Ganeti Deep Dive

Technical details of changes since last GanetiCon (Part 1)

Klaus Aehlig <aehlig@google.com>

Sep 2, 2014

Daemon Refactoring

jobs as processes

Prelude: Rename `queryd` to `luxid`

- The renaming was already done in 2.9
  as the first step of a big daemon refactoring

Prelude: Rename `queryd` to `luxid`

- The renaming was already done in 2.9
  as the first step of a big daemon refactoring
- *that's all for 2.9, but big plans...*

Prelude: Rename `queryd` to `luxid`

- The renaming was already done in 2.9
  as the first step of a big daemon refactoring
- *that's all for 2.9, but big plans...*
  - `luxid` will handle all luxi requests
  - Ganeti jobs will run as processes
  - `masterd` will go away

`luxid` grows to the new role

All quiet in 2.10, but in 2.11...

`luxid` grows to the new role

All quiet in 2.10, but in 2.11...

- `luxid` learns *all* luxi commands

`luxid` grows to the new role

All quiet in 2.10, but in 2.11...

- `luxid` learns *all* luxi commands
  ...and becomes the standard luxi interface

`luxid` grows to the new role

All quiet in 2.10, but in 2.11...

- `luxid` learns *all* luxi commands
  ...and becomes the standard luxi interface
- Accepting jobs, `luxid` also writes to disk
  and does queue management

`luxid` grows to the new role

All quiet in 2.10, but in 2.11...

- `luxid` learns *all* luxi commands
  ...and becomes the standard luxi interface
- Accepting jobs, `luxid` also writes to disk
  and does queue management
  - limit number of jobs to be run at once
    cluster run-time tunable `--max-running-jobs`
  - hand over to `masterd` for execution: `PickupJob` request
  - watch job files for updates (via inotify; `--max-tracked-jobs`)

`luxid` grows to the new role

All quiet in 2.10, but in 2.11...

- `luxid` learns *all* luxi commands
  ...and becomes the standard luxi interface
- Accepting jobs, `luxid` also writes to disk
  and does queue management
  - limit number of jobs to be run at once
    cluster run-time tunable `--max-running-jobs`
  - hand over to `masterd` for execution: `PickupJob` request
  - watch job files for updates (via inotify; `--max-tracked-jobs`)

So `queryd` is gone...
if needed, would be easy to add an query-only option
(Speak out if you need it!)

Enter `wconfd`

- Authorative copy of the configuarion still in `masterd`
  but that is going away

Enter `wconfd`

- Authorative copy of the configuarion still in `masterd`
  but that is going away

⇝ Add a new daemon, `wconfd`,
to keep track of the configuration and locks

Enter `wconfd`

- Authorative copy of the configuarion still in `masterd`
  but that is going away

⇝ Add a new daemon, `wconfd`,
to keep track of the configuration and locks

  - query/update RPC via domain socket
  - changes written in batches and confirmed once on disk
  - asynchronous replication

Enter `wconfd`

- Authorative copy of the configuarion still in `masterd`
  but that is going away

⇝ Add a new daemon, `wconfd`,
to keep track of the configuration and locks

  - query/update RPC via domain socket
  - changes written in batches and confirmed once on disk
  - asynchronous replication

- Now fork/exec to start a new job

`locks.data` and live-locks

- In the good old days, when `masterd` died all its threads died
  No longer true!

`locks.data` and live-locks

- In the good old days, when `masterd` died all its threads died
  No longer true!

⇝ Persist lock status (as `locks.data`)
(only locally on master; if a node dies, all processes die)

`locks.data` and live-locks

- In the good old days, when `masterd` died all its threads died
  No longer true!

⇝ Persist lock status (as `locks.data`)
(only locally on master; if a node dies, all processes die)
Again, batch write, confirm once on disk

`locks.data` and live-locks

- In the good old days, when `masterd` died all its threads died
  No longer true!

⇝ Persist lock status (as `locks.data`)
(only locally on master; if a node dies, all processes die)
Again, batch write, confirm once on disk

- A dying job also doesn't kill `wconfd`

`locks.data` and live-locks

- In the good old days, when `masterd` died all its threads died
  No longer true!

⇝ Persist lock status (as `locks.data`)
(only locally on master; if a node dies, all processes die)
Again, batch write, confirm once on disk

- A dying job also doesn't kill `wconfd`

⇝ Each lock owner must prove he is still alive
We use advisory locks for this, on "live-lock files"

Pending requests and notify

- jobs need to wait for locks, administrated in a separate process

Pending requests and notify

- jobs need to wait for locks, administrated in a separate process

⇝ Request to assign locks as soon as available

  - notification via signal 1 (`HUP`)
  - job still has to verify that the request was granted

Pending requests and notify

- jobs need to wait for locks, administrated in a separate process

⇝ Request to assign locks as soon as available

  - notification via signal 1 (`HUP`)
  - job still has to verify that the request was granted

- Extensional change:
  a lock request will only be granted once all locks are available
  (in particular, no partial assignments)

Pending requests and notify

- jobs need to wait for locks, administrated in a separate process

⇝ Request to assign locks as soon as available

  - notification via signal 1 (`HUP`)
  - job still has to verify that the request was granted

- Extensional change:
  a lock request will only be granted once all locks are available
  (in particular, no partial assignments)
- To make better use of this feature,
  lock requests of adjacent levels are collated *(where possible)*

Opportunistic Locking

- new locking also allows for more complex requests like “some of those locks, but at least *n*”

Opportunistic Locking

- new locking also allows for more complex requests like "some of those locks, but at least *n*"
- Significantly reduces the number of `ECODE_TEMP_NORES`

Opportunistic Locking

- new locking also allows for more complex requests
  like "some of those locks, but at least *n*"
- Significantly reduces the number of `ECODE_TEMP_NORES`
  (especially when lots of instances are requested simultaneously,
  as `NAL` *wouldn't help there)*

News from the htools

`hail`, `hspace`, `hbal`, `hsqueeze`

Metrics computation in instance allocation

Background: `hspace` performance, changed in 2.10.5

Metrics computation in instance allocation

Background: `hspace` performance, changed in 2.10.5

- On instance allocation, all possible placements are considered and best scoring is taken
- Cluster score essentially is a sum of standard deviations and most nodes remain unchanged

$\rightsquigarrow$ Standard statistics $(n, \sum x, \sum x^2)$ can easily be updated

Metrics computation in instance allocation

Background: `hspace` performance, changed in 2.10.5

- On instance allocation, all possible placements are considered and best scoring is taken
- Cluster score essentially is a sum of standard deviations and most nodes remain unchanged

$\rightsquigarrow$ Standard statistics $(n, \sum x, \sum x^2)$ can easily be updated or $(n, \sum x, V)$ to be closer to the old values

! still extensional change in behavior
as floating-point round effective serves as a tie breaker

Improvement: factor 10 on 80-node cluster
(so sorry for the overhead to all small-cluster owners)

`hspace --independent-groups` *and* `hspace --accept-existing-errors`

- `hspace` hypothetically adds instances
  while keeping all nodes N+1-happy, then reports

`hspace --independent-groups` *and* `hspace --accept-existing-errors`

- `hspace` hypothetically adds instances
  while keeping all nodes N+1-happy, then reports
- Corollary: if one node is not N+1-happy, capacity is 0

`hspace --independent-groups` *and* `hspace --accept-existing-errors`

- `hspace` hypothetically adds instances
  while keeping all nodes N+1-happy, then reports
- Corollary: if one node is not N+1-happy, capacity is 0
- Might be a bit too conservative an estimate
  Estimate higher capacity by considering independent
  - `--independent-groups` the node groups
  - `--accept-existing-errors` the nodes
    (might over-estimate!)

`hbal --restricted-migration`

- New option `--restricted-migration` added to htools
  "This parameter disallows any replace-primary moves (frf), as well as those replace-and-failover moves (rf) where the primary node of the instance is not drained."

`hbal --restricted-migration`

- New option `--restricted-migration` added to htools
  "This parameter disallows any replace-primary moves (frf), as well as those replace-and-failover moves (rf) where the primary node of the instance is not drained."
- Use case: Updating the hypervisor
  for minor updates live-migration is possible—but only from the old to the new version

`hbal --restricted-migration`

- New option `--restricted-migration` added to htools
  "This parameter disallows any replace-primary moves (frf), as well as those replace-and-failover moves (rf) where the primary node of the instance is not drained."
- Use case: Updating the hypervisor
  for minor updates live-migration is possible—but only from the old to the new version
  - Drain node
  - `hbal -L -X --evac-mode --restricted-migration`
  - update, undrain, drain next node
  - `hbal -L -X --evac-mode --restricted-migration`
  - `...`

hsqueeze

- new htool, result of an informal discusion at last GanetiCon
  (That's why all those coffee breaks and dinners are essential!)

hsqueeze

- new htool, result of an informal discusion at last GanetiCon
  (That's why all those coffee breaks and dinners are essential!)
- Use case: clusters with huge usage variation
  $\rightsquigarrow$ Power down machines during low-usage times

hsqueeze

- new htool, result of an informal discusion at last GanetiCon *(That's why all those coffee breaks and dinners are essential!)*
- Use case: clusters with huge usage variation
  $\rightsquigarrow$ Power down machines during low-usage times
- Intended to be run by cron; will act if free resources per node
  - below `--minimal-resources`; power on nodes and balance only nodes tagged `htools:standby`
  - above `--target-resources`; balance, power down, and tag if afterwards still above

  Resources are measured in multiples of a standard instance

hsqueeze

- new htool, result of an informal discusion at last GanetiCon *(That's why all those coffee breaks and dinners are essential!)*
- Use case: clusters with huge usage variation
  $\rightsquigarrow$ Power down machines during low-usage times
- Intended to be run by cron; will act if free resources per node
  - below `--minimal-resources`; power on nodes and balance only nodes tagged `htools:standby`
  - above `--target-resources`; balance, power down, and tag if afterwards still above

  Resources are measured in multiples of a standard instance
- Please report about your experience by next GanetiCon!