Slide 0

echoping, mesurer les applications

Stéphane Bortzmeyer

AFNIC

⟨bortz@users.sourceforge.net⟩

4 octobre 2004

Slide 1

Mesurer les applications

- On mesure souvent le temps de réponse du réseau (RTT) avec ping
- Mais celui des applications peut être très différent
  1. Unix : traitement par le noyau ou par une application qui *swappe*
  2. Délais spécifiques à l'application (contacter la base de données...)
  3. Coupe-feux qui bloquent ICMP
  4. *Layer-4 or Layer-7 switches* qui traitent différemment ICMP et TCP

Slide 2

Limite des outils classiques

Certains outils n'ont pas de mesure du RTT : on les emballe avec `time`. Inconvénients :

1. On mesure le temps de résolution DNS,
2. On mesure le temps de lancement du programme (s'il est en Perl, cela fausse tout)...

Certains outils ont la mesure de RTT (`dig`). Mais pas de possibilité de répéter le test donc, il faut utiliser `awk` si on veut des statistiques.

Exemple avec dig, le script qtest, qui détermine le serveur le plus rapide (BIND utilise un algorithme analogue pour savoir quel serveur d'un domaine interroger).

```
#!/bin/sh
#
# From: Joe Abley <jabley@isc.org>

query=$1; shift
#
[ -z "$*" ] && echo "Syntax: $0 query server..." && exit 1
#
for i in 0 1 2; do
   for server in $*; do
     dig @${server} ${query};
   done
done | \
   awk '/^;; Query time:/ { query_time = $4; } \
     /^;; SERVER: / { sum[$3] += query_time; num[$3]++; } \
     END { for (server in sum) { print int(sum[server]/num[server]), server; } }' | \
   sort -n | head -1
```

Slide 3

echoping

`http://echoping.sourceforge.net/`

9 ans d'utilisation aujourd'hui.

A surtout décollé dans le cadre du projet Renater-Cache (mesurer le temps de réponse du cache national, par exemple avant et après un réglage).

Mesurer le RTT d'une requête applicative. Le chrono est lancé **après** la résolution DNS.

Tourne sur tout Unix, en ligne de commandes, pour utilisation par des scripts (mon, Nagios, MRTG, Smokeping).

Smokeping (`http://www.smokeping.org/`) est de loin le plus gros utilisateur d'echoping.

Slide 4

Protocoles utilisés

1. Au début, echo, discard et chargen
2. Puis HTTP (Renater-Cache), toujours le plus utilisé
3. Demain, DNS, whois, PostgreSQL, LDAP...

Slide 5

Répétition des tests

Comme ping, on peut répéter les tests et avoir des statistiques.

```
% echoping -h 10 -h / webmail.nic.af
...
Minimum time: 1.799637 seconds (142 bytes per sec.)
Maximum time: 16.494219 seconds (16 bytes per sec.)
Average time: 3.356738 seconds (76 bytes per sec.)
Standard deviation: 3.730219
Median time: 2.003397 seconds (128 bytes per sec.)
```

Slide 6

Moyenne et médiane

Sur l'Internet, ou sur un serveur chargé, la moyenne est peu significative : un seul test très lent la change beaucoup.

La médiane est en général plus informative.

```
Elapsed time: 3.030966 seconds
Elapsed time: 122.533918 seconds
Elapsed time: 2.394198 seconds
Elapsed time: 2.012312 seconds
Elapsed time: 1.432407 seconds
...
---
Minimum time: 1.432407 seconds (179 bytes per sec.)
Maximum time: 122.533918 seconds (2 bytes per sec.)
Average time: 11.393709 seconds (22 bytes per sec.)
Standard deviation: 26.072556
Median time: 2.712582 seconds (94 bytes per sec.)
```

Slide 7

Les greffons

Beaucoup d'utilisateurs réclamaient le support de tel ou tel protocole.

Souvent, ils envoyaient le *patch*.

Le code devenait de plus en plus complexe. Et dépendant de bibliothèques extérieures.

echoping 6 introduira donc les greffons (*plug-ins*), écrits à part et chargés dynamiquement.

Slide 8

Utiliser le greffon DNS

```
% echoping -v -m dns ns2.nic.fr -t NS nic.fr
This is echoping, version 6.0-BETA.

Running start() for the plugin dns.so...
Trying to call plugin dns.so for internet address 192.134.0.4 53...
Elapsed time: 0.001717 seconds
```

Slide 9

Utiliser le greffon PostgreSQL

```
% echoping -v -m postgresql localhost -c 'dbname=essais' \
        'SELECT * FROM Foobar'

This is echoping, version 6.0-BETA.

Running start() for the plugin postgresql.so...
Trying to call plugin postgresql.so...
3 tuples returned
Elapsed time: 0.058392 seconds
```

Slide 10

API des greffons

Les greffons se programment en C

- `char * init (const int argc, const char **argv, const echoping_options options)` Initialise
- `void start (struct addrinfo *res)` Se connecte
- `int execute ()` Le chronomètre est lancé juste avant
- `void terminate ()`

Exemple avec le greffon whois, le plus simple vue la simplicité du protocole (RFC 3912[1]) :

```
char *
init (const int argc, const char **argv, echoping_options global_options)
{
  /* Analyse les arguments */
  return "nicname";
}

void
start (struct addrinfo *res)
{
  whois_server = *res;
}

int
execute ()
{
  int nr = 0;
  char recvline[MAX_LINE + 1];
  char complete_request[MAX_REQUEST];
  if ((sockfd =
       socket (whois_server.ai_family, whois_server.ai_socktype,
       whois_server.ai_protocol)) < 0)
    err_sys ("Can't open socket");
  if (connect (sockfd, whois_server.ai_addr, whois_server.ai_addrlen) < 0)
    err_sys ("Can't connect to server");
  if ((files = fdopen (sockfd, "r")) == NULL)
    err_sys ("Cannot fdopen");
  sprintf (complete_request, "%s\r\n", request);
  n = strlen (complete_request);
  if (writen (sockfd, complete_request, n) != n)
    err_sys ("writen error on socket");
  /* Read from the server */
  while ((nr = readline (files, recvline, MAX_LINE, 0)) > 0)
    if (dump)
      printf ("%s", recvline);
  if (dump)
    printf ("\n");
  close (sockfd);
  return 1;
}

void
terminate ()
{
}
```

[1] Pour voir le RFC de numéro NNN, `http://www.ietf.org/rfc/rfcNNN.txt`, par exemple `http://www.ietf.org/rfc/rfc3912.txt`