A validation system for Data Preservation in HEP
What you will know after this talk

- Aspects of Data-Preservation
- Integration into your Software
  - by introducing a validation framework which is based on virtualization
- Your Benefits of using Data-Preservation
DPHEP - International Study Group on Data Preservation

Chair: Cristinel Diaconu (DESY/CPPM)

Working Groups
- Physics Cases: François Le Diberder (SLAC/LAL)
- Preservation Models: D. South (DESY), Homer Neal (SLAC)
- Technologies: Stephen Wolbers (FNAL), Yves Kemp (DESY)
- Governance: Salvatore Mele (CERN)

International Steering Committee
- Participants from ee, ep and pp collider experiments
- Associated computing centres at the labs
- Some funding agencies

International Advisory Committee
- Chairs: Jonathan Dorfan (SLAC), Siegfried Bethke (MPIM)
- Advisers: Gigi Rolandi (CERN), Michael Peskin (SLAC), Dominique Boutigny (IN2P3), Young-Kee Kim (FNAL), Hiroaki Aihara (IPMU/Tokyo), Alex Szalay (JHU)

Series of DPHEP workshops held since 2009

Group has grown since 2008 to over 100 contact persons
Endorsed by ICFA summer 2009
LHC experiments joined in 2011

http://www.dphep.org/
DPHEP - International Study Group on Data Preservation

2009 SLAC

2009 DESY

2010 KEK

2011 FERMILAB

2009 CERN
Pizza Preservation

- different scope & timeline for preservation

- preserve the taste & look & smell ...

- for a couple of days
  - put it into a fridge

- for a couple of month
  - put it into a freezer

- for a couple of years ?? ??
  - ingredients still available? recipe still known? ...
Pizza Preservation

- one way of preserve it for many years
  - not only concentrate on the pizza
  - concentrate on the process of creating the pizza
  - after creating the pizza
    -> review it, the taste & smell & . . .
  - repeat it every month or better, every week
What we have developed :o)
Putting software in the fridge / freezer?

- How to guarantee access in N-years?
- How to guarantee correctness of data / output?

Different strategies have been developed

- Virtualization: put the software into a VM
  -> good for limited timeline & scope

- Continuous migration
  -> cook the same recipe ever & ever again
  -> try to automate this process
The Generic Recipe
(Atomic Test Life-Cyle)

I
Instantiate VM

II
Prepare Platform

III
Build Test-Software

IV
Run Test-Software

V
Validate Test-Results

VI
Shutdown & Save VM
... and the two Cooks

I
Instantiate VM

II
Prepare Platform

VI
Shutdown & Save VM

V
Validate Test-Results

IV
Run Test-Software

III
Build Test-Software

separation of IT and User
For each configuration: Run this test cycle often

You will soon detect when things break e.g.

- A needed library is no longer available in the distro
- Software (SW) does not compile anymore because of some update
- SW does not run: Internal error e.g. some API changed
- SW does not run: External error e.g. Access to mass storage changed
- SW validation fails: Internal error e.g. compiler optimization behaves different
- SW validation fails: External error e.g. new chip generation computes different

You can run daily tests by hand ... but easier to use virtualization
A simple Test Set-Up for **HelloWorld** (hello.c)

- **your code**
  - `hello.c`

- **a build.sh script**
  - `./configure && make && make install`

- **a run.sh script**
  - `./hello > hello.out 2 > hello.err`

- **a validation.sh script**
  - `md5sum hello.out hello.err`

- **additional packages in the VM image**
  - gcc in version 4.N.N

- **information about the desired VM image**
  - ScientificLinux 6.N 64bit
A walk through the system developed at DESY

Users View: Describe the Test
Configuration Example for Testing ROOT

Build Configuration Files

- configuration.txt
- contextualisation.txt
- rpm.txt
- vmTemplate.txt
- software.txt
- validator.txt
- vm.txt
Build Configuration Files

- configuration.txt
- rpm.txt
- software.txt
- validator.txt
- vm.txt

- contextualisation.txt
- vmTemplate.txt
Content of Configuration Files

```json
{"testcollection":
  {
    "name" : "ROOT compiling test",
    "description": "ROOT compiling test",
    "owner":
      {
        "name" : "marco",
        "email" : "marco@localhost.com"
      }
  }
}
```
Content of Configuration Files

```json
{"packages":
{"gcc-c++":
{"version": "4.1.2",
"arch": "x86_64",
"summary": "C++ support for GCC"},
"libX11-devel":
{"version": "1.0.3",
"arch": "x86_64",
"summary": "X.Org X11 libX11 development package"},
"libXft-devel":
{"version": "2.1.10",
"arch": "x86_64",
"summary": "X.Org X11 libXft development package"},
"libXpm-devel":
{"version": "3.5.5",
"arch": "x86_64",
"summary": "X.Org X11 libXpm development package"},
"libXext-devel":
{"version": "1.0.1",
"arch": "x86_64",
"summary": "X.Org X11 libXext development package"}}
}```
Configuration Example for Testing ROOT

Content of Configuration Files

- configuration.txt
- rpm.txt
- software.txt
- validator.txt
- vm.txt

```json
{"experiment_software":
{"archive":
"http://some.web.server:80/root_sptest.tar.gz",
"builder": "build.sh",
"executable": "run.sh"}
}
```

Web-Services hosting the referenced file
Content of Configuration Files - Test-Logic Reference

```json
{"experiment_software":
{"archive":
  "http://some.web.server:80/root_sptest.tar.gz",
  "builder": "build.sh",
  "executable": "run.sh"}
}
```
Content of Configuration Files

- configuration.txt
- rpm.txt
- software.txt
- validator.txt
- vm.txt

Configuration Example for Testing ROOT

```json
{"validator":
  {"archive":
    "http://some.web.server:80/root_validator.tar.gz",
    "executable": "validator.sh"}
}
```

Web-Services hosting the referenced file
Content of Configuration Files - Script Payload

```bash
#!/bin/sh

PACKAGE="root_v5.26.00.source.tar.gz"
TARGET=./rootSrc

mkdir $TARGET
cd $TARGET

echo "extracting '$PACKAGE'..."
tar xzvf ../$PACKAGE"

#set env
export ROOTSYS=$(pwd)/root
ROOTSYS=$(pwd)/root

#configure ROOT
cd $ROOTSYS
./configure linuxx8664gcc

#make ROOT
make
```
#!/bin/sh

export ROOTSYS=$(pwd)/rootSrc/root
export PATH=$ROOTSYS/bin:$PATH
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH

cd ${ROOTSYS}/test/

#part of the run-step can also be a build-call
make

echo "running 'stressHepix' test..."
./stressHepix

echo "running 'bench' test..."
./bench
### Content of Configuration Files

**configuration.txt**

```markdown
# Configuration Contents
```

**rpm.txt**

```markdown
# RPM Packages
```

**software.txt**

```markdown
# Software Setup
```

**validator.txt**

```markdown
# Validation Logic
```

**vm.txt**

```json
{
    "virtual_machine": {
        "template": "http://grid-lab024:18001/cloud/templates/349",
        "type": "small"
    }
}
```
Register and Launch a Test

Configuration Example for Testing ROOT

Register Test

Start Test-Run

```
curl -F "file=testsuite.tar.gz;
filename=testsuite.tar.gz"
http://spsystem:18003/api/experiment/
testsuite/

{"RETURN_CODE": "0",
"TESTCOLLECTION_UUID": "215a4c8a3934b9e8c957fd6650d3b7b5"}
```

```
curl -o response.txt -F CONFIG=
{"MESSAGE": 
{"NAME": "START_TESRUN",
"TESTCOLLECTION_UUID": "215a4c8a3934b9e8c957fd6650d3b7b5" }}
http://spsystem:18003/api/message/

{"RETURN_CODE": "0",
"TSTRUN_UUID": "1302111637.1498589515",
"TESTCOLLECTION_UUID": "215a4c8a3934b9e8c957fd6650d3b7b5"}
```
Automatic Execution inside a VM - **Build Step**

Configuration Example for Testing ROOT
Configuration Example for Testing ROOT

Automatic Execution inside a VM - Run Step
Configuration Example for Testing ROOT

Automatic Execution inside a VM - Validate Step

V
Validate Test-Software

III
Build Test-Software

IV
Run Test-Software

Results

validate.sh

Process

Results
Configuration Example for Testing ROOT

Get Results

curl -o build.log
http://spsystem:18003/api/experiment/results/215a4c8a3934b9e8c957fd6650d3b7b5/logfiles/1302111637.1498589515/build.log

build.log

http://spsystem:18003/api/reports/task_status_for_single_test/215a4c8a3934b9e8c957fd6650d3b7b5

TEST_STEP
startTest
provisioningVirtualMachine
virtualMachineUpAndRunning
installPackages
extractSoftware
buildSoftware
runSoftware
validateResults
shutdownVirtualMachine
testFinished
**Status of a Specific Tests**

<table>
<thead>
<tr>
<th>TEST_STEP</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTest</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>provisioningVirtualMachine</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>virtualMachineUpAndRunning</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>installPackages</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>extractSoftware</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>buildSoftware</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>runSoftware</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>validateResults</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>shutdownVirtualMachine</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>testFinished</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
### Configuration Example for Testing ROOT

#### Status for All Tests

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
wget http://sp-system:18003/api/experiment/results/
2f955c190b44b11c1957b5d9ba655b18/logfiles/1308752935.3536250591/run.log

**HOMEDIR at the beginning of run.sh:**
> export HOMEDIR=/home/dphep
> echo /home/dphep
> /home/dphep
> export HOST=dpheptest
> echo dpheptest
dpheptest
> export AFSFILE="/usr/vice/etc/ThisCell"
> /etc/init.d/afs start
Starting AFS client.....
dnsdomainname: Unknown host
afs: No cell defined. Trying to figure it out failed. Not starting.
getting afs token...
klog: error reading cell database Can't get local cell name!
CHECKING...
> ls /afs/company.de/view/experiment/ndb_read_only
ls: /afs/company.de/view/experiment/ndb_read_only: No such file or directory
> cd /home/dphep
(...)

Configuration Example for Testing ROOT

Error-Tracking - Investigate failed Test-Step
First Experiences from Experiments

- tested by HERA experts
  - detected external dependencies: cvs, db, afs, ...
  - set-up their test-environment using SL5 as one large sequential test

- feedback used to improve prototype

- experiments wish list
  - debug-access to VM
  - external data access, e.g. to dCache
  - run tests on different hardware architectures
Benefits of validation for software projects

- improved code-quality
  - identify external dependencies
  - documentation
- simplifies migration efforts
  - automatically run tests on new platforms
- helps increasing your software-project “Bus Factor”
Summary

> Framework designed for software-

- verification
- validation
- migration support

> Not designed for

- mass production
- large scale analysis
Outlook

- New Hardware for Testbed ordered
- Support for different Hardware Architectures
- CSV Output, enabling self-generated reports by each user
- investigations on dCache access (PASSIVE-mode)
Media Reference

- http://www.youtube.com/watch?v=nv7VUqPE8AE
- http://images.gastrodax.de/product_images/info_images/b5421high.jpg
- http://angelabilbo.tripod.com/sitebuildercontent/sitebuilderpictures/cookbook-color.gif
- http://www.siliconcloud.com/Portals/55887/images/ingredients%20of%20inbound%20marketing.jpg
Backup