Ganeti

The Cluster Virtualization Management Software

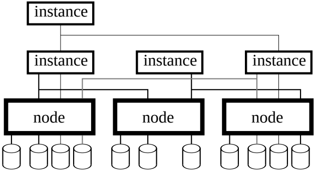
Helga Velroyen (helgav@google.com)
Klaus Aehlig (aehlig@google.com)

August 24, 2014

Cluster

For Ganeti, a cluster is

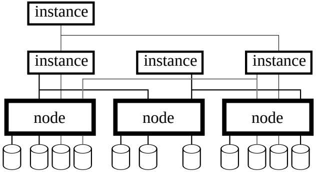
- virtual machines ("instances")
- on physical machines ("nodes") using some hypervisor (Xen, kvm, . . .)
- and some storage solution (DRBD, shared storage, . . .).



Cluster Management

Ganeti helps

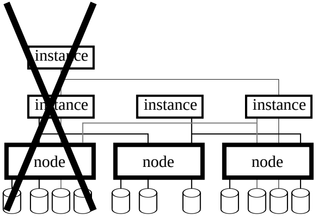
- to get there
  - uniform interface *hypervisors/storage/...*
  - policies, balanced allocation



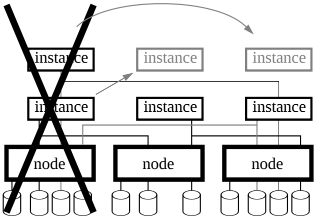
Cluster Management

Ganeti helps

- to get there
  - uniform interface *hypervisors/storage/...*
  - policies, balanced allocation
- and to stay there



Cluster Management



Ganeti helps

- to get there
  - uniform interface *hypervisors/storage/...*
  - policies, balanced allocation *keeping $N + 1$ redundancy*
- and to stay there
  - failover instances
  - rebalance
  - Restart instances after power outage
  - ...

Basic Interaction—Cluster creation

- `gnt-cluster init -s 192.0.2.1 clusterA.example.com`

Basic Interaction—Cluster creation

- `gnt-cluster init -s 192.0.2.1 clusterA.example.com`
- `gnt-node add -s 192.0.2.2 node2.example.com`

Basic Interaction—Cluster creation

- `gnt-cluster init -s 192.0.2.1 clusterA.example.com`
- `gnt-node add -s 192.0.2.2 node2.example.com`
- ...

Basic Interaction—Cluster creation

- `gnt-cluster init -s 192.0.2.1 clusterA.example.com`
- `gnt-node add -s 192.0.2.2 node2.example.com`
- ...
- `gnt-instance add -t drbd -o debootstrap -s 2G --tags=foo,bar instance1.example.com`

Basic Interaction—Cluster creation

- `gnt-cluster init -s 192.0.2.1 clusterA.example.com`
- `gnt-node add -s 192.0.2.2 node2.example.com`
- ...
- `gnt-instance add -t drbd -o debootstrap -s 2G --tags=foo,bar instance1.example.com`

The `-o debootstrap` references the OS definition to be used. An OS definition essentially is a collection of scripts to `create`, `import`, `export`, . . . an instance.

Basic Interaction—Planned Node maintenance

Evacutating a node

-
```
gnt-node modify --drained=yes node2.example.com
```

Basic Interaction—Planned Node maintenance

Evacutating a node

- `gnt-node modify --drained=yes node2.example.com`
- `hbal -L -X`

Basic Interaction—Planned Node maintenance

Evacutating a node

- `gnt-node modify --drained=yes node2.example.com`
- `hbal -L -X`
- `gnt-node modify --offline=yes node2.example.com`

Basic Interaction—Planned Node maintenance

Evacutating a node

- `gnt-node modify --drained=yes node2.example.com`
- `hbal -L -X`
- `gnt-node modify --offline=yes node2.example.com`

Using the node again

- `gnt-node modify --online=yes node2.example.com`

Basic Interaction—Planned Node maintenance

Evacutating a node

- `gnt-node modify --drained=yes node2.example.com`
- `hbal -L -X`
- `gnt-node modify --offline=yes node2.example.com`

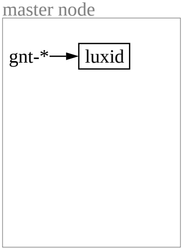
Using the node again

- `gnt-node modify --online=yes node2.example.com`
- `hbal -L -X`

Ganeti Jobs

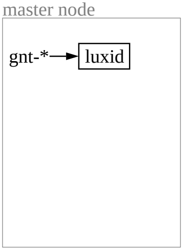
Ganeti Jobs

- gnt-* don't execute tasks they just submit jobs



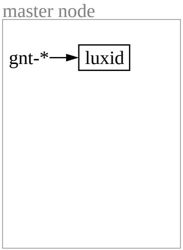
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
  - CLI does not have to wait; `--submit`
  - can be queried with `gnt-job info`



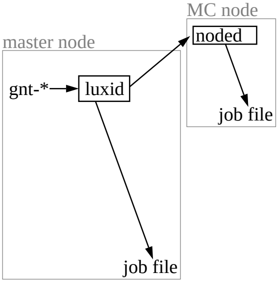
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs



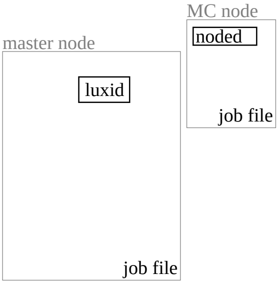
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
  - written to disk
  - replicated to some other nodes
    (the "master candidates")



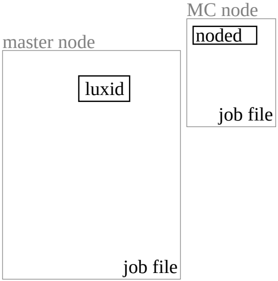
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job



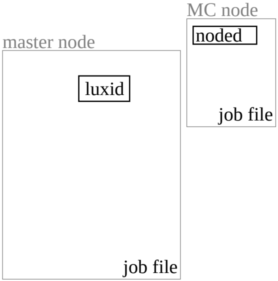
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- queued
  - limit on jobs running simultaneously
    (NEW: run-time tunable)



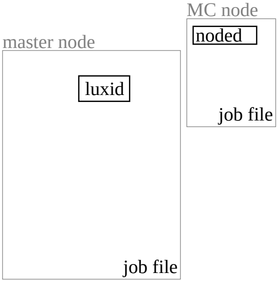
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- queued
  - limit on jobs running simultaneously
    (NEW: run-time tunable)
  - job dependency
    (NEW: honored at queuing stage)



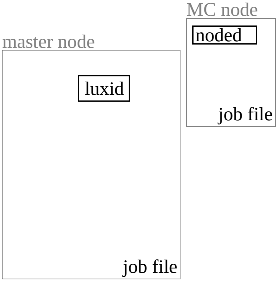
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- queued
  - limit on jobs running simultaneously
    (NEW: run-time tunable)
  - job dependency
    (NEW: honored at queuing stage)
  - ad-hoc rate limiting
    (NEW in Ganeti 2.13; more later)



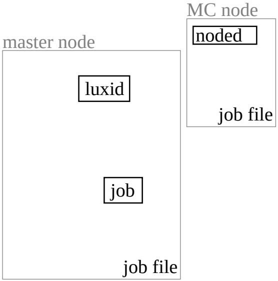
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- queued



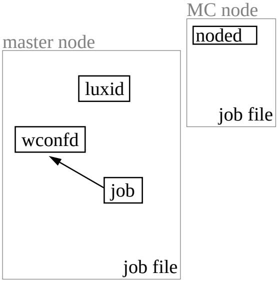
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- `queued`
- `waiting`
  - forked off, but still waiting for locks *(instances, nodes, . . .)*



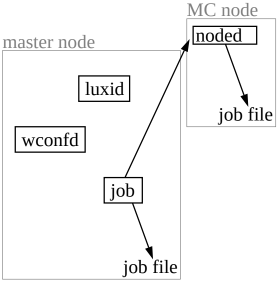
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- `queued`
- `waiting`
  - forked off, but still waiting for locks *(instances, nodes, . . .)*
  - Reading configuration



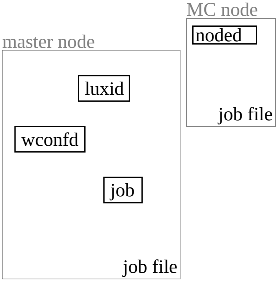
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- `queued`
- `waiting`
  - forked off, but still waiting for locks *(instances, nodes, . . .)*
  - Reading configuration
  - Already responsible for its own job file



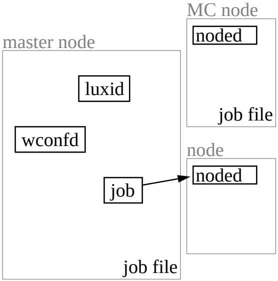
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- `queued`
- `waiting`



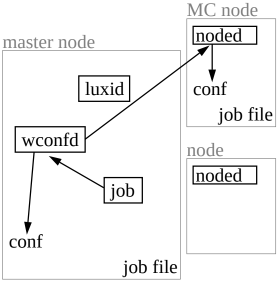
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- queued
- waiting
- running
  - Actual manipulation of the world via noded



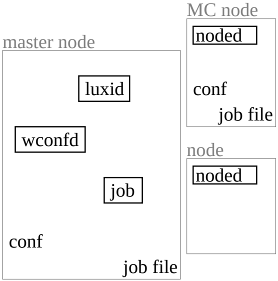
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- `queued`
- `waiting`
- `running`
  - Actual manipulation of the world via `noded`
  - Updates the configuration



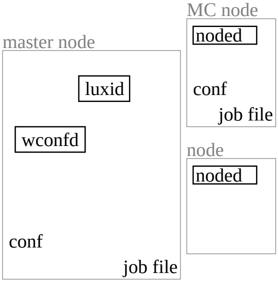
Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- `queued`
- `waiting`
- `running`



Ganeti Jobs

- gnt-* don't execute tasks
  they just submit jobs
- luxid recieves job
- `queued`
- `waiting`
- `running`
- `success`
  (hopefully; or `error`, `canceled`*)*



Reason Trail

- Instead of running, jobs can also expand to other jobs

Reason Trail

- Instead of running, jobs can also expand to other jobs
  - cluster verification *(parallel verification of node groups)*

Reason Trail

- Instead of running, jobs can also expand to other jobs
  - cluster verification *(parallel verification of node groups)*
  - node evacuation *(parallel instance moves)*
  - . . .

Reason Trail

- Instead of running, jobs can also expand to other jobs

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
  - `hbal -L -X`

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
  - `hbal -L -X`
  - External tools on top of Ganeti

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
- To keep track why a particular job is run,
  parts are annotated with a “reason trail”

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
- To keep track why a particular job is run, parts are annotated with a “reason trail”
  - List of (source, reason, time) triples

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
- To keep track why a particular job is run,
  parts are annotated with a “reason trail”
  - List of (source, reason, time) triples
  - Every entity touching can (and usually does) extend

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
- To keep track why a particular job is run, parts are annotated with a “reason trail”
    - List of (source, reason, time) triples
    - Every entity touching can (and usually does) extend
    - Inherited on job expansion

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
- To keep track why a particular job is run, parts are annotated with a “reason trail”
    - List of (source, reason, time) triples
    - Every entity touching can (and usually does) extend
    - Inherited on job expansion
- The “reason trail” is also used for rate limiting *(Ganeti 2.13+)*

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
- To keep track why a particular job is run,
  parts are annotated with a "reason trail"
  - List of (source, reason, time) triples
  - Every entity touching can (and usually does) extend
  - Inherited on job expansion
- The "reason trail" is also used for rate limiting *(Ganeti 2.13+)*
  - Reasons starting with `rate-limit:`*n*`:` are rate-limit buckets

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
- To keep track why a particular job is run,
  parts are annotated with a “reason trail”
  - List of (source, reason, time) triples
  - Every entity touching can (and usually does) extend
  - Inherited on job expansion
- The “reason trail” is also used for rate limiting *(Ganeti 2.13+)*
  - Reasons starting with `rate-limit:`*n*`:` are rate-limit buckets
  - At most *n* such jobs run in parallel

Google

Reason Trail

- Instead of running, jobs can also expand to other jobs
- High-level commands can submit many Ganeti jobs
- To keep track why a particular job is run,
  parts are annotated with a "reason trail"
  - List of (source, reason, time) triples
  - Every entity touching can (and usually does) extend
  - Inherited on job expansion
- The "reason trail" is also used for rate limiting *(Ganeti 2.13+)*
  - Reasons starting with `rate-limit:`*n*`:` are rate-limit buckets
  - At most *n* such jobs run in parallel

```
gnt-group evacuate
--reason="rate-limit:7:maintenance 123" groupA
```

Instance placement

Instance placement

- Ganeti tries to keep utilization equal at all nodes

Instance placement

- Ganeti tries to keep utilization equal at all nodes
- Especially do so when creating new instances!
  (Saves later moves)

Instance placement

- Ganeti tries to keep utilization equal at all nodes
- Especially do so when creating new instances!
  (Saves later moves)
- IAllocator protocol
  - delegate decission where to place to external program
  - Given: cluster description and needed resources
  - Answer: node(s) to place instance(s)
- Most popular allocator `hail`
  Same algorithm as `hbal`

Instance placement

- Ganeti tries to keep utilization equal at all nodes
- Especially do so when creating new instances!
  (Saves later moves)
- IAllocator protocol
  - delegate decission where to place to external program
  - Given: cluster description and needed resources
  - Answer: node(s) to place instance(s)
- Most popular allocator `hail`
  Same algorithm as `hbal`
- Locking
  - need to guarantee that resources are still available once nodes are chosen
  - lock all nodes, release remaining after choice

Instance placement

- Ganeti tries to keep utilization equal at all nodes
- Especially do so when creating new instances!
  (Saves later moves)
- IAllocator protocol
  - delegate decission where to place to external program
  - Given: cluster description and needed resources
  - Answer: node(s) to place instance(s)
- Most popular allocator `hail`
  Same algorithm as `hbal`
- Locking
  - need to guarantee that resources are still available once nodes are chosen
  - lock all nodes, release remaining after choice

⇝ Instance creation sequential
Even if other nodes will eventually be chosen!

Opportunistic Locking

Parallel instance creation with `--opportunistic-locking`

Opportunistic Locking

Parallel instance creation with `--opportunistic-locking`

- Grab just the available node locks

Opportunistic Locking

Parallel instance creation with `--opportunistic-locking`

- Grab just the available node locks
- Choose among those nodes
  and release the remaining

Opportunistic Locking

Parallel instance creation with `--opportunistic-locking`

- Grab just the available node locks
- Choose among those nodes and release the remaining

⇝ New error type ("try again") if not enough resources on the available nodes

Opportunistic Locking

Parallel instance creation with `--opportunistic-locking`

- Grab just the available node locks
  NEW: but at least one (two for DRBD)
- Choose among those nodes
  and release the remaining

⇝ New error type ("try again") if not enough resources
on the available nodes

Opportunistic Locking

Parallel instance creation with `--opportunistic-locking`

- Grab just the available node locks
  NEW: but at least one (two for DRBD)
- Choose among those nodes
  and release the remaining

⇝ New error type ("try again") if not enough resources
on the available nodes

Planned: internal retry

Deployment at Scale

- RAPI
- Hspace
- Dedicated
- ExtStorage

RAPI

- RAPI = remote API
- RESTful
- Client library hides all the details
- You need the cluster name and credentials (for writing)
- Virtual IP for cluster master failover

RAPI - Python Client

Example usage of the Python client:

```
import ganeti_rapi_client as grc
import pprint

rapi = grc.GanetiRapiClient(’cluster1.example.com’)

print rapi.GetInfo()
pp = pprint.PrettyPrinter(indent=4).pprint
instances = rapi.GetInstances(bulk=True)
pp(instances)
```

RAPI - Python Client

Read/Write requires credentials:

```
import ganeti_rapi_client as grc

rapi = grc.GanetiRapiClient(’cluster1.example.com’)
rapi = grc.GanetiRapiClient(
’cluster1’, username=’USERNAME’, password=’PASSWORD’)

rapi.AddClusterTags(tags=[’dns’])
```

RAPI - Curl

Of course, you can just use with curl on the commandline:

```
> curl -k https://mycluster.example.com:5080/2/nodes
[{"id": "mynode1.example.com",
"uri":: "/2/nodes/mynode1.example.com"},
{"id": "mynode2.example.com",
"uri": "/2/nodes/mynode2.example.com"},

curl -k -X POST -H "Content-Type: application/json"
--insecure -d '{ "master_candidate": false }'
https://username:password@mycluster.example.com:5080 \
/2/nodes/mynode3.example.com/modify
```

Hspace - Capacity Planning

Running clusters, you might want to know:

- How many more instances can I put on my cluster?
- Which resource will I run out first?
- How many new machines should I buy for demand X?

Hspace simulates resource consumption:

- It simulates to add new instances till we run out of resources
- Allocation done like with hail
- Start with maximal size of instance (according to ipolicy)
- Reduce size if we hit the limit for one resource

Hspace - on a live cluster

```
> hspace -L

The cluster has 3 nodes and the following resources:
  MEM 196569, DSK 10215744, CPU 72, VCPU 288.
There are 2 initial instances on the cluster.
Tiered (initial size) instance spec is:
  MEM 1024, DSK 1048576, CPU 8, using disk template ’drbd’.
Tiered allocation results:
  -   4 instances of spec MEM 1024, DSK 1048576, CPU 8
  -   2 instances of spec MEM 1024, DSK 258304, CPU 8
  - most likely failure reason: FailDisk
  - initial cluster score: 1.92199260
  -   final cluster score: 2.03107472
  - memory usage efficiency:  3.26%
  -   disk usage efficiency: 92.27%
  -   vcpu usage efficiency: 18.40%
[...]
```

Hspace - Simulation Backend

Planning a cluster that does not exist yet

- Simulates an empty cluster with given data
- Format:
  - allocation policy (p=preferred, a=last resort, u=unallocatable)
  - number of nodes (in this group)
  - disk space per node (in MiB)
  - RAM (in MiB)
  - number of physical CPUs
- use `--simulate` several times for more node groups

Hspace - Cluster Simulation

```
> hspace --simulate=p,3,34052480,65523,24 \
  --disk-template=drbd --tiered-alloc=1048576,1024,8

The cluster has 3 nodes and the following resources:
  MEM 196569, DSK 102157440, CPU 72, VCPU 288.
There are no initial instances on the cluster.
Tiered (initial size) instance spec is:
  MEM 1024, DSK 1048576, CPU 8, using disk template 'drbd'.
Tiered allocation results:
  -  33 instances of spec MEM 1024, DSK 1048576, CPU 8
  -   3 instances of spec MEM 1024, DSK 1048576, CPU 7
  - most likely failure reason: FailCPU
  - initial cluster score: 0.00000000
  -   final cluster score: 0.00000000
  - memory usage efficiency: 18.75%
  -   disk usage efficiency: 73.90%
  -   vcpu usage efficiency: 100.00%
[...]
```

Ganeti Dedicated - Use Case

Use case:

- Offer machines to customers which require exclusive disk resources
- No two instances using the same disks
- Solution could be to use bare metal, but ...

You still want the benefits of virtualization:

- A different OS than the standard host OS
- Easy migration if hardware fails

Ganeti Dedicated offers exactly that.

Ganeti Dedicated - Realisation

Setup:

- Use Ganeti nodes with LVM storage (plain or DRBD)
- Make sure no two physical volumes share the same physical disk
- Flag nodes in a node group with `exclusive_storage`

Ganeti will:

- Not place more than one instance on the same physical volume
- Respect this restriction in operations like cluster balancing (hbal) and capacity planning (hspace)

ExtStorage - Setup

Ganeti's integration of shared / distributed / networked storage

- All nodes have access to an external storage (SAN/NAS appliance etc.)
- Instance disks reside inside that storage
- Instances are able to migrate/failover to any other node
- The ExtStorage interface is a generic way to access external storage

ExtStorage - Implementation

- For each type of appliance, Ganeti expected an 'ExtStorage provider'
- A bunch of scripts to do carry out these operations:
    - Create / grow / remove an instance disk on the applicance
    - Attach / detach a disk to / from a Ganeti node
    - SetInfo on a disk (add metadata)
    - Verify the provider's supported parameters
- Parameters transmitted via environment variables

ExtStorage - Examples

Assume you have two appliance of different vendors:

- `/usr/share/ganeti/extstorage/emc/*`
- `/usr/share/ganeti/extstorage/ibm/*`

Some example usages:

-
  ```
  gnt-instance add -t ext
  --disk=0:size=2G,provider=emc
  --disk=2:size=10G,provider=ibm
  ```

-
  ```
  gnt-instance modify --disk
  3:add,size=20G,provider=ibm
  ```

-
  ```
  gnt-instance migrate [-n nodeX.example.com]
  testvm1
  ```

-
  ```
  gnt-instance modify --disk
  2:add,size=3G,provider=emc,param5=value5
  ```

Google

Current Development - 2.10

- 2.10.7, available in debian wheezy backports
- KVM:
  - hotplug support
  - direct access to RBD storage
- Cross-cluster instance moves:
  - automatic node allocation on destination cluster
  - convert disk templates on the fly
- Cluster balancing based on CPU load
- Ganeti upgrades

Ganeti upgrades

Before:

- On all nodes:
  - `/etc/init.d/ganeti stop`
  - `apt-get install ganeti2=2.7.1-1 ganeti-htools=2.7.1-1`
- On the master node:
  - `/usr/lib/ganeti/tools/cfgupgrade`
- On all nodes:
  - `/etc/init.d/ganeti start`
- On the master node:
  - `gnt-cluster redist-conf`
- … lots of other steps, depending on the version
- If something goes wrong, fix the mess manually.

Ganeti upgrades

From 2.10 on, Ganeti comes with a built-in upgrade mechanism:

- On all nodes:
  - `apt-get install ganeti-2.11`
- On the master node:
  - `gnt-cluster upgrade --to 2.11`
- To roll back:
  - `gnt-cluster upgrade --to 2.10`

Note that you still have to install the new and deinstall the old packages manually.

Current Development - 2.11

- Current stable version, available in Debian Jessie
- RPC security: individual node certificates
- Compression for instance moves / backups / imports
- Configurable SSH ports per node group
- Gluster support (experimental)
- hsqueeze

hsqueeze

Huddle your instances during a cold cold night!

- Instances with shared storage (= live migration cheap)
- High load during peak times, low utilization otherwise
- Goal: During low utilization times, squeeze as many instances together as possible and shutdown unused nodes
- Use: Hsqueeze!
    - Calculates migration plan for instances
    - Aims to drain as many nodes as possible
    - But not too many to not cause resource congestion
    - Uses hbal to calculate balanced load
- In 2.11, only planning; in 2.13 including execution

Google

LXC

- LXC = Linux Containers
- Was experimental for a looong time (because nobody got time for it)
- Now: Google Summer of Code Project
- Goal: make it production ready, including a proper test chain
- Status: Going well, probably to be released in 2.13
- Works with LXC 1.0
- Live-migration still experimental

Disk Template Conversions

- Ganeti offers various disk templates for instances:
  - file, lvm, drbd, sharedfile, external storage
- So far, converting between those is only partially fun
- Google Summer of Code Project to make conversions smooth
- Status: Going well, probably release in 2.13

The Future

No guarantees!

- Improved Jobqueue management
- Network improvements (IPv6, more flexibility)
- Storage: more work on shared storage
- Heterogeneous clusters
- Improvements on cross-cluster instance moves
- Improvements on SSH key handling

Conclusion

- Check us out at `https://code.google.com/p/ganeti/`
- Or just search for "Ganeti"

Questions? Feedback? Ideas? Flames?

Upcoming Events:

- Ganeticon, Portland, Oregon, Sep 2nd - 4th