Automatic deployment of the Network Weather Service using the Effective Network View

Arnaud Legrand and Martin Quinson

Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon

IPDPS 2004

Outline

The Network Weather Service overview

Overview

Goal: (Grid) system availabilities measurement and forecasting

Project from UCSB, used by AppLeS, Globus, NetSolve, Ninf, DIET, ...

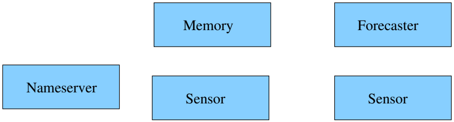
Architecture

Sensor: conducts measurements

Memory: stores the results

Forecaster: future tendencies (statistically)

Name server: directory service (like LDAP)



Distributed system

The Network Weather Service overview

Overview

Goal: (Grid) system availabilities measurement and forecasting

Project from UCSB, used by AppLeS, Globus, NetSolve, Ninf, DIET, ...

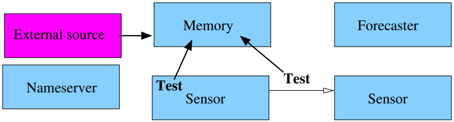
Architecture

Sensor: conducts measurements

Memory: stores the results

Forecaster: future tendencies (statistically)

Name server: directory service (like LDAP)



Steady state: regular tests

The Network Weather Service overview

Overview

Goal: (Grid) system availabilities measurement and forecasting

Project from UCSB, used by AppLeS, Globus, NetSolve, Ninf, DIET, ...

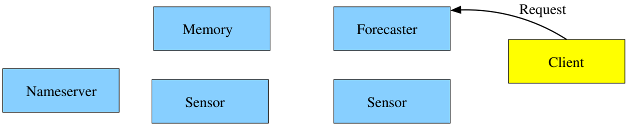
Architecture

Sensor: conducts measurements

Memory: stores the results

Forecaster: future tendencies (statistically)

Name server: directory service (like LDAP)



Handling of a request

The Network Weather Service overview

Overview

Goal: (Grid) system availabilities measurement and forecasting

Project from UCSB, used by AppLeS, Globus, NetSolve, Ninf, DIET, ...

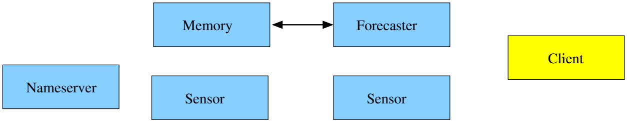
Architecture

Sensor: conducts measurements

Memory: stores the results

Forecaster: future tendencies (statistically)

Name server: directory service (like LDAP)



Handling of a request

The Network Weather Service overview

Overview

Goal: (Grid) system availabilities measurement and forecasting

Project from UCSB, used by AppLeS, Globus, NetSolve, Ninf, DIET, ...

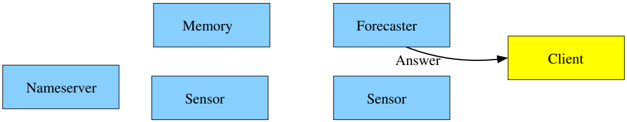
Architecture

Sensor: conducts measurements

Memory: stores the results

Forecaster: future tendencies (statistically)

Name server: directory service (like LDAP)



Handling of a request

NWS measurements and forecasting

Provided metrics

bandwidthTcp, latencyTcp (Default: 64Kb in 16Kb messages; buffer=32Kb), availableCpu (for an incoming process), currentCpu (for existing processes), connectTimeTcp, freeDisk, freeMemory, ...

Statistical forecasting

Selection of the best statistical method (mean, median, gradian, last value, ...)

Data = serie: $D_1, D_2, \ldots, D_{n-1}, D_n$. We want D_{n+1}.
Methods are applied on $D_1, D_2, \ldots, D_{n-1}$. each one predict D_n.
Selection of the best on D_n to predict D_{n+1}.

NWS configuration & deployment

Deployment requirements

Correction **Do not let experiments interfere.**
Two test packets on same link ⇒ each report half of bandwidth
Clique: set on which tests are done in a mutually exclusive manner

Scalability **Keep cliques small** for sufficient frequency and reactivity.

Completeness **Estimate each host pair**
⇒ aggregation when lacking direct measurement
⇒ cliques should follow sub-net tilling

Intrusiveness **Conduct only needed test** One pair is representative on a hub

Configuring NWS is a difficult task

Need to know both **the tool** and **the topology** (link with potential collisions).

NWS configuration & deployment

Deployment requirements

Correction **Do not let experiments interfere.**
Two test packets on same link ⇒ each report half of bandwidth
Clique: set on which tests are done in a mutually exclusive manner

Scalability **Keep cliques small** for sufficient frequency and reactivity.

Completeness **Estimate each host pair**
⇒ aggregation when lacking direct measurement
⇒ cliques should follow sub-net tilling

Intrusiveness **Conduct only needed test** One pair is representative on a hub

Configuring NWS is a difficult task

Need to know both **the tool** and **the topology** (link with potential collisions).

NWS configuration & deployment

Deployment requirements

Correction **Do not let experiments interfere.**
Two test packets on same link ⇒ each report half of bandwidth
Clique: set on which tests are done in a mutually exclusive manner

Scalability **Keep cliques small** for sufficient frequency and reactivity.

Completeness **Estimate each host pair**
⇒ aggregation when lacking direct measurement
⇒ cliques should follow sub-net tilling

Intrusiveness **Conduct only needed test** One pair is representative on a hub

Configuring NWS is a difficult task

Need to know both **the tool** and **the topology** (link with potential collisions).

NWS configuration & deployment

Deployment requirements

Correction **Do not let experiments interfere.**
Two test packets on same link ⇒ each report half of bandwidth
Clique: set on which tests are done in a mutually exclusive manner

Scalability **Keep cliques small** for sufficient frequency and reactivity.

Completeness **Estimate each host pair**
⇒ aggregation when lacking direct measurement
⇒ cliques should follow sub-net tilling

Intrusiveness **Conduct only needed test** One pair is representative on a hub

Configuring NWS is a difficult task

Need to know both **the tool** and **the topology** (link with potential collisions).

NWS configuration & deployment

Deployment requirements

Correction **Do not let experiments interfere.**
Two test packets on same link ⇒ each report half of bandwidth
Clique: set on which tests are done in a mutually exclusive manner

Scalability **Keep cliques small** for sufficient frequency and reactivity.

Completeness **Estimate each host pair**
⇒ aggregation when lacking direct measurement
⇒ cliques should follow sub-net tilling

Intrusiveness **Conduct only needed test** One pair is representative on a hub

Configuring NWS is a difficult task

Need to know both **the tool** and **the topology** (link with potential collisions).

The Effective Network View mapping solution

Overview

Goal: Mapping the network topology

Authors: Gary Shao *et Al* (UCSD)

Motivation: Master/slave scheduling

Methodology: Active interference tests

Related work

Method	Restricted	Focus	Routers	Notes
SNMP	authorized	path	all	passive, LAN
traceroute	ICMP	path	all	level 3 of OSI
pathchar	root	path	all	link bandwidth, slow
Other tomography	no	path	$d_{in} \neq d_{out}$	tree bipartite [Rabbat03]
ENV	no	interference	some	tree only

Mapping algorithm (1/4)

Naive algorith

For all hosts (a, b, c, d), measure:

$bw(ab)$: bandwidth on (ab)

$bw_{/\!/ cd}(ab)$: idem when (cd) is saturated

Interference if $\frac{bw_{/\!/ cd}(ab)}{bw(ab)} \simeq 0.5$

Naivety is a bad habit

Network stabilization

$\Rightarrow$ 2 tests per minutes

$\Rightarrow$ 50 days for 20 hosts

ENV algorithm

- Tree view: Interferences between streams from a *master* node to any
  $\Rightarrow$ from $O(n^4)$ to $O(n^3)$
- Various other optimizations to reduce the number of tests

Mapping algorithm (1/4)

Naive algorith

For all hosts (a, b, c, d), measure:

$bw(ab)$: bandwidth on (ab)

$bw_{/\!/ cd}(ab)$: idem when (cd) is saturated

Interference if $\frac{bw_{/\!/ cd}(ab)}{bw(ab)} \simeq 0.5$

Naivety is a bad habit

Network stabilization

$\Rightarrow$ 2 tests per minutes

$\Rightarrow$ 50 days for 20 hosts

ENV algorithm

- Tree view: Interferences between streams from a *master* node to any
  $\Rightarrow$ from $O(n^4)$ to $O(n^3)$
- Various other optimizations to reduce the number of tests

Mapping algorithm (1/4)

Naive algorith

For all hosts (a, b, c, d), measure:

$bw(ab)$: bandwidth on (ab)

$bw_{/\!/ cd}(ab)$: idem when (cd) is saturated

Interference if $\frac{bw_{/\!/ cd}(ab)}{bw(ab)} \simeq 0.5$

Naivety is a bad habit

Network stabilization

$\Rightarrow$ 2 tests per minutes

$\Rightarrow$ 50 days for 20 hosts

ENV algorithm

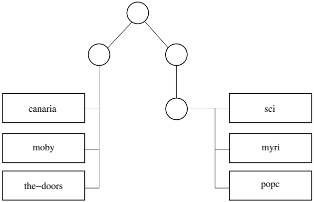
- Tree view: Interferences between streams from a *master* node to any
  $\Rightarrow$ from $O(n^4)$ to $O(n^3)$
- Various other optimizations to reduce the number of tests

Algorithm (2/4): master-independent data collection

Structural topology

Topology first guess:

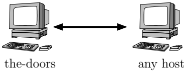
1. Each node traceroute to an external location
2. Merging results gives a tree



Algorithm (3/4): master-dependent data collection

Successive refinements of the topology

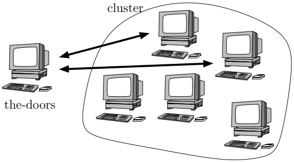
- **Host to host bandwidth**
  split out machines having different
  bandwidth to the master
- **Pairwise host bandwidth**
  measure bandwidth concurrently
  compare to previous step
  split cluster if transfers independent
- **Internal cluster bandwidth**
- **Jammed bandwidth**
  $\frac{bw_{/\!/ bc}(Ma)}{bw(Ma)} \simeq 0.5 \Longrightarrow$ internal network shared



Algorithm (3/4): master-dependent data collection

Successive refinements of the topology

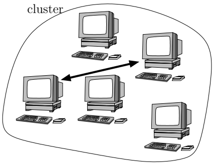
- **Host to host bandwidth**
  split out machines having different
  bandwidth to the master
- **Pairwise host bandwidth**
  measure bandwidth concurrently
  compare to previous step
  split cluster if transfers independent
- **Internal cluster bandwidth**
- **Jammed bandwidth**
  $\frac{bw_{/\!/ bc}(Ma)}{bw(Ma)} \simeq 0.5 \Longrightarrow$ internal network shared



Algorithm (3/4): master-dependent data collection

Successive refinements of the topology

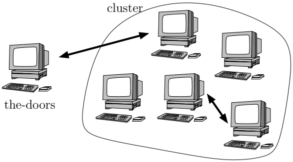
- **Host to host bandwidth**
  split out machines having different
  bandwidth to the master
- **Pairwise host bandwidth**
  measure bandwidth concurrently
  compare to previous step
  split cluster if transfers independent
- **Internal cluster bandwidth**
- **Jammed bandwidth**
  $\frac{bw_{/\!/bc}(Ma)}{bw(Ma)} \simeq 0.5 \Longrightarrow$ internal network shared



Algorithm (3/4): master-dependent data collection

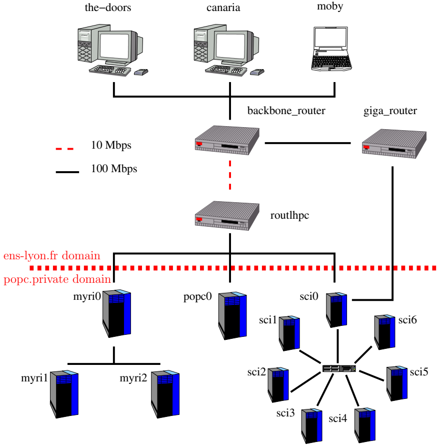
Successive refinements of the topology

- **Host to host bandwidth**
  split out machines having different
  bandwidth to the master
- **Pairwise host bandwidth**
  measure bandwidth concurrently
  compare to previous step
  split cluster if transfers independent
- **Internal cluster bandwidth**
- **Jammed bandwidth**
  $\frac{bw_{/\!/ bc}(Ma)}{bw(Ma)} \simeq 0.5 \Longrightarrow$ internal network shared

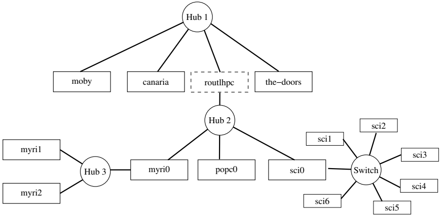


Algorithm (4/4): Result on the ENS-Lyon network

Physical topology



Effective topology from the-doors



Summary about ENV

Tradeoffs

Master/Slave: tree view only (price for efficiency?)

Intrusiveness: inject large amount of traffic (price for simplicity?)

Known problems

Asymmetric routes: not taken into account (yet?)

Makes mapping faster, but such inconsistencies are common

Open questions

Reliability and accuracy:

- Platform evolution ($\Rightarrow$ mapping speed)
- Experimental thresholds (empirically determined)

Summary about ENV

Tradeoffs

Master/Slave: tree view only (price for efficiency?)

Intrusiveness: inject large amount of traffic (price for simplicity?)

Known problems

Asymmetric routes: not taken into account (yet?)

Makes mapping faster, but such inconsistencies are common

Open questions

Reliability and accuracy:

- Platform evolution ($\Rightarrow$ mapping speed)
- Experimental thresholds (empirically determined)

Summary about ENV

Tradeoffs

Master/Slave: tree view only (price for efficiency?)

Intrusiveness: inject large amount of traffic (price for simplicity?)

Known problems

Asymmetric routes: not taken into account (yet?)

Makes mapping faster, but such inconsistencies are common

Open questions

Reliability and accuracy:

- Platform evolution ($\Rightarrow$ mapping speed)
- Experimental thresholds (empirically determined)

Deployment design

Typical Grid testbeds are constellation of trees ⇒ hierarchical monitoring
Manually: 2 levels (inter-organization vs intra-organization)
Here: N levels (one per group)

Bottom-up algorithm along the tree
shared group (hub): every pair is representative of internal connectivity

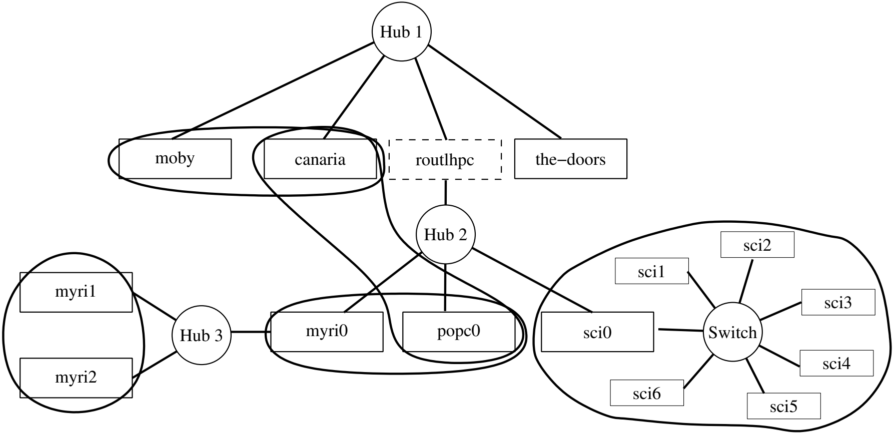
- ⇒ Form a clique with two arbitrarily chosen hosts
- ☹ NWS cannot substitute a pair with the chosen one, must be application level

not shared group (switch): transfers interfere only if same host in both
($AB \not\parallel CD \Leftrightarrow \{AB\} \cap \{CD\} = \emptyset$)

- ⇒ Host-based locking needed (but not supported by NWS)
- ⇒ Form a clique with all hosts (ensure validity, deteriorate frequency)

Example on the ENS-Lyon network

Result of the algorithm in our case



How to apply the configuration once computed?

Previous (NWS) solution

`ssh` to each host; pass options to daemons on command-line

Our solution

- Make a global configuration file; dispatch it using *e.g.* NFS
- Manager script on each host to apply it (after `ssh`)

Ease platform startup and shutdown

Future

Watchdog, or real management solution (JINI)
would allow error detection and recovery

Conclusion

- NWS is the *de-facto* standard for Grid availability monitoring
- Ensuring correction, scalability, completeness, limiting intrusiveness requires topology knowledge (interferences: potential collisions)
- ENV provides an interference-focused network mapping
- Those informations sufficient for an efficient configuration planning

Open questions & future work

About ENV:

- Asymmetric routes + tree limitation
- Automatic threshold discovery

About NWS:

- Host-based locking
- Lookup: aggregation, substitute pairs

Automatic deployment of NWS using ENV:

- Quantify quality of configuration (simulator)
- Platform evolutions
- Real management solution

Conclusion

- NWS is the *de-facto* standard for Grid availability monitoring
- Ensuring correction, scalability, completeness, limiting intrusiveness requires topology knowledge (interferences: potential collisions)
- ENV provides an interference-focused network mapping
- Those informations sufficient for an efficient configuration planning

Open questions & future work

About ENV:

- Asymmetric routes + tree limitation
- Automatic threshold discovery

About NWS:

- Host-based locking
- Lookup: aggregation, substitute pairs

Automatic deployment of NWS using ENV:

- Quantify quality of configuration (simulator)
- Platform evolutions
- Real management solution