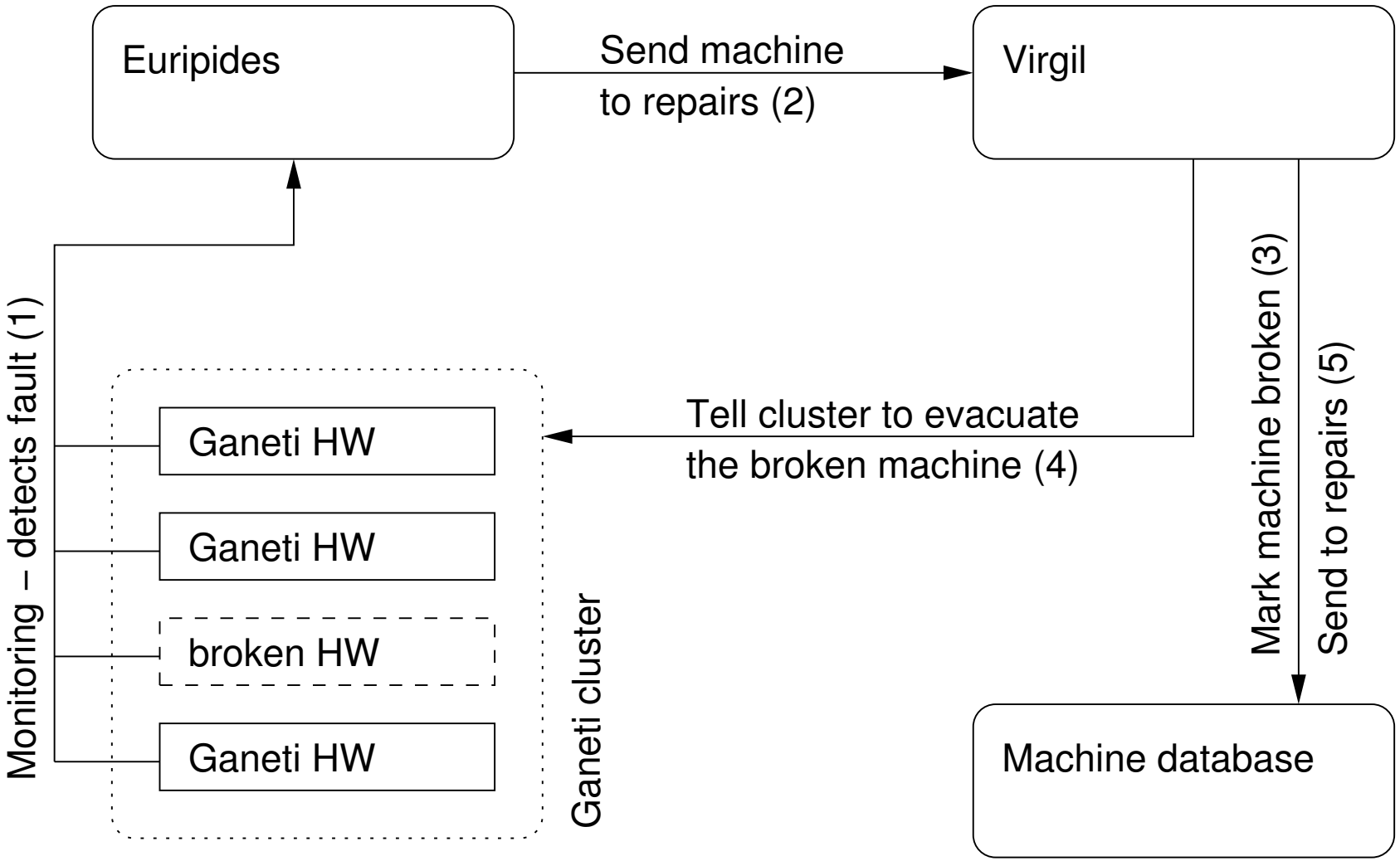
VM allocation

1. Virgil gets an allocation request (region, cluster type)
2. creates machine record (DNS, other systems)
3. selects "best" cluster based on VM spec, capacity data
4. and tells it to create the VM
5. cluster selects best physical machine(s) to host the VM
6. VM is created, and OS installation scripts are run
   - install software
   - configure authentication



Google

Handling machine failures



Google

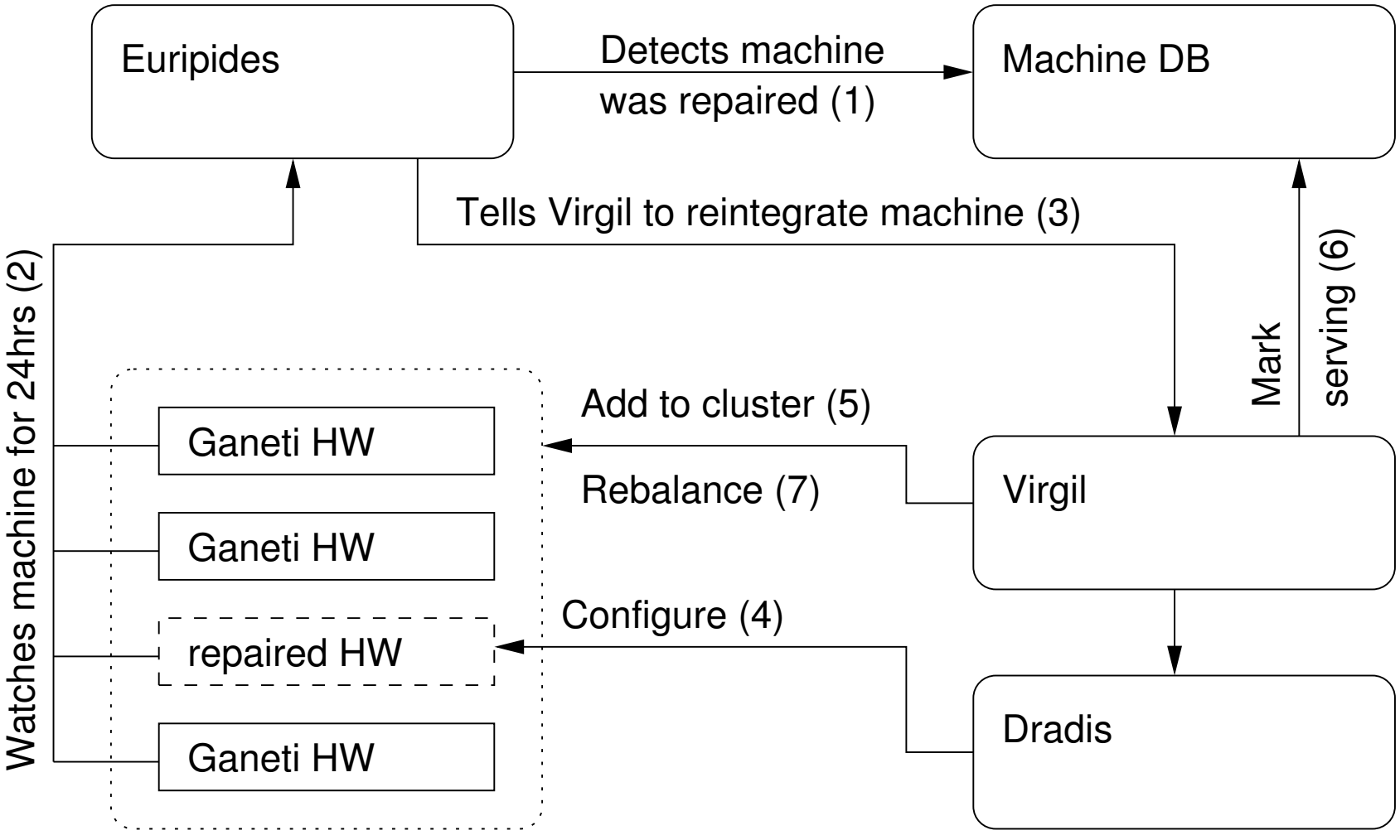
Handling machine failures

1. monitoring detects a HW problem (e.g. disk error, memory problem, etc.)
2. Euripides (for non-critical problems) tells Virgil a machine needs to be sent to repairs
   - for critical events (machine dead), on-call person is paged, instructs Euripides how to proceed
3. Virgil first marks the machine as "not in production"
4. then tells the cluster to evacuate the VMs from it
5. finally requests repairs by local tech

Note

- for "known" errors, the process if fully automated
- otherwise, an "exception" case is created for investigation

Handling repaired/new machines



Handling repaired/new machines

1. Euripides detects new or repaired machine in Machine DB
   - at first, it's being kept "under watch" for a period of time
2. it tells Virgil to integrate new machine
3. Virgil calls Dradis to configure the machine appropriately
4. Virgil tells the cluster to add the new machine
5. finally the new machine is marked as serving
6. the cluster will be rebalanced in order to utilise the machine

Note

- assuming no errors in the OS installation, configuration, etc., the process is fully automated

Internal cluster workflows

- *htools* component shipped with Ganeti can
  - balance the cluster
  - compute cluster capacity
  - compute node evacuation strategy
  - do automatic selection of nodes for VM placement
- plugin versus API
  - node evacuation and instance placement use internal "IAllocator" plugin framework
  - the other are command line tools that run talk to Ganeti using its external APIs
- the documentation explains how to use all of these

Outline

Google

Software road map I

- improve the cluster resource model
  - currently models only RAM/VCPUs/disk space as resources
  - will add spindles/networking (IO) resources
  - will add dynamic memory (ballooning/tmem) support
  - will improve support for non-Xen hypervisors
  - addressing these in the next releases
- improve remote API: eliminate the need for SSH
  - some operations not available over RAPI
  - will integrate e.g. cluster capacity reporting
  - ongoing effort towards full parity
  - eventually SSH will not be needed for operations
- will add "hard-partitioning" model (still being designed)

Software road map II

- improve the VM OS deployment model:
  - currently OS scripts run on the physical machine
  - this requires trusted source for OS templates
  - Ganeti users must write their own installation scripts
- improve Xen CPU scheduler control
- add smart LVM allocation
  - currently based on simple biggest-free-space model
  - optimisations possible for DRBD layout, flash usage, etc.
- better handling of SAN/NAS storage

Deployment road map I

- looking at using ballooning/tmem
  - first need support in Ganeti and capacity planning tools
  - plan to start testing early next year
- fleet refresh in progress
  - current fleet a mix of older and newer hardware
  - many machines still using only 1GbE
  - machine heterogeneity makes cluster algorithms more complex
- investigating "embedded OS" `dom0` model
  - current `dom0` is a regular Linux distribution
  - sub-optimal with many machines performing the same role
  - all machines should be (roughly) identical at all times
  - aiming at an image-based setup in order to eliminate the installation step and individual package upgrades

Deployment road map II

dom0 kernel versions unification:

- currently running a mix of kernel versions
  - originally we used 2.6.18 + "native" Xen patch
  - "native" Xen had better performance than "pvops"
  - for us, 2.6.3x kernels have I/O performance problems
- still trying to identify a current, well performing kernel
- this prevents us from properly cooperating with upstreams
  - would like to give feedback on stability and performance
  - but hard to track down patterns across multiple versions

Questions?

Thanks!

Links

Ganeti homepage `http://code.google.com/p/ganeti`

Code repositories `http://git.ganeti.org/`

Documentation `http://docs.ganeti.org/ganeti/current/html/`

Ganeti Web Manager `http://code.osuosl.org/projects/ganeti-webmgr`

Image-based OS template
`http://code.osuosl.org/projects/ganeti-image`

Presentation on virtual workstations `http://neatx.googlecode.com/files/herding-virtual-workstations-fisl-2009.pdf`