

Systèmes de fichiers distribué : comparaison de GlusterFS, MooseFS et Ceph avec déploiement sur la grille de calcul Grid'5000.

Jean-François Garçia, Florent Lévigne,
Maxime Douheret, Vincent Claudel

Table des matières

1	Introduction	3
1.1	Contexte	3
1.2	Système de fichiers distribué	3
1.3	Le Grid'5000	3
2	NFS	4
2.1	Présentation	4
3	GlusterFs	5
3.1	Présentation	5
4	MooseFS	6
4.1	Présentation	6
4.1.1	Présentation générale	6
4.1.2	Aspect technique	6
4.2	Mise en place	7
4.2.1	Environnement logiciel	7
5	Ceph	8
5.1	Présentation	8
6	Comparaison	9
6.1	Test de performances	9
7	Conclusion	10
A	Répartition des tâches	11
A.1	Florent Lévine	11
A.2	Jean-François Garcia	11
B	Scripts	12
B.1	GlusterFs	12
B.2	MooseFs	14
B.3	Ceph	14
B.4	NFS	16
B.5	Benchmark	17

Partie 1

Introduction

1.1 Contexte

Étudiants en licence professionnelle ASRALL (Administration de Systèmes, Réseaux, et Applications à base de Logiciels libres), notre formation prévoit une période de x mois à mi-temps pour la réalisation d'un projet tuteuré.

Le projet que nous avons choisi consiste à comparer diverses solutions de systèmes de fichiers distribués.

1.2 Système de fichiers distribué

Un système de fichiers (file system en anglais) est une façon de stocker des informations et de les organiser dans des fichiers, sur des périphériques comme un disque dur, un CD-ROM, une clé USB, etc. Il existe de nombreux systèmes de fichiers (certains ayant des avantages sur d'autres), dont entre autres l'ext (Extented FS), le NTFS (New Technology FileSystem), ZFS (Zettabyte FS), FAT (File Allocation Table).

Un système de fichiers distribué est un système de fichiers permettant le partage de données à plusieurs clients au travers du réseau. Contrairement à un système de fichier local, le client n'a pas accès au système de stockage sous-jacent, et interagit avec le système de fichier via un protocole adéquat.

Un système de fichier distribué est donc utilisé par plusieurs machines en même temps (les machines peuvent ainsi avoir accès à des fichiers distants, l'espace de noms est mis en commun). Un tel système permet donc de partager des données entre plusieurs clients, et pour certains de répartir la charge entre plusieurs machines, et de gérer la sécurité des données (par réplication)

1.3 Le Grid'5000

Partie 2

NFS

2.1 Présentation

NFS (Network File System, système de fichiers en réseau en français) est un système de fichiers développé par Sun Microsystems, permettant de partager des données par le réseau.

Partie 3

GlusterFs

3.1 Présentation

Partie 4

MooseFS

4.1 Présentation

4.1.1 Présentation générale

MooseFS (Moose File System) est un système de fichiers répartis à tolérance de panne, développé par Gemius SA. Le code préalablement propriétaire a été libéré et mis à disposition publiquement le 5 mai 2008. Il permet de déployer assez facilement un espace de stockage réseau, réparti sur plusieurs serveurs.

Cette répartition permet de gérer la disponibilité des données, lors des montées en charge ou lors d'incident technique sur un serveur. L'atout principal de MooseFS, au delà du fait qu'il s'agisse d'un logiciel libre, est sa simplicité de mise en œuvre.

En effet le tutoriel de prise en main, disponible sur le site du projet, explique de manière claire comment mettre en place une architecture distribuée en quelques heures. Concernant les utilisations, elles sont multiples et surtout, après la phase de configuration, l'évolution du système est très simple. L'ajout de serveurs, d'espace disque peuvent être gérés très facilement.

4.1.2 Aspect technique

Les montages du système de fichiers par les clients se font à l'aide de FUSE.

MooseFS est constitué de trois types de serveurs :

1. Un serveur de métadonnées (MDS)
Ce serveur gère la répartition des différents fichiers
2. Un serveur métajournal (Metalogger server)
Ce serveur récupère régulièrement les métadonnées du MDS et les stocke en tant que sauvegarde.
3. Des serveurs Chunk (CSS)
Ce sont ces serveurs qui stockent les données des utilisateurs.

Le point le plus important étant de bien dimensionner le serveur Master (qui stocke les métadonnées) afin de ne pas être limité par la suite. Donc pour ceux qui ne peuvent pas mettre en place des systèmes de stockage réseaux propriétaires assez coûteux, je vous conseille d'étudier cette possibilité. Elle vous permettra de partager des données sur plusieurs machines, de manière rapide, fiable, sécurisée et surtout peu coûteuse.

4.2 Mise en place

4.2.1 Environnement logiciel

Le système utilisé est une Debian Squeeze. MosseFs ne faisant pas partie des dépôts de la distribution, nous avons compilé le paquet à partir des sources dans leurs dernières version (1.6.20).

Partie 5

Ceph

5.1 Présentation

Partie 6

Comparaison

6.1 Test de performances

Afin de ne pas avoir de différence de matériel lors nos test, ceux-ci ont tous été réalisés sur un même cluster du Grid'5000 : Graphene.

Ce cluster est composé de 144 noeuds, avec pour caractéristique :

- 1 CPU Intel de quatre cœurs cadencé à 2.53 GHz
- 16 Go de RAM
- 278 Go d'espace disque

Nous avons réalisé un benchmark mesurant les performances (débit) de quatre type d'opérations sur le système de fichier distribué :

Écriture de petits fichiers : écriture des sources du noyau linux (décompressé).

Écriture de gros fichiers : écriture d'un fichier de 3 Go¹.

Lecture de petits fichiers : lecture des fichiers du noyau linux. Pour cela, nous avons compressé le dossier contenant le noyau (impliquant la lecture des fichiers), en redirigeant la sortie vers /dev/nul (afin que les performances du disque ne rentrent pas en jeux).

Lecture de gros fichiers : lecture du fichier de 3 Go. Opération réalisé en faisant un "cat" du fichier, et en redirigeant la sortie vers /dev/nul afin de ne pas "polluer" le terminal.

1. Fichier créé avec la commande : `dd if=/dev/zero of=/lieu/voulu bs=1G count=3`

Partie 7

Conclusion

Partie A

Répartition des tâches

A.1 Florent Lévigne

- Étude sur la mise en place de GlusterFS
- Réalisation d'un script de déploiement de GlusterFS
- Étude sur la mise en place de MooseFS
- Réalisation d'un script de déploiement de MooseFS
- Réalisation d'un script de benchmark pour système de fichiers distribué

A.2 Jean-François Garcia

- Étude sur la mise en place de GlusterFS
- Étude sur la mise en place de NFS
- Réalisation d'un script de déploiement de NFS
- Étude sur la mise en place de CephFS
- Réalisation d'un script de déploiement de CephFS

Partie B

Scripts

B.1 GlusterFs

Fichier deploymentGluster.rb :

```
1 #!/usr/bin/ruby -w
2 # encoding: utf-8
3
4 # r s e r v a t i o n  d e s  n o e u d s  ( a  l a n c e r  m a n u e l l e m e n t )
5 # o a r s u b  -I  -t  d e p l o y  -l  n o d e s = 8 , w a l l t i m e = 2
6 # o a r s u b  -I  -t  d e p l o y  -l  n o d e s = 8 , w a l l t i m e = 2  -p  " c l u s t e r = ' g r a p h e n e ' "
7
8 i f  A R G V [ 0 ]  = =  n i l
9   p u t s  " d o i t  _ p r e n d r e  _ e n  _ p a r a m e t r e  _ l e  _ n o m b r e  _ d e  _ s e r v e u r s "
10   e x i t ( 1 )
11 e n d
12
13
14 # d o i t  c o n c o r d e r  a v e c  l a  c o m m a n d e  o a r s u b
15 n u m b e r O f C l i e n t s  =  5  #  i n u t i l e  :  p r e n d  l e s  m a c h i n e s  d i s p o  r e s t a n t e s  c o m m e  c l i e n t s
16 n u m b e r O f S e r v e r s  =  " #{ A R G V [ 0 ] } ". t o _ i
17
18 i n f i n i b a n d  =  1  #  1  :  a c t i v  ,  0  :  n o n  a c t i v  ( n e  c h a n g e  r i e n  p o u r  l ' i n s t a n t )
19
20 # c r a t i o n  d ' u n  f i c h i e r  c o n t e n a n t  l a  l i s t e  d e s  n o e u d s  r s e r v s
21 ' t o u c h  l i s t O f N o d e s '
22 F i l e . o p e n ( " l i s t O f N o d e s " , ' w ' )  d o  | f i l e |
23   f i l e  <<  ' c a t  $ O A R _ F I L E _ N O D E S  | s o r t  - u '
24 e n d
25
26 # c r a t i o n  d e  d e u x  f i c h i e r s  c o n t e n a n t  l a  l i s t e  d e s  s e r v e u r s ,  e t  d e s  c l i e n t s
27 ' t o u c h  l i s t O f C l i e n t s  l i s t O f S e r v e r s '
28 s e r v e r W r i t e d  =  0
29 F i l e . o p e n ( " l i s t O f N o d e s " , ' r ' )  d o  | n o d e |
30   F i l e . o p e n ( " l i s t O f S e r v e r s " , ' w ' )  d o  | s e r v e r |
31     F i l e . o p e n ( " l i s t O f C l i e n t s " , ' w ' )  d o  | c l i e n t |
32       w h i l e  l i n e  =  n o d e . g e t s
33         i f  s e r v e r W r i t e d  <  n u m b e r O f S e r v e r s
34           s e r v e r  <<  l i n e
35           s e r v e r W r i t e d  +=  1
36         e l s e
37           c l i e n t  <<  l i n e
38         e n d
39     e n d
end
```

```

40     end
41 end
42 end
43
44 # d ploiement des machines
45 puts "Machines_en_cours_de_d ploiement..."
46 'kadeploy3 -k -e squeeze-collective -u flevisne -f listOfNodes' # image collective
47
48
49 # Envoie d'un script de cr ation d'un r pertoire dans /tmp/sharedspace sur les
    serveurs
50 File.open("listOfServers", 'r') do |file|
51     while line = file.gets
52         machine = line.split.join("\n")
53         'ssh root@#{machine} < createFolders.sh'
54     end
55 end
56
57 # Envoie d'un script de cr ation d'un r pertoire dans /media/glusterfs sur les
    clients
58 File.open("listOfClients", 'r') do |file|
59     while line = file.gets
60         machine = line.split.join("\n")
61         'ssh root@#{machine} < createMountDirectory.sh'
62     end
63 end
64
65 masterServer = 'head -n 1 listOfServers'.split.join("\n")
66
67 # g n ration des fichiers de conf, et envoie des fichiers de conf aux machines (
    serveurs et clients)
68 puts "Configuration_des_serveurs_et_des_clients..."
69 'scp listOfServers root@#{masterServer}:'
70 'scp listOfClients root@#{masterServer}:'
71 'scp glusterfs-volgen.rb root@#{masterServer}:'
72 'ssh root@#{masterServer} ./ glusterfs-volgen.rb'
73 # 'ssh root@#{masterServer} < execScript/ex-glusterfs-volgen.sh'
74
75 # d marriage des serveurs
76 puts "D marriage_des_serveurs..."
77 File.open("listOfServers", 'r') do |file|
78     while line = file.gets
79         machine = line.split.join("\n")
80         'ssh root@#{machine} < startGluster.sh'
81     end
82 end
83
84 # montage du r pertoire par les clients
85 puts "Montage_du_r pertoire_par_les_clients..."
86 File.open("listOfClients", 'r') do |file|
87     while line = file.gets
88         machine = line.split.join("\n")
89         'ssh root@#{machine} < mountFs.sh'
90     end
91 end
92

```

```

93 # r sum des machines
94 puts "GlusterFS_op rationnel"
95 puts "\nMachines_clients_:"
96 puts `cat listOfClients `
97
98 puts "\nMachines_serveurs_:"
99 puts `cat listOfServers `
100
101 puts "\nServeur_maitre_:_{masterServer}"
102
103 # nettoyage
104 #`rm listOfNodes listOfClients listOfServers `

```

B.2 MooseFs

B.3 Ceph

Fichier deploimentCeph.rb :

```

1  #!/usr/bin/ruby -w
2  # encoding: utf-8
3
4  #####
5
6  # File Name : deploimentCeph.rb
7
8  # Purpose :
9
10 # Creation Date : 11-03-2011
11
12 # Last Modified : jeu. 17 mars 2011 14:54:51 CET
13
14 # Created By : Helldar
15
16 #####
17
18 # doit concorder avec la commande oarsub
19
20 if ARGV[0] != nil
21   numberOfServers = ARGV[0].to_i
22   puts "Nb_serveur_:_{numberOfServers}\n"
23 else
24   puts "Veuillez_relancer_le_script_avec_les_bons_param_tres!\n"
25   exit
26 end
27
28 # cr ation d'un fichier contenant la liste des noeuds r serv s
29 `touch listOfNodes `
30 File.open("listOfNodes", 'w') do |file|
31   file << `cat $OAR_FILE_NODES | sort -u`
32 end
33 # cr ation de deux fichiers contenant la liste des serveurs, et des clients
34
35 `touch listOfClients listOfServers `

```

```

36 serverWrited = 0
37 File.open("listOfNodes", 'r') do |node|
38     File.open("listOfServers", 'w') do |server|
39         File.open("listOfClients", 'w') do |client|
40             while line = node.gets
41                 if serverWrited < numberOfServers
42                     server << line
43                     serverWrited += 1
44                 else
45                     client << line
46                 end
47             end
48         end
49     end
50 end
51
52 # d ploiement des machines
53 #puts "Machines en cour de d ploiement..."
54 #`kadeploy3 -k -e squeeze-collective -u flevigne -f listOfNodes` # image collective
55
56 # configuration du serveur
57 serveur_1 = `head -1 listOfServers | cut -d "." -f1`.strip
58 ip_serveur = `ssh root@#{serveur_1} hostname -i`.strip
59
60 # g n ration du fichier de ceph.conf
61
62 `touch ceph.conf`
63 File.open("ceph.conf", 'w') do |file|
64     file << "[global]
65     _____pid_file=_/var/run/ceph/$name.pid
66     _____debug_ms=_1
67     _____keyring=_/etc/ceph/keyring.bin
68     [mon]
69     _____mon_data=_/tmp/partage/mon$id
70     [mon0]
71     _____host=_#{serveur_1}
72     _____mon_addr=_#{ip_serveur}:6789
73     [mds]
74     _____debug_mds=_1
75     _____keyring=_/etc/ceph/keyring.$name"
76     if numberOfServers > 3
77         1.upto(3) { |i| file << "
78     [mds#{i-1}]"
79         host = `sed -n #{i + 1}p listOfServers | cut -d '.' -f1`.strip
80         file << "
81     _____#{host}" }
82     else
83         file << "[mds0]"
84         host = `sed -n 2p listOfServers | cut -d '.' -f1`.strip
85         file << "
86     _____#{host}"
87     end
88     file << "
89     [osd]
90     _____sudo=_true
91     _____osd_data=_/tmp/partage/osd$id

```

```

92 | #####keyring==_/etc/ceph/keyring.$name
93 | #####debug_osd==1
94 | #####debug_filstore==1
95 | #####osd_journal==_/tmp/partage/osd$id/journal
96 | #####osd_journal_size==1000"
97 | 1.upto(numberOfServers - 1) { |i| file << "
98 | [osd#{i-1}]}"
99 |     host = `sed -n #{i + 1}p listOfServers | cut -d '.' -f1 '.strip
100 |     file << "
101 | ######{host}" }
102 | end
103 | # copie du fichier ceph.conf vers le serveur
104 | `scp ceph.conf root@#{serveur_1}:/etc/ceph`
105 | puts "Envoy !"
106 | # g n ration du fichier keyring.bin
107 | `ssh root@#{serveur_1} cauthtool --create-keyring -n client.admin --gen-key keyring
108 | .bin`
109 | `ssh root@#{serveur_1} cauthtool -n client.admin --cap mds 'allow' --cap osd 'allow
110 | *' --cap mon 'allow rwx' keyring.bin`
111 | `ssh root@#{serveur_1} mv keyring.bin /etc/ceph/`
112 | puts "Keyring_g n r !"
113 | # montage
114 | `ssh root@#{serveur_1} mount -o remount,user_xattr /tmp`
115 | 1.upto(numberOfServers - 1) { |i| serveurs = `sed -n #{i + 1}p listOfServers | cut
116 | -d "." -f1 '.strip
117 |     `ssh root@#{serveurs} mount -o remount,user_xattr /tmp` }
118 | puts "Montage_fait!"
119 | # d marriage du serveur
120 | `ssh root@#{serveur_1} mkcephfs -c /etc/ceph/ceph.conf --allhosts -v -k /etc/ceph/
121 | keyring.bin`
122 | `ssh root@#{serveur_1} /etc/init.d/ceph -a start`
123 | puts "Serveur_ceph_d marr !"
124 | # configuration des clients
125 | 0.upto(`wc -l listOfClients` - 1) { |i| clients = `sed -n #{i + 1}p listOfClients |
126 | cut -d "." -f1 '.strip
127 |     `ssh root@#{clients} mkdir /ceph`
128 |     `ssh root@#{clients} cfuse -m #{ip_serveur} /ceph` }
129 | puts "Clients_mont s!"

```

B.4 NFS

Fichier deploymentNFS.rb :

```

1 | #!/usr/bin/ruby -w
2 | # encoding: utf-8
3 |
4 | #####
5 |
6 | # File Name : deploymentNFS.rb
7 |
8 | # Purpose :
9 |
10 | # Creation Date : 17-03-2011
11 |
12 | # Last Modified : jeu. 17 mars 2011 16:29:07 CET

```



```

13
14 # Created By : Helldar
15
16 #####
17
18 'cat $OAR_FILE_NODES | sort -u > listOfNodes '
19
20 # D ploiment des machines
21 #puts "Machines en cour de d ploiment...\n"
22 #'kadeploy3 -k -e squeeze-collective -u flevigne -f listOfNodes # image collective '
23
24 serveur = 'head -1 listOfNodes '.strip
25 puts "Le_serveur_:_#{serveur}!\n"
26 # Suppression du serveur de la liste
27 'sed -i 1d listOfNodes '
28
29 puts "Configuration_du_serveur...\n"
30 'scp exports root@#{serveur}:/etc/'
31 'ssh root@#{serveur} /etc/init.d/nfs-kernel-server restart '
32
33 puts "Configuration_des_clients...\n"
34 line = 'wc -l listOfNodes | cut -d '_' -f1 '.strip.to_i
35 puts "Il_y_a_#{line}_nodes"
36 1.upto(line) { |i| clients = 'sed -n #{i}p listOfNodes | cut -d "." -f1 '.strip
37   'ssh root@#{clients} mkdir /tmp/partage '
38   'ssh root@#{clients} mount #{serveur}:/tmp -t nfs /tmp/partage ' }

```

B.5 Benchmark

Fichier benchmark.rb :

```

1 #!/usr/bin/ruby -w
2 # encoding: utf-8
3
4
5 if ARGV[0] == nil || ARGV[1] == nil || ARGV[2] == nil
6   puts "Usage_correcte:"
7   puts "param1:_nombre_de_clients_participant_au_bench"
8   puts "param2:_fichier_de_sortie"
9   puts "param3:_url_de_la_liste_des_clients"
10  puts "param4:_lieu_d'ecriture_du_bench"
11  exit(1)
12 end
13
14 $clientsOfBench = "#{ARGV[0]}"
15
16 # chemin du fichier contenant la liste des clients
17 #listOfClients = "/home/flevigne/glusterFs/listOfClients"
18 listOfClients = "#{ARGV[2]}"
19
20 # chemin ou crire les donn es du benchmark
21 #whereToWrite = "/media/glusterfs"
22 whereToWrite = "#{ARGV[3]}"
23
24 # chemin du fichier contenant les r sultats

```

```

25 $outputRes = "#{ARGV[1]}"
26
27 # le client doit avoir dans /home/flevigne :
28 # - linux-2.6.37.tar.bz2 : noyau linux compress
29 # - bigFile : un fichier de 3 Go
30
31 # fichier contenant la liste des clients participant au benchmark
32 'touch clientOfBench'
33 'head -#{ $clientsOfBench } #{listOfClients} > clientOfBench'
34
35 # si le fichier $outputRes n'existe pas, on le cr e .
36 if !File.exist?($outputRes)
37   'touch #{ $outputRes }'
38 end
39
40 'echo "\nBenchmark_sur_#{ $clientsOfBench }_clients" >> #{ $outputRes }'
41
42 $numberOfClients = open("clientOfBench").read.count("\n").to_i
43
44 puts "Lancement_du_benchmark_sur_#{ $numberOfClients }_clients."
45
46
47 # lance un travail
48 # parametres :
49 # - name : nom du travail (str)
50 # - work : chemin du script de travail (str)
51 # - whereToWrite : chemin ou cr ire les donn es du benchmark (str)
52 # - size : taille (en Mo) du/des fichier(s) a ecrire/lire (float)
53 def startBench(name, work, whereToWrite, size)
54   puts "bench_:_#{name}_en_cours..."
55
56   totalSize = size.to_i * $clientsOfBench.to_i
57   workFinished = 0
58   startOfBench = Time.now
59
60   # execution du sript pour tous les clients
61   File.open("clientOfBench", 'r') do |file|
62     while line = file.gets
63       fork do
64         machine = line.split.join("\n")
65         'scp #{work} root@#{machine}:/root'
66         'ssh root@#{machine} ./#{work} #{whereToWrite}'
67         exit(0)
68       end
69     end
70   end
71
72   # on attend que tous les clients aient fini leur travail
73   1.upto($numberOfClients) do
74     pid = Process.wait
75     workFinished += 1
76     puts "Machine(s)_ayant_termin _leur_travail_:_#{workFinished}"
77   end
78
79   endOfBench = Time.now
80   duration = endOfBench - startOfBench

```

```

81
82 puts "Toute les machines ont termin  leur travail."
83
84 puts "――> Le benchmark \"#{name}\" a dur  #{duration} secondes. (debit : #{
      totalSize / duration} Mo/s)"
85
86 'echo "#{name} : #{duration} sec : #{totalSize / duration} Mo/s" >> #{ $outputRes
      }'
87 end
88
89
90 # lancement du benchmark
91 startBench("ecriture_de_petits_fichiers", "writingSmallFiles.sh", whereToWrite,
      479)
92 startBench("ecriture_de_gros_fichiers", "writingBigFiles.sh", whereToWrite, 3076)
93 startBench("lecture_de_petits_fichiers", "readingSmallFiles.sh", whereToWrite, 479)
94 startBench("lecture_de_gros_fichiers", "readingBigFile.sh", whereToWrite, 3076)
95
96 # nettoyage du syst me de fichier distribue (necessaire pour encha ner les
      benchmark)
97 puts "Nettoyage de l'espace de travail..."
98 oneClient = 'head -1 clientOfBench'.strip
99 'ssh root@#{oneClient} rm -r #{whereToWrite}/*'
100
101 puts "\nBenchmark termine"

```

Fichier writingSmallFiles.sh :

```

1 #!/ bin / bash
2
3 whereToWrite=$1
4
5 nameOfMachine='uname -n'
6
7 # creation du repertoire de travail de la machine
8 mkdir "$whereToWrite/$nameOfMachine"
9
10 # decompression dans ce repertoire
11 cd "$whereToWrite/$nameOfMachine"
12 tar -xf /home/flevigne/linux-2.6.37.tar.bz2

```

Fichier writingBigFiles.sh :

```

1 #!/ bin / bash
2
3 whereToWrite=$1
4
5 nameOfMachine='uname -n'
6
7 # on copie le gros fichier au lieu voulu
8 cp /home/flevigne/bigFile "$whereToWrite/$nameOfMachine"

```

Fichier readingSmallFiles.sh :

```

1 #!/ bin / bash
2
3 whereToWrite=$1
4

```

```
5 nameOfMachine='uname -n'
6
7 cd "$whereToWrite/$nameOfMachine"
8
9 # lecture des fichiers du noyau linux (compression (donc lecture) redirig vers /
  dev/nul)
10 tar -cf /dev/null linux-2.6.37
```

Fichier readingBigFile.sh :

```
1 #!/bin/bash
2
3 whereToWrite=$1
4
5 nameOfMachine='uname -n'
6
7 cd "$whereToWrite/$nameOfMachine"
8
9 # lecture du gros fichier
10 cat bigFile > /dev/nul
```