Journées Mathrice
octobre 2010

rZnapshot

David Delavennat, Centre de Génétique Moléculaire, CNRS

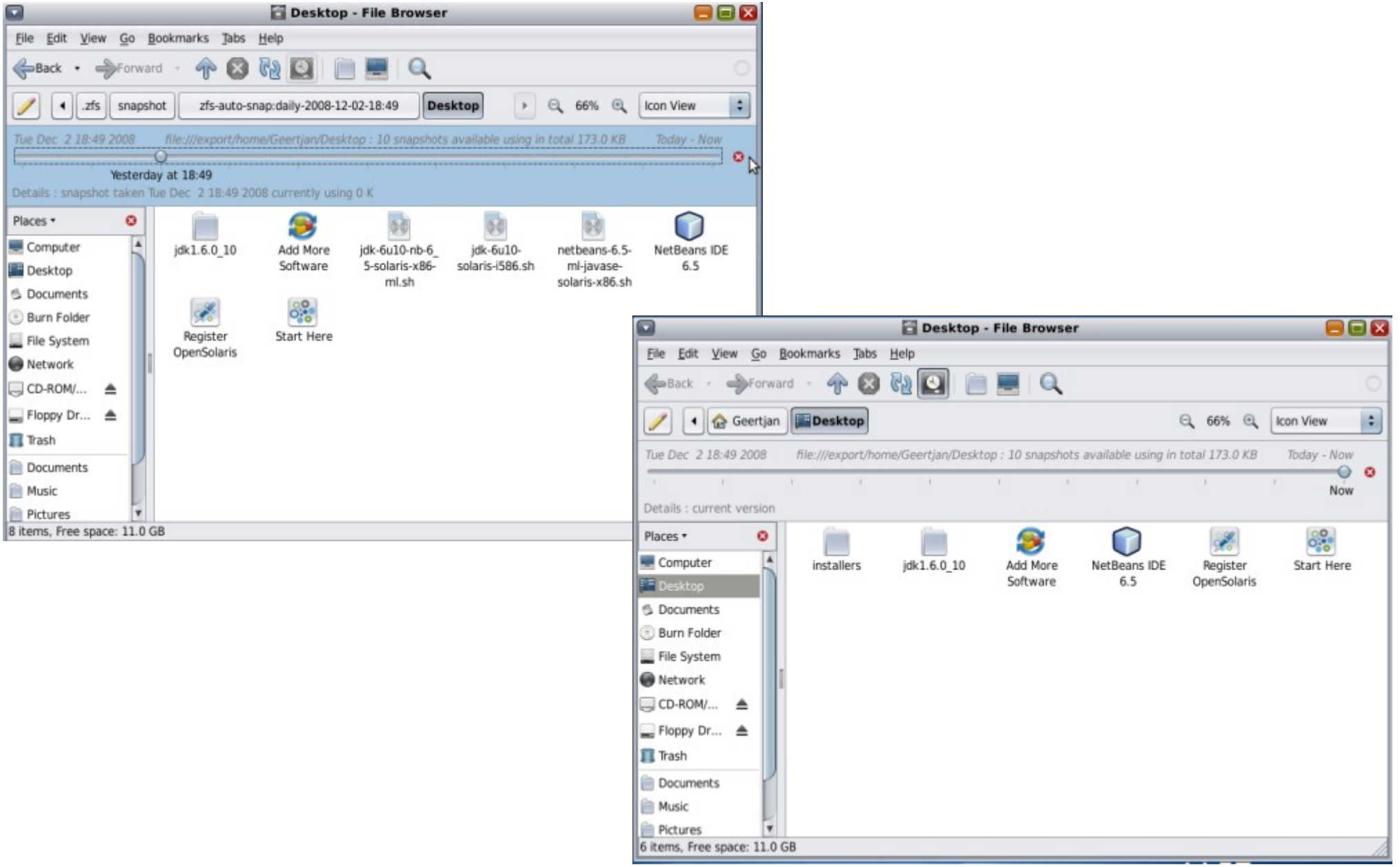
Plan

- Pourquoi faire?
- Architecture
- {Open}Solaris / TimeSlider
- ORM / Sequel / sqlite
- Logs
- rZnapshot
- Quelques chiffres

Pourquoi faire?

- Replication géolocalisée de serveur de fichier
- Réplication Incrémental → sur réseau WAN
- Plan de Reprise d'Activité

TimeSlider


Desktop - File Browser
File Edit View Go Bookmarks Tabs Help
Back Forward
.zfs snapshot zfs-auto-snap:daily-2008-12-02-18:49 Desktop 66% Icon View
Tue Dec 2 18:49 2008 file:///export/home/Geertjan/Desktop : 10 snapshots available using in total 173.0 KB Today - Now
Yesterday at 18:49
Details : snapshot taken Tue Dec 2 18:49 2008 currently using 0 K
Places
Computer
Desktop
Documents
Burn Folder
File System
Network
CD-ROM/...
Floppy Dr...
Trash
Documents
Music
Pictures
jdk1.6.0_10
Add More Software
jdk-6u10-nb-6_5-solaris-x86-ml.sh
jdk-6u10-solaris-i586.sh
netbeans-6.5-ml-javase-solaris-x86.sh
NetBeans IDE 6.5
Register OpenSolaris
Start Here
8 items, Free space: 11.0 GB
Desktop - File Browser
File Edit View Go Bookmarks Tabs Help
Back Forward
Geertjan Desktop 66% Icon View
Tue Dec 2 18:49 2008 file:///export/home/Geertjan/Desktop : 10 snapshots available using in total 173.0 KB Today - Now
Now
Details : current version
Places
Computer
Desktop
Documents
Burn Folder
File System
Network
CD-ROM/...
Floppy Dr...
Trash
Documents
Music
Pictures
installers
jdk1.6.0_10
Add More Software
NetBeans IDE 6.5
Register OpenSolaris
Start Here
6 items, Free space: 11.0 GB

Object Relational Mapping

- crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle
- correspondances entre cette base de données et les objets du langage utilisé
- associe une ou plusieurs classes avec une table, et chaque attribut de la classe avec un champ de la table.
- Par exemple, la classe Customer est associée avec la table CUSTOMER, et les attributs sont associés comme suit:
  - Customer.customerId → CUSTOMER.CUSTOMER_ID
  - Customer.customerName → CUSTOMER.CUSTOMER_NAME
  - Customer.customerAddress → CUSTOMER.CUSTOMER_ADDRESS

Sequel / sqlite

- Sequel → ORM ruby
- Sqlite → base de donnée fichier / mémoire

```ruby
require "rubygems"
require "sequel"

# connect to an in-memory database
DB = Sequel.sqlite

# create an items table
DB.create_table :items do
  primary_key :id
  String      :name
  Float       :price
end

# create a dataset from the items table
items = DB[:items]

# populate the table
items.insert(:name => 'abc', :price => rand * 100)
items.insert(:name => 'def', :price => rand * 100)
items.insert(:name => 'ghi', :price => rand * 100)

# print out the number of records
puts "Item count: #{items.count}"

# print out the average price
puts "The average price is: #{items.avg(:price)}"
```

rZnapshot : principe

- libzfs.rb
  - DB sqlite mémoire
  - cache de l'état de synchronisation des filesystèmes
    - Auto-expiration du cache
    - Auto-rafraichissement du cache
  - ZFS send et ZFS recv
- cli snapsync et snapstate utilisant libzfs.rb

rZnapshot : cache

```
Class Cache
  SECOND_01 =  1
  MINUTE_01 = 60 * SECOND_01
  def initialize(...)
    @refresh_rate = MINUTE_01
    @last_refresh = Time.now - ( @refresh_rate)
    @db = Sequel.sqlite
    @db.create_table :filesystems do
      primary_key     :snapshot_order
      String          :filesystem_name
      String          :source_snapshot_name
      String          :target_snapshot_name
      String          :snapshot_type
      String          :snapshot_creation
      Integer         :snapshot_creation_unix_time
    end
    @filesystems = @db[:filesystems]
  end
  def filesystems
    time_since_last_refresh = Time.now - @last_refresh
    if time_since_last_refresh > @refresh_rate
      puts "cache needs refreshing"
      refresh
      @last_refresh = Time.now
    end
    @filesystems
  end
  def refresh
    @filesystems.delete
    acquire_source_and_target_filesystems_properties
  end
end
```

rZnapshot : cache

```
def cache_source_filesystems_properties(filesystems_properties)
  filesystems_properties.each{|filesystem_properties|
    @filesystems.insert(
      :filesystem_name               => filesystem_properties[:filesystem_name],
      :source_snapshot_name          => filesystem_properties[:snapshot_name],
      :snapshot_type                 => filesystem_properties[:snapshot_type],
      :snapshot_creation             => filesystem_properties[:snapshot_creation],
      :snapshot_creation_unix_time   => filesystem_properties[:snapshot_creation_unix_time]
    )
  }
end
def cache_target_filesystems_properties(filesystems_properties)
  filesystems_properties.each{|filesystem_properties|
    If # no match found
      @filesystems.filter(
        :filesystem_name             => filesystem_properties[:filesystem_name],
        :source_snapshot_name        => filesystem_properties[:snapshot_name]
      ).count == 0
    then # filesystem have been deleted from master
      @filesystems.insert(
        :filesystem_name             => filesystem_properties[:filesystem_name],
        :target_snapshot_name        => filesystem_properties[:snapshot_name],
        :snapshot_creation           => filesystem_properties[:snapshot_creation],
        :snapshot_creation_unix_time => filesystem_properties[:snapshot_creation_unix_time]
      )
    else # filesystem exists on slave AND master
      @filesystems.filter(
        :filesystem_name             => filesystem_properties[:filesystem_name],
        :source_snapshot_name        => filesystem_properties[:snapshot_name]
      ).update(
        :target_snapshot_name        => filesystem_properties[:snapshot_name]
      )
    end
  }
end
```

rZnapshot : filesystems

```
class Filesystems
  attr_reader :name
  def initialize(cache,filesystem_name)
    @cache   = cache
    @dataset = cache.filesystems
    @name    = filesystem_name
  end
  def source
    @dataset = @dataset.select(
      :filesystem_name
    ).filter(
      ~{ :source_snapshot_name => nil }
    ).distinct
    return self
  end
  def target
    @dataset = @dataset.select(
      :filesystem_name
    ).filter(
      ~{ :source_snapshot_name => nil }
    ).distinct
    return self
  end
  def name
    @dataset.select(
      :filesystem_name
    )
    return self
  end
  def each
    @dataset.all.each{|data|
       yield Filesystem.new(@cache,data)
    }
  end
end
```

Log

```
Creating lock file /var/run/snapsync.pid
cache needs refreshing
Acquiring source filesystems properties[DONE]
Acquiring target filesystems properties[DONE]
Merging filesystems properties         [DONE]
================================================================================
Synchronizing filesystem data
 ==> Removing expired snapshots
+-----------------------------------------+----------------------+-------------+-----------------------------------------+
|source_snapshot_name                     |snapshot_creation     |snapshot_order|target_snapshot_name                     |
+-----------------------------------------+----------------------+-------------+-----------------------------------------+
|data@zfs-auto-snap:monthly-2010-05-01-00:00 |Sat May  1 00:00:00 2010|            1|data@zfs-auto-snap:monthly-2010-05-01-00:00 |
...
|                                         |Thu Sep  9 08:00:00 2010|         2123|data@zfs-auto-snap:hourly-2010-09-09-08:00  |
|data@zfs-auto-snap:hourly-2010-09-09-09:00  |Thu Sep  9 09:00:00 2010|            7|                                         |
+-----------------------------------------+----------------------+-------------+-----------------------------------------+
ssh occam_4g 'zfs destroy data@zfs-auto-snap:hourly-2010-09-09-08:00'
 ==> Updating unsynchronized snapshots
+-----------------------------------------+----------------------+-------------+-----------------------------------------+
|source_snapshot_name                     |snapshot_creation     |snapshot_order|target_snapshot_name                     |
+-----------------------------------------+----------------------+-------------+-----------------------------------------+
|data@zfs-auto-snap:monthly-2010-05-01-00:00 |Sat May  1 00:00:00 2010|            1|data@zfs-auto-snap:monthly-2010-05-01-00:00 |
...
|data@zfs-auto-snap:daily-2010-09-09-00:00   |Thu Sep  9 00:00:00 2010|            6|data@zfs-auto-snap:daily-2010-09-09-00:00   |
|data@zfs-auto-snap:hourly-2010-09-09-09:00  |Thu Sep  9 09:00:00 2010|            7|                                         |
+-----------------------------------------+----------------------+-------------+-----------------------------------------+
Source.snapshot.first: data@zfs-auto-snap:monthly-2010-05-01-00:00
Synchronize.from:      data@zfs-auto-snap:daily-2010-09-09-00:00
Synchronize.to:        data@zfs-auto-snap:hourly-2010-09-09-09:00
date; zfs send -v -I 'data@zfs-auto-snap:daily-2010-09-09-00:00' 'data@zfs-auto-snap:hourly-2010-09-09-09:00' | /opt/csw/bin/pv -r |
ssh occam_4g zfs recv -dFv 'data'; date
Thu Sep  9 10:00:08 CEST 2010
sending from @zfs-auto-snap:daily-2010-09-09-00:00 to data@zfs-auto-snap:hourly-2010-09-09-09:00
receiving incremental stream of data@zfs-auto-snap:hourly-2010-09-09-09:00 into data@zfs-auto-snap:hourly-2010-09-09-09:00
received 312B stream in 3 seconds (104B/sec)
Thu Sep  9 10:00:11 CEST 2010
...
Removing lock file /var/run/snapsync.pid
```

Quelques chiffres

- En production depuis 6 mois
- Nombre de filesystèmes concernés et taille du pool au 09/09

```
servinfo@babel:~$ date;zfs list -t snapshot|grep 'data/' | wc -l
Thu Sep  9 10:55:12 CEST 2010
2115
servinfo@babel:~$ zpool list |grep data
data   21.8T  16.4T  5.36T    75%  ONLINE  -
```