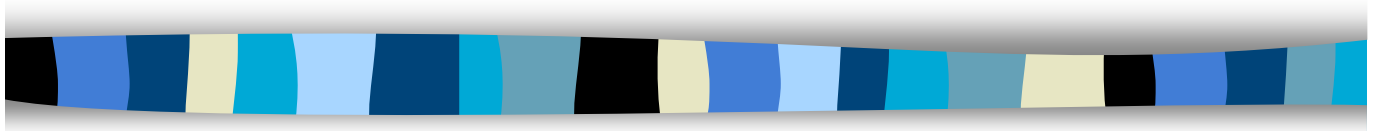




# The LCG Test Suites



**Gilbert Grosdidier**

*LAL-Orsay/IN2P3/CNRS  
& LCG*

25 mai 2004

LCG Test Suites @ HEPiX Edinburgh - GG

1



## Talk plan, and Credits

- Suite contents and purposes
- Test S/W design
  - Configure step
  - Test Loop
  - Plug-ins
  - Definition files
  - Presenter step
- The different kinds of tests
  - more on the storms
- The CLI options
  - with an example of submission
- Result examples
  - and other useful links
- The CTB
- The current set of Testing Suites was mainly contributed by:
  - Miquel Barcelo
  - Frédérique Chollet
  - Gilbert Grosdidier
  - Andrey Kiryanov
  - Charles Loomis
  - Gonzalo Merino
  - René Météry
  - Danila Oleynik
  - Andrea Sciabà
  - Elena Slabospitskaya
- Many other people contributed to the design and ideas leading to the current suites
  - among them the INFN Testing Team

25 mai 2004

LCG Test Suites @ HEPiX Edinburgh - GG

2



## Acronyms for the newbies

- CE: Computing Element
  - the gateway to the WNs
- SE: Storage Element
- SRM: Storage Resource Manager
- (E)RM: (EDG) Replica Manager
  - the 3 above are the 3 Devils
- BDII: Information Index
- RB: Resource Broker
- PX: Myproxy Server
- WN: Worker Node(s)
  - ie the Batch Worker(s)
- UI: User Interface
  - the passport for Hell ... or Heaven ?
- LCG: L(HC) Computing Grid



## Four levels of testing required

- Installation and Configuration Testing
  - targets each machine/service individually
- Unit Testing
  - targets each basic functionality of a given service independently from the rest of the TB
- Functionality Testing
  - for the whole TB, exercises full functionalities, from a user point of view, but one by one
- Stress Testing
  - same as above, but with sophisticated jobs (several functionalities required at the same time), and with a huge number of jobs.
- The most basic testing was dropped because of lack of manpower
  - Meaning Install & Config, and Unit Testing



## Four levels of testing required

- Installation and Configuration Testing **Almost dropped**
  - targets each machine/service individually
- Unit Testing **Only ~50% developed**
  - targets each basic functionality of a given service independently from the rest of the TB
- Functionality Testing **Fully developed**
  - for the whole TB, exercises full functionalities, from a user point of view, but one by one
- Stress Testing **Fully developed**
  - same as above, but with sophisticated jobs (several functionalities required at the same time), and with a huge number of jobs.
- **The most basic testing was dropped because of lack of manpower**
  - Meaning Install & Config, and Unit Testing



## The TSTG suite Dedication

- **Site Certification**
  - initial check of the install
  - check when changes occur
  - regular re-checking (daily checks) <- this precisely implies Unit Testing
    - FS full, memory exhaustion, DB full, list full, no more inodes, hanging server(s)
- **TB Certification**
  - Basic Functional Tests of components (services)
  - Basic Grid Functionalities, with individual tests
  - Full Grid Functionalities on a full TB, including remote sites, with global/group tests
    - **Beyond (HEPCAL and Exp. Beta Testing) is outside of TSTG scope**
- **M/W Validation (Functionalities, Robustness, Stress Testing)**
  - Basic Functional Tests of components (services)
  - Basic Grid Functionalities, on a well defined/known TB, like the Cert TB
  - Full Grid Functionalities, including stress testing and group testing
    - **task dedicated to performance/functionality comparisons with previous version**
      - thru definition tests
      - the major requirement being then **automated running** (thru cron jobs)
        - no prompting, no passphrases

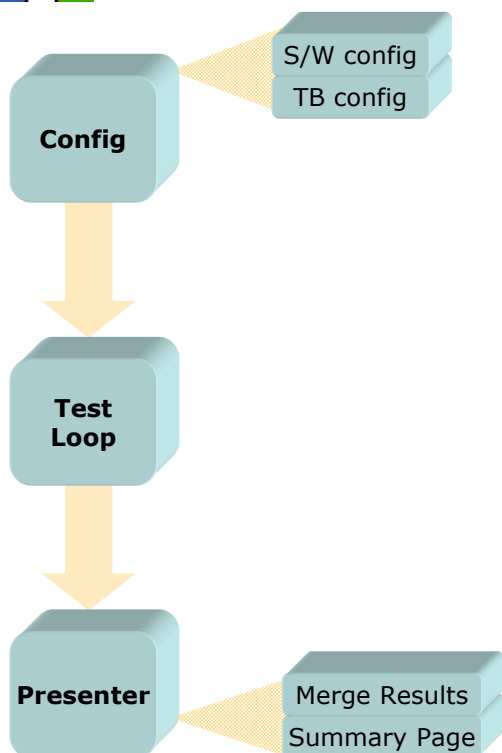


## Test S/W design, & Configure Task

- It is split into 2 main parts, dedicated to 3 tasks
  - the **top level driver**, written in Perl
  - the **plug-ins**, one for each specific test
- The top level driver is responsible for these 3 tasks:
  - **Configure**: build environment (S/W libraries & configurations, TB config)
  - **Test Loop**: run one or several selected tests
  - **Presenter**: merge the results and build the Web enabled summary page
- The **Configure** task
  - sets the environment for test S/W **libraries**
  - and the **testbed** configuration itself (i.e. the nodes to be tested)
  - 2 sets of **options** are available
    - the **general** options for the top level configuration
      - selecting the TB, verbosity, VO, main RB,
    - the options **specific** to each plug-in which will be run

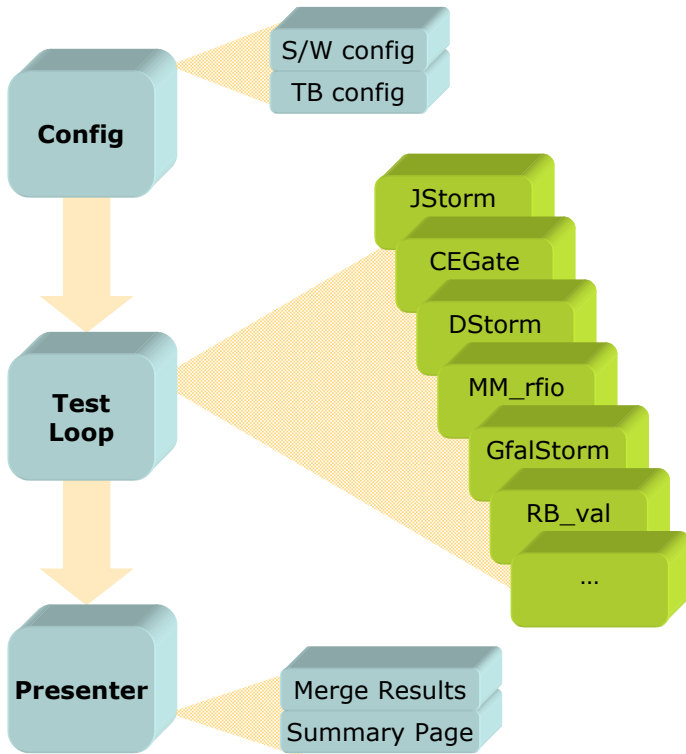


## The TSTG S/W structure

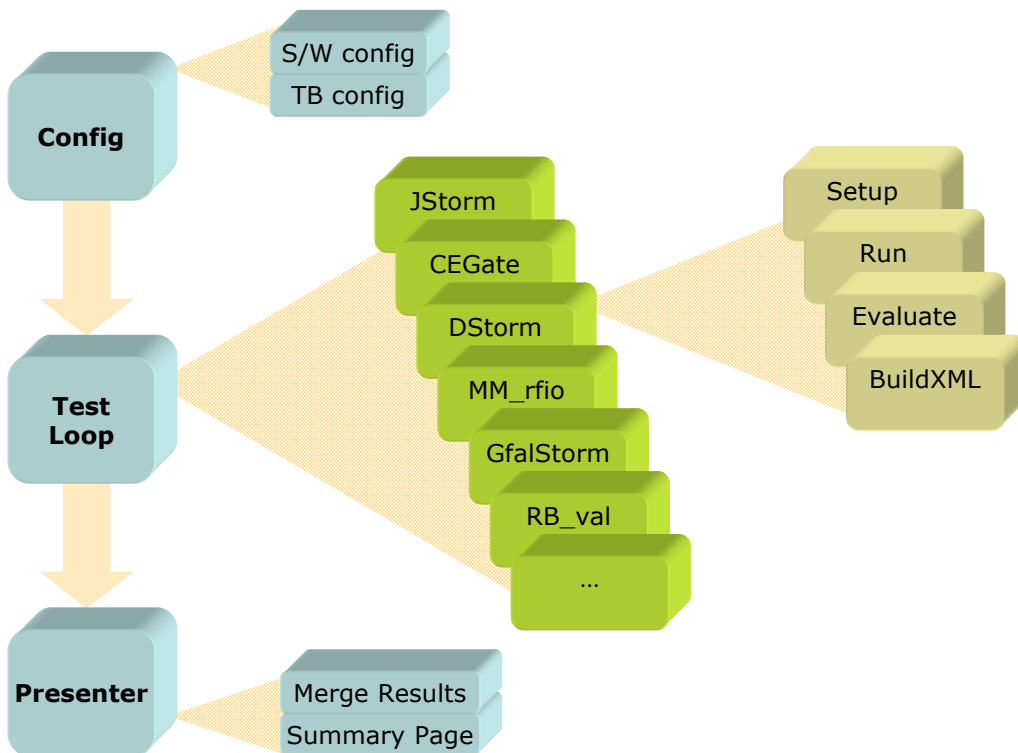




# The TSTG S/W structure

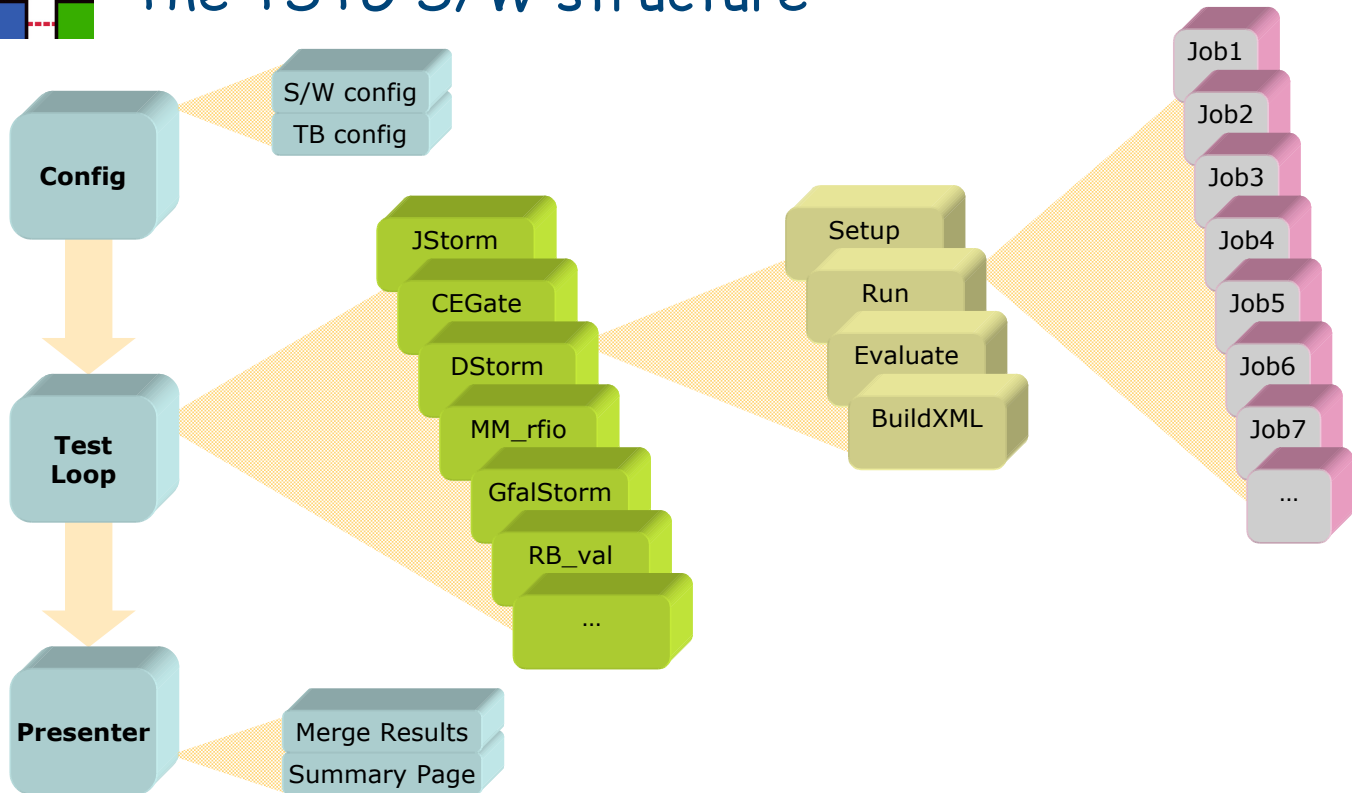


# The TSTG S/W structure





## The TSTG S/W structure



## The Test Loop design

### ■ The Test Loop features:

- it is built out of a lot of more or less independent **plug-ins**
- this offers more **robustness** in case one crashes
  - execution of the suite can proceed to the next one harmlessly
- different **languages** are allowed (flexibility and openness)
  - Perl OO technique for most of them
  - but also a few of them in bash
  - other languages are welcome (Java, C, C++, ...)
- provided the plug-ins share the same **input** (CLI) and **output** (XML) **design**
  - Input requires switches to target all or some nodes only
  - Output includes: exit status, summary results printed to STDOUT, detailed results going to an XML file
- they should be **killable** (thru the top level process)
- they should create **no side effect** for the other tests
  - no processes still spinning when they are done
  - no left over jobs without being cancelled
  - no files left on the SEs whenever possible



## More on the Plug-in design

- The Test phase allows to run several tens of generic tests
  - using all the same environment
    - a specific testbed for example
  - however each one is aiming at a specific kind of services
    - all the CEs of a given TB for example
  - one may select a bunch of them, or run them one at a time
- It is often required to run a whole bunch of tests where one needs to specify different targets or configurations for different tests
  - One may want to choose
    - only 3 CEs out of 7 for the RB tests
    - while using all available CEs for the SRM-SE stress test on dCache SEs
- The solution goes thru the so-called « definition files »
  - allowing for very flexible construction of test batches



## The « definition file » feature

- Each line in such a file is indeed:
  - targeting only one of the generic tests
  - with very detailed specifications (thru option parameters)
    - on the target machines
    - on the conditions of the test (input, output, speed, etc...)
- In a **definition file**, the same generic test can be repeated several times
  - with different conditions or specifications
  - targeting different subsets of nodes
- This feature is used in several opportunities
  - where a preformatted bunch of tests has to be run
    - when running a cron job
    - when comparing different results is required
      - for TB certification or M/W validation
  - when building a test with an automated tool or GUI
  - when a single shot is required to launch many tests at the same time



## The Presenter level

- After each step in the test loop, information is gathered about the step results:
  - exit status, output files,
  - elapsed time,
  - effective options, actual command line used,
  - nodes effectively tested, ...
- When the loop is over, an **overall summary** Web page is created:
  - to merge all these informations in an « at a glance » fashion
  - to allow navigation down to detailed files to track the cause of the failures
  - to give access for each step to a documentation page gathering:
    - the description of the step
    - the way to re-issue the command
      - to reproduce the failure (conditions, options, nodes)
    - or to clone it into a different environment, or to re-use some parts of it in a different way to cross-check its (weird ;-) results



## The different kinds of Test [v0.1.12]

- **Watch out:** test names are far from meaningful
- **CEGate:** Globus Gatekeeper Unit Testing (CEs)
  - 11 tests achieved on each CE node
- **CECycle:** submit jobs to specific CEs systematically
- **UI\_ST:** UI functionality tests for SiteTesting
- **FTP:** GridFTP functionality tests (RBs, CEs, SEs)
- **DNS:** reverse DNS Tests (RBs, CEs, SEs, PX)
- **RB\_val:** Functionality tests for RBs (Unit testing)
  - a suite of 5 small jobs submitted through JDL files
- **SEwsCycle:** Checkup of SE setup (WP5-SEs) - does not work yet
- **RMCycle:** Checkup of RM setup (SEs, RM)
- **PXRenew:** to check Proxy Renewal from inside a WN job
  - very touchy



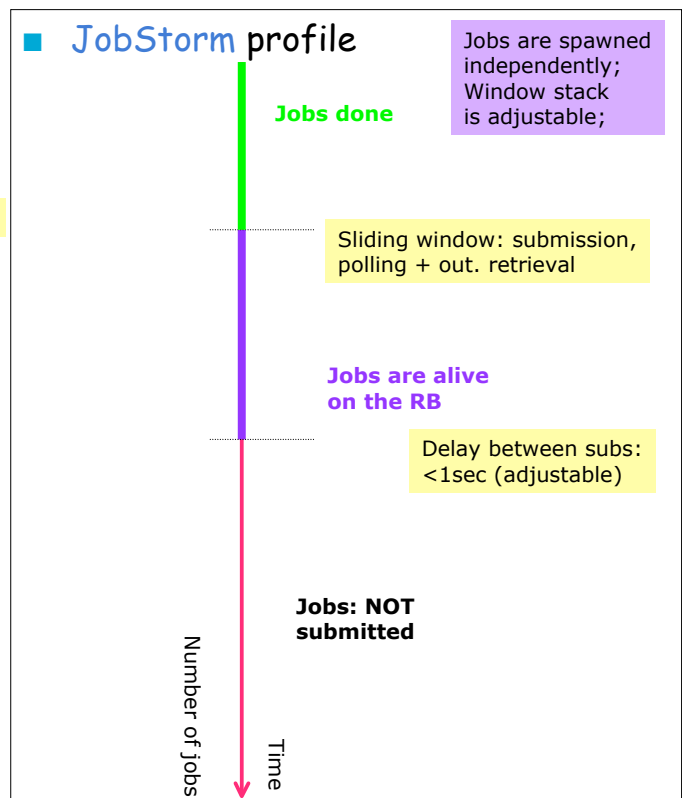
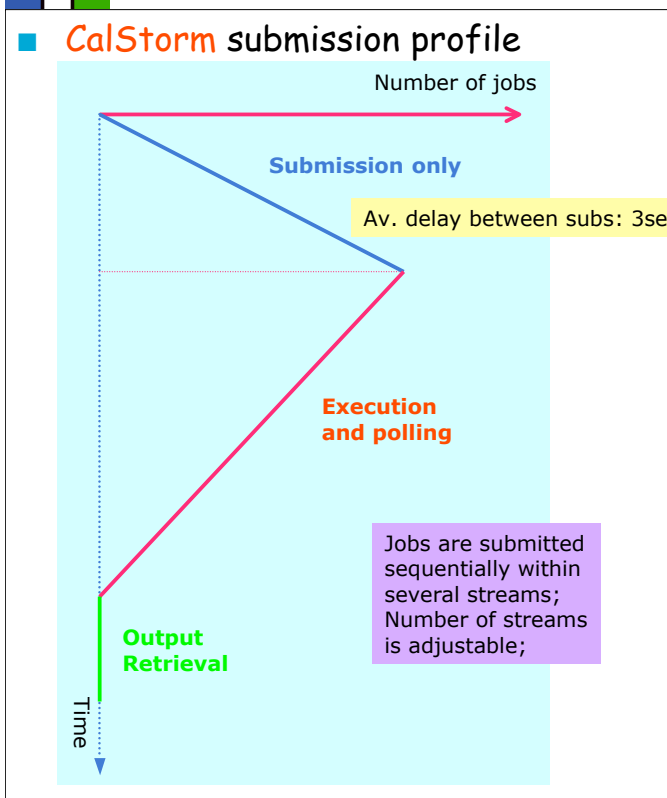


# The different kinds of Test (2)

- **The Storms** (All components, Global & Stress testing):
  - **JStorm**: Job Storm
    - Simple jobs with InpOut sandbox transfers, and check of output contents
    - --batchSleep option also available
  - **CStorm**: Copy Storm
    - Achieves random GridFTP transfers from jobs running on WNs
  - **RStorm**: Replica Storm
    - Broadcasts files thru RM from the UI,
    - and checks for availability on SEs from the WNs
  - **KStorm**: Checksum Storm
    - Achieves big file sandbox transfers, with both end checksums
  - **UStorm**: User Storm
    - Where the user may provide his own JDL xor Script files
  - **CalStorm**: the storm engine is a different one
    - this allows sending more jobs in a row, and is more stressful for the submit phase
    - usually, the jobs are 10sec. sleepers
    - but 50-100min sleepers aim to check load balance on CEs



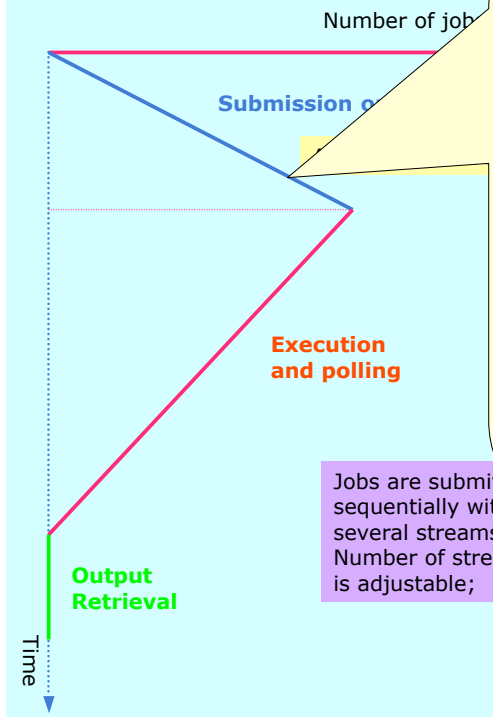
# Why 2 kinds of Storm ?



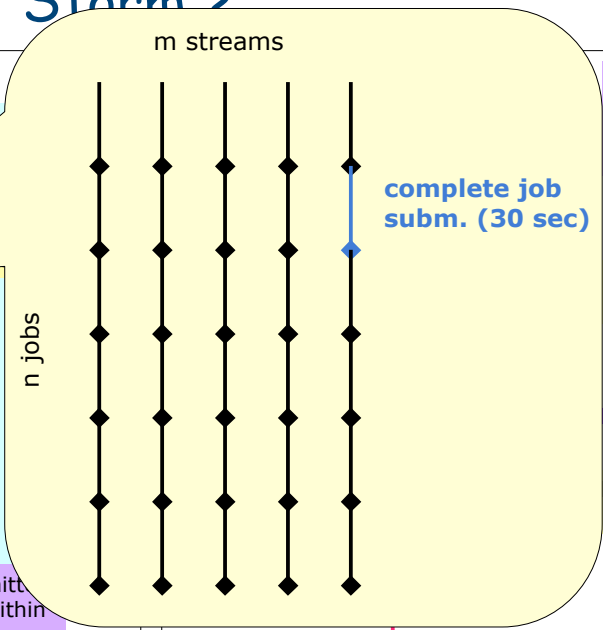


# Why 2 kinds of Storm 2

## CalStorm submission profile



Jobs are submitted sequentially within several streams; Number of streams is adjustable;

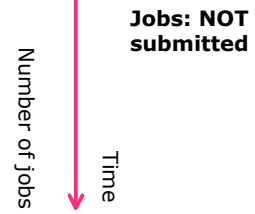


Jobs are spawned independently; Window stack is adjustable;

Window: submission, out. retrieval

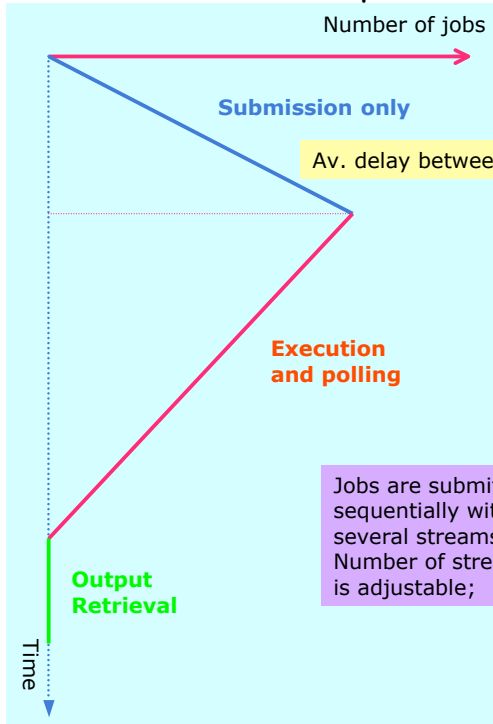
live

Delay between subs: sec (adjustable)



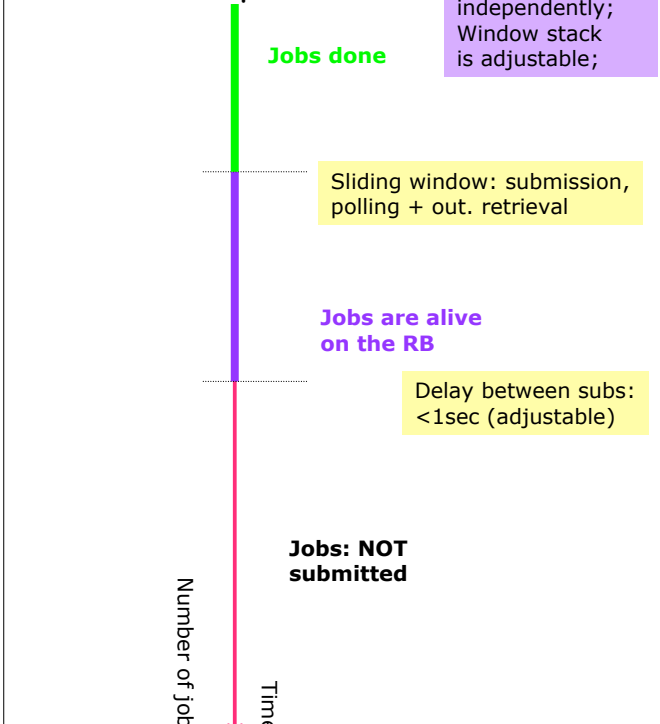
# Why 2 kinds of Storm ?

## CalStorm submission profile



Jobs are submitted sequentially within several streams; Number of streams is adjustable;

## JobStorm profile

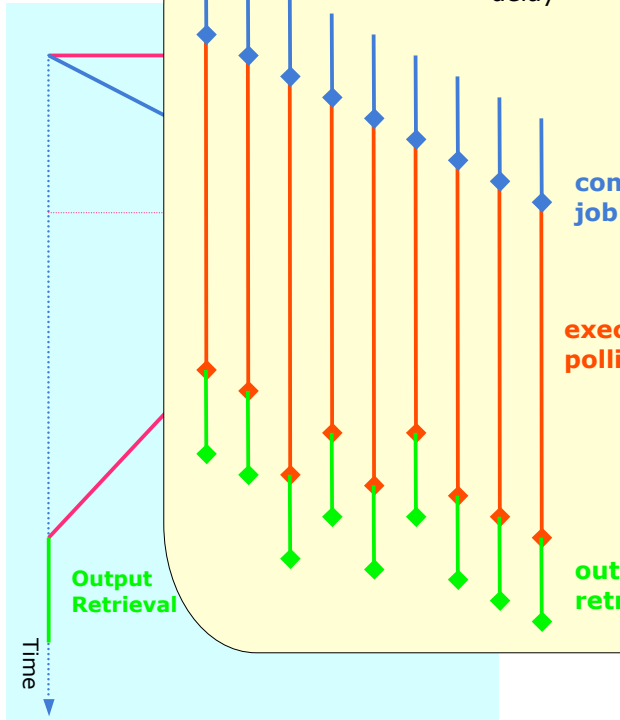


Jobs are spawned independently; Window stack is adjustable;

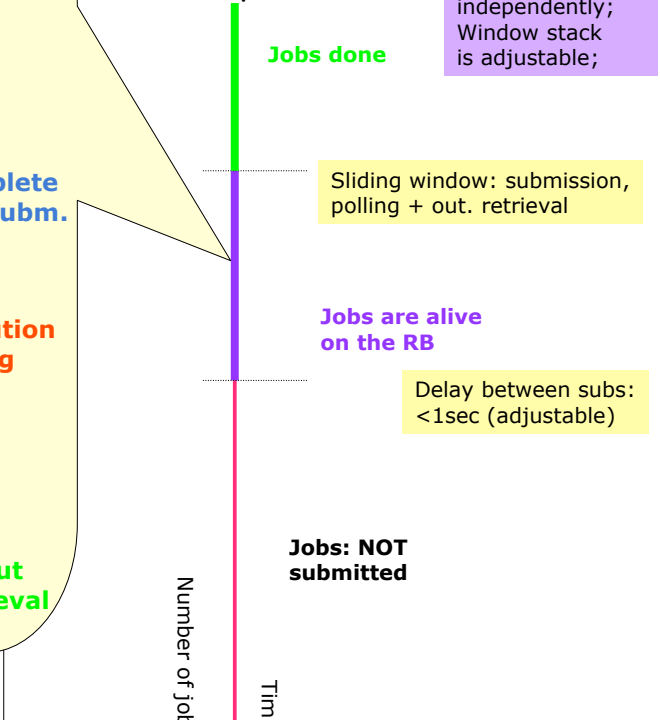


# Why?

## CalStorm

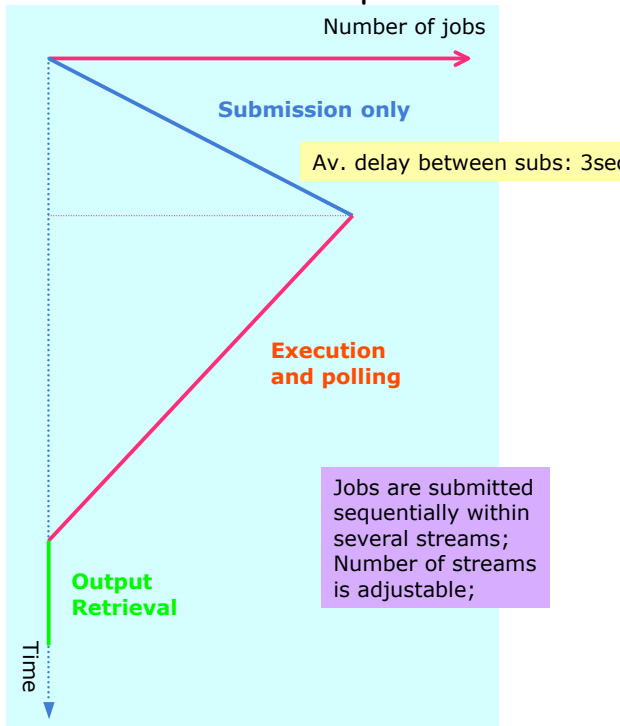


## JobStorm profile

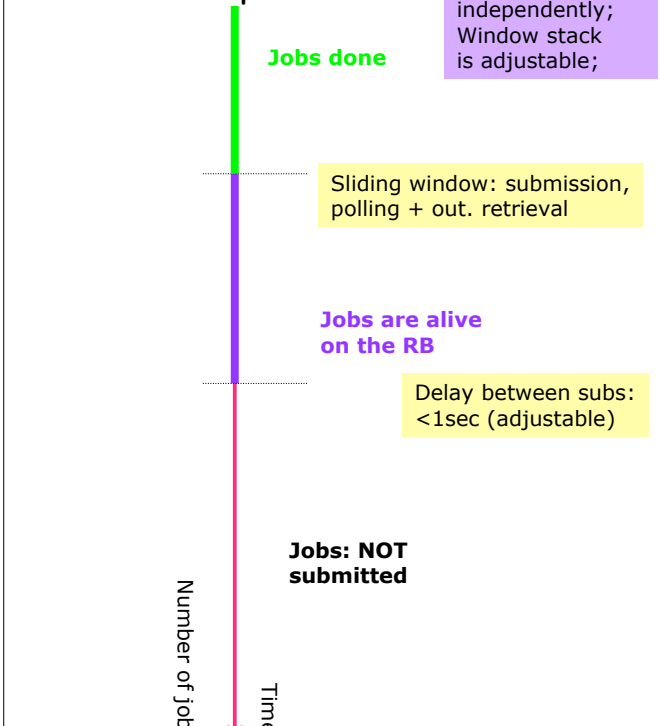


# Why 2 kinds of Storm ?

## CalStorm submission profile



## JobStorm profile





## Similarities and Differences

### ■ CalStorm

- polling adjustable
- **retrycount** adjustable
- myproxyhost selectable

### ■ JobStorm

- **timeout** adjustable
- retrycount adjustable
- possible to **resubmit** a failed job
- CE can be selected
- can work with several RBs in turn
- possibility to clean all ghost jobs
  - previous to the storm

### ■ Similarities

- a very simple Hello script is sent for execution on WNs
  - a sleep time option is available
- the polling period is adjustable (however does not mean the same thing)
- resources can be specified
  - ranking, CPU, other classAD requests, CE selection, ...
- RB can be selected



## The different kinds of Test (3)

### ■ More Storms: the Data Storms now

#### ■ DStorm: Data Storm

- Replica (RM) file transfers from the **WNs**, and check for contents on UI
- Currently testing either of Classic/Castor/dCache SRM-SEs

#### ■ HStorm: David's Storm

- same as above, but using file names with metadata contents
- this allows to spot when a job mistakenly ran several times
  - RB or Condor-G debugging

#### ■ GfalStorm: is a special Data Storm

- Uses Gfal lib to write/read/stat/unlink/close a file from a WN on a remote SE
  - thru a small C application
- Currently testing either of Classic/Castor/dCache SRM-SE

### ■ The storms, when used for submitting few jobs, are also extremely **powerful** to spot configuration or M/W failures on many different components

- they exercise many distributed parts of the M/W, and allow for fine grain debug



## The different kinds of Test (4)

- **MM**: MatchMaking Test for RB
  - exercising either of (file)/gridftp/rfio/gsidcap protocols
  - one of the most important and sophisticated one
- **RLS**: Basic functionality Test for RMC/LRC/RLS
- **SEs**: GridFTPUMask checks for SEs (should be merged with ?)
  
- Deprecated (to be reactivated):
- **ProXyf**: Security Test for RB (stealing proxies - better if failing)
- **MDS**: Consistency checks for (MDS +) BDII (2 tests in sequence)



## The general purpose Options

- `. MainScript --TList="test1 test2 ..."`
  - testX = CEGate CECycle FTP DNS RB SEwsCycle RMCycle
    - also: MDS RLS MM UI\_ST
    - and storms: JStorm CStorm RStorm KStorm DStorm UStorm GfalStorm...
- `. "MainScript --List" :`
  - Prints the List of available Tests.
- `. "MainScript --help" :`
  - Prints this README file, plus the full option List.
- `. "MainScript --MDebug" :`
  - Prints some Variable values from inside the MainScript.
- `. "MainScript --TList="test" --fullHelp" :`
  - To Force printing of a detailed Help about the selected Tests.
- `. "MainScript --TList="test" --showME" :`
  - To Force printing of option values and machine names for the selected Tests.



## More specific top level Options

- Many other options, meaningful only at individual test level, are available
  - though all of them may **NOT** be available for some specific tests (use `--showME` option)
- `. MainScript --TList="test" --forcingTB="yourTBname"`
  - To Force a TB other than "CertTB". This option is mandatory.
- `. MainScript --TList="test" --addOptionList="--opt1=val1 --opt2=val2 ..."`
  - To Provide a list of additional Options to the Tests to be achieved.
- `. MainScript --TList="test" --forceMachineList="node1 node2"`
  - To Provide a list of Machines to be used in the tests, overriding the default
- `. MainScript --TList="test" --addMachineList="node1 node2 ..."`
  - To Provide a list of Machines to be used in the tests, adding them to the default ones
- `. MainScript --TList="test" --forcingRB="fullRBname"`
  - To Provide an alternate RB to work with, overriding the default one provided on the UI
- `. MainScript --TList="test" --forcingVO="otherVO"`
  - To Force a VO other than "dteam". Useful in most of the tests now.
- These are only some of them ...



## The CLI: an Example

- An actual example of a test submission command
  - although the test name is a dummy one :-)

```
MainScript --forcingTB=CertTB --verbose --forcingVO=atlas \  
--forcingRB=lxshare0240.cern.ch --TList=DummyStorm \  
--addOptionList="--reqLapse=1 --maxStack=50 --singleSubmit --maxSubs=20 \  
--pollingPeriod=60 --keepTempDir --circular --useCEList --serie=11022 \  
--selectNodes='lxshare0277.cern.ch lxshare0290.cern.ch' " \  
--forceMachineList="lxshare0236.cern.ch lxshare0278.cern.ch"
```

- In this case, the generic command was:

```
MainScript --TList=DummyStorm \  
--addOptionList="--reqLapse=2 --maxStack=25 --maxSubs=2 --pollingPeriod=30"
```

- it was submitting jobs to all available **CEs**
- it was acting on all **SEs** of the site, by default.

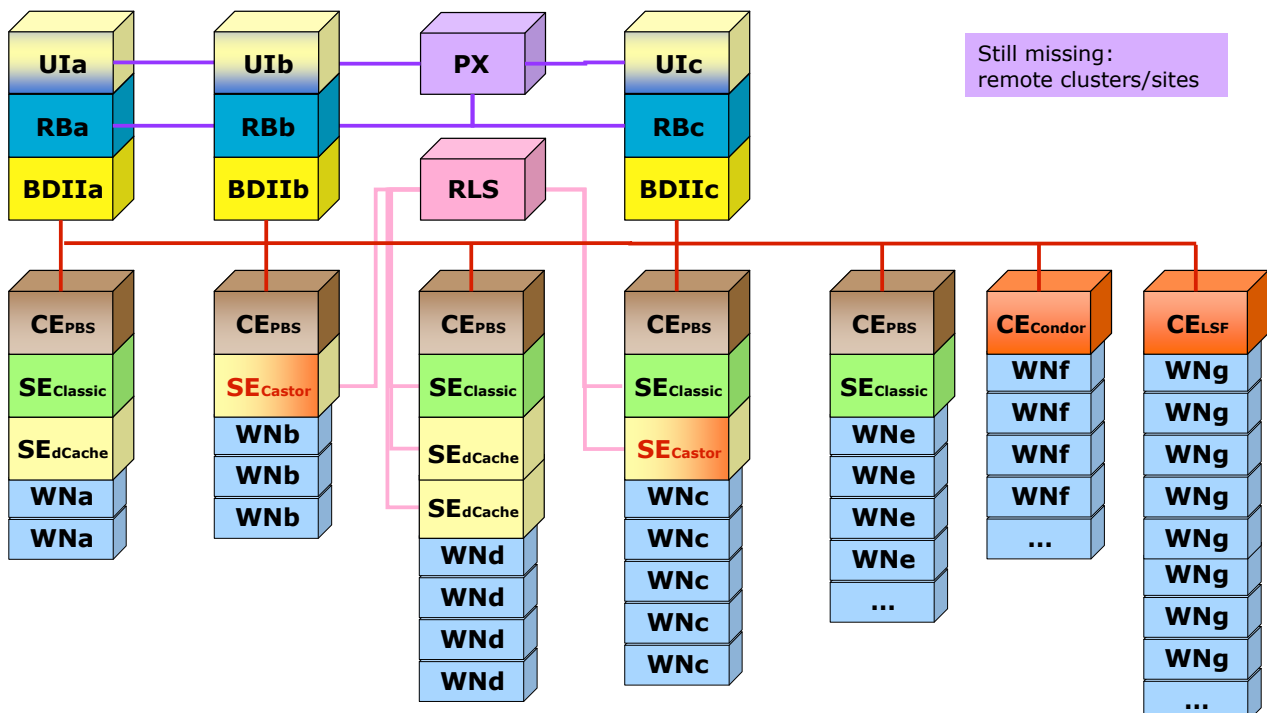


## Detailed example, and other links

- A detailed example of a recent Result Web page, produced on the CertTB (15/05/04, morning) is available in the LCG/TSTG area:
  - [040515-040505 RTest](#)
    - [http://grid-deployment.web.cern.ch/grid-deployment/tstg/validation/040515-040505\\_RTest](http://grid-deployment.web.cern.ch/grid-deployment/tstg/validation/040515-040505_RTest)
  
- This presentation is available in:
  - EdinburghTSTG [[ppt](#)], [[pdf](#)].
    - <http://grid-deployment.web.cern.ch/grid-deployment/tstg/docs/EdinburghTSTG.ppt>
  
- Install help for TSTG RPMs and Tarball:
  - [Install URL](#)
    - <http://grid-deployment.web.cern.ch/grid-deployment/tstg/docs/LCG-Certification-help>



## Certification & Testing Testbed





## Conclusion: Useful or not ?

- Yes, the TSTG suites are **useful and powerful**
  - they are used daily to spot misconfigurations on the CTB
    - each time some new piece of S/W is (re-)installed
  - some specific suites are also run as a daily cron-job
  - other, longer lasting or more stressful are run every week (cron)
  - they most often allow to discover problems or features
    - they are even used to debug development S/W out of the box
  - however, interpretation of results not always straightforward
    - but it was not expected to be the other way round :-)
- **Documentation and Dissemination** must be improved
  - and this talk was part of it
- **Not every kind** of required test is provided yet
  - but new tests are often easy to derive from existing ones
  - thanks to definition files, it is often easy to cover a need by reusing 2 or several existing pieces together
  - new additions will soon be required
    - WP5-SEs, R-GMA, N-MON