





Ganeti

Ganeti Core Team - Google
LISA '13 - 5 Nov 2013



Monitoring

How and what to monitor in Ganeti

- Guido Trotter <ultrotter@google.com>
- Helga Velroyen <helgav@google.com>
- Some slides contributed by Michele Tartara <mtartara@google.com>

Latest version of these slides

Please find the latest version of these slides at:

<https://code.google.com/p/ganeti/wiki/LISA2013>

What to monitor

- "Right Now" things (pagable)
 - node down
 - instance down
 - cluster status
 - DRBD issues
- "Historical" things
 - Capacity
 - Utilization

Monitoring Clusters

The master IP should:

- ping
- answer to SSH
- respond to RAPI calls (`version` is a good 'no op')
- `gnt-cluster verify` output should not contain the word "ERROR"

Monitoring Nodes

- nodes should be pingable, ssh'able
- load average:
 - Xen: shouldn't be above 2.0 for very long
 - KVM: ...more complicated...
- DRBD issues
 - Nagios monitoring for DRBD:
 - <http://code.google.com/p/ganeti/wiki/DrbdDevicesMonitoring>

Monitoring Instances

- Do you provide monitoring for your users or are they responsible for it?
- Instance owner does it:
 - Do they have the skill?
 - Can you give them access to your monitoring system?
- Monitoring on behalf of the owner:
 - Best way to know if you are providing good service
 - "Fixed before the user notices"
 - Owner should be able to temporarily disable paging if instance will be down intentionally
 - Who gets paged? you or instance owner or both?

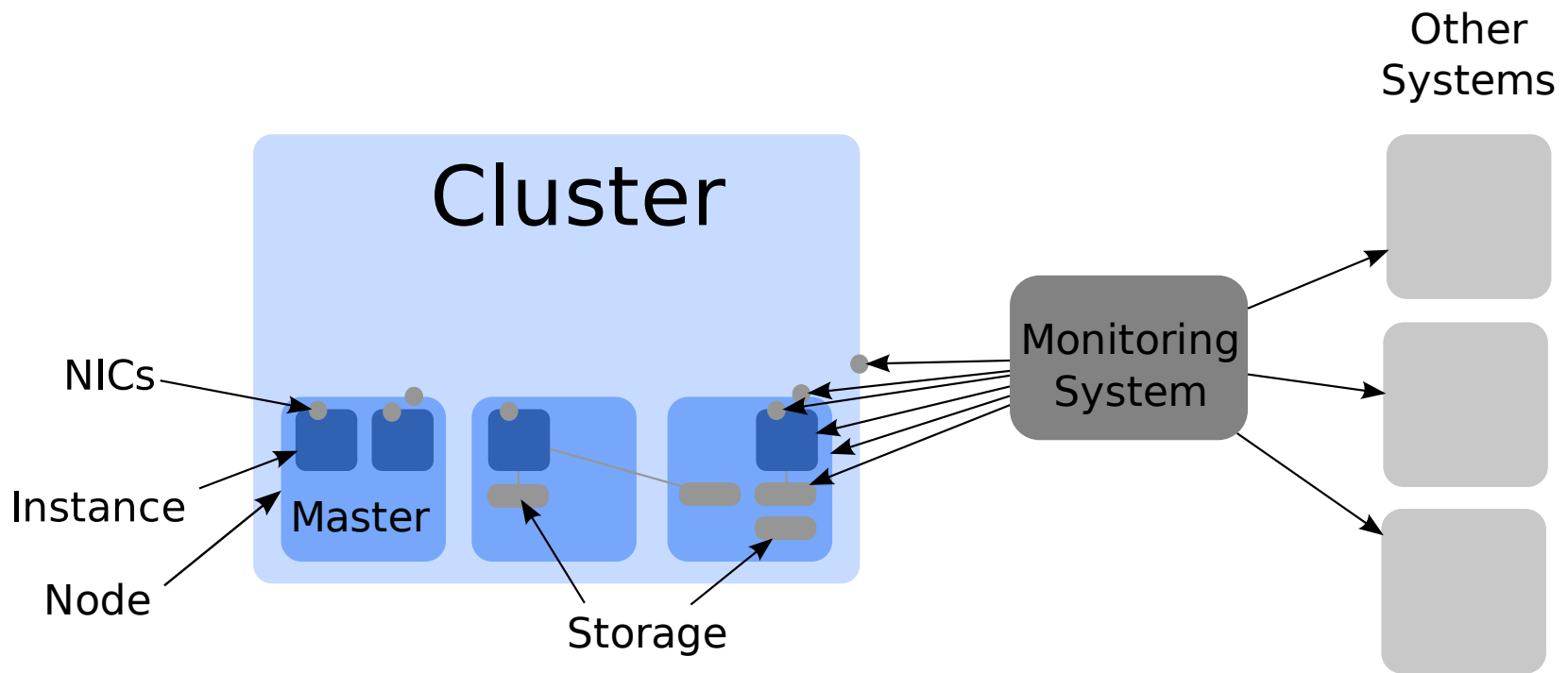
Historical Monitoring

Keep long history of utilization for capacity planning, budgeting, and troubleshooting.

- What to collect:
 - Hourly (or more) history of disk I/O, RAM and network utilization
 - Daily/weekly history of # nodes, # instances, # instance create/delete
 - Uptime statistics
- Troubleshooting:
 - Did problem appear when disk I/O reached a certain level?
 - Is the current network utilization heavy or light?
- Capacity and budget planning:
 - When will resources be exhausted? (don't be surprised)
 - Based on current growth, what do we need to buy?
 - How much money have we saved by using virtualization?
 - How many users/machines do we have?

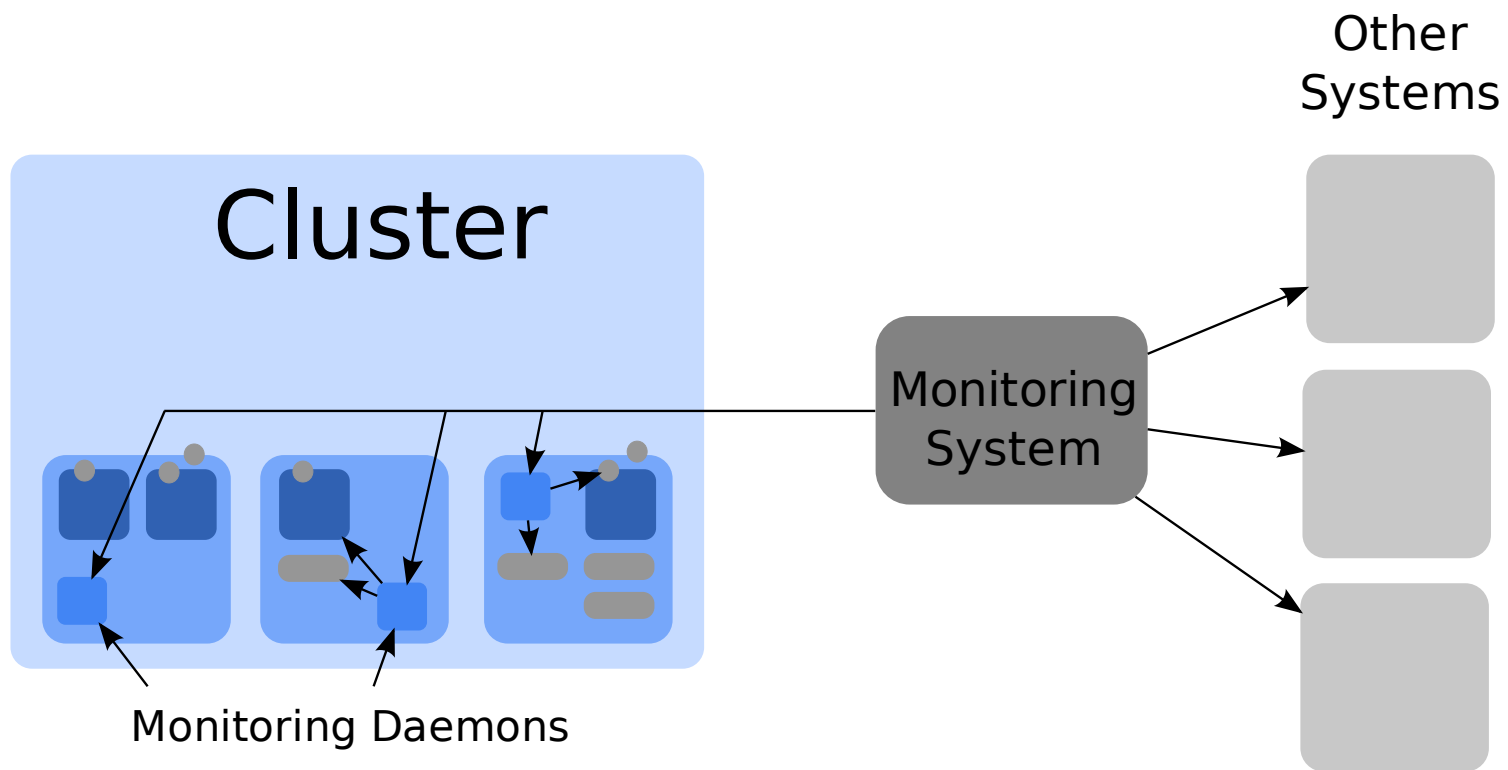
Monitoring a cluster

The old school way



Monitoring a cluster

Using the monitoring daemon



What is the monitoring daemon?

Provides information:

- about the cluster state
- about the cluster health
 - automatically computed
- live
- read-only

design doc: `design-monitoring-agent.rst`

How is the monitoring daemon?

- HTTP daemon
- Replying to REST-like queries
 - Actually, GET only
- Providing JSON replies
 - Easy to parse in any language
 - Already used in all the rest of Ganeti
- Running on every node (Not: only master-candidates, VM-enabled)
- Additionally: `mon-collector`: quick 'n dirty CLI tool

Data collectors

- provide data to the daemon
- one collector, one report
- one collector, one category:
 - storage, hypervisor, daemon, instance
- two kinds: performance reporting, status reporting
- new feature: stateful data collectors

Data collectors

What data can be retrieved right now?

Now:

- instance status (Xen only) (category: instance)
- diskstats information (storage)
- LVM logical volumes information (storage)
- DRBD status information (storage)
- Node OS CPU load average (no category, default)

Soon(-ish):

- instance status for KVM (instance)
- Ganeti daemons status (daemon)
- Hypervisor resources (hypervisor)
- Node OS resources report (default)

The report format

JSON

```
{
  "name" : "TheCollectorIdentifier",
  "version" : "1.2",
  "format_version" : 1,
  "timestamp" : 1351607182000000000,
  "category" : null,
  "kind" : 0,
  "data" : { "plugin_specific_data" : "go_here" }
}
```

- **name**: the name of the plugin. Unique string.
- **version**: the version of the plugin. A string.
- **format_version**: the version of the **data** format of the plugin. Incremental integer.
- **timestamp**: when the report was produced. Nanoseconds. Can be zero-padded.

Status reporting collectors: report

They introduce a mandatory part inside the **data** section.

JSON

```
"data" : {  
  ...  
  "status" : {  
    "code" : <value>  
    "message": "some summary goes here"  
  }  
}
```

- **<value>**: by increasing criticality level
 - 0: working as intended
 - 1: temporarily wrong. Being auto-repaired
 - 2: unknown. Potentially dangerous state
 - 4: problems. External intervention required

How to use the daemon?

- Accepts HTTP connections on `node.example.com:1815`
 - Not authenticated: read only
 - Just firewall, or bind on local address only
- GET requests to specific addresses
- Each address returns different info according to the API

```
/ (return the list of supported protocol version)
/1/list/collectors
/1/report/all
/1/report/[category]/[collector_name]
```

The reason trail

Introduction

- Initially required for the instance status (Xen) collector
 - Why did the instance last change its status?
 - Not just a message, but a complete track of what happened
 - More information available in the design doc: `doc/design-reason-trail.rst`

The reason trail

What format?

List of triples (source, reason, timestamp)

```
[("user", "Cleanup of unused instances", 1363088484000000000),  
 ("gnt:client:gnt-instance", "stop", 1363088484020000000),  
 ("gnt:opcode:shutdown", "job=1234;index=0", 1363088484026000000),  
 ("gnt:daemon:noded:shutdown", "", 1363088484135000000)]
```

PYTHON

- **source**: the entity deciding to perform/forward the command. Free form, but the **gnt:** prefix is reserved
- **reason**: why the entity decided to perform the operation
- **timestamp**: timestamp since epoch, in nanoseconds

The reason trail

How is it generated?

- Automatically, from RAPI/CLI down to opcode level
- Before opcode generation:
 - User message (now):
 - CLI: `--reason`
 - RAPI: `reason` parameter added to the request
- After opcode's job execution:
 - Specialized usages and manual implementations
 - Instance state change reason (start, stop, reboot. Serialized on file)

Thank You!

Questions?

Survey at <https://www.usenix.org/lisa13/training/survey>



- © 2010 - 2013 Google
- Use under GPLv2+ or CC-by-SA
- Some images borrowed / modified from Lance Albertson and Justin Pop
- Some slides were borrowed / modified from Tom Limoncelli

