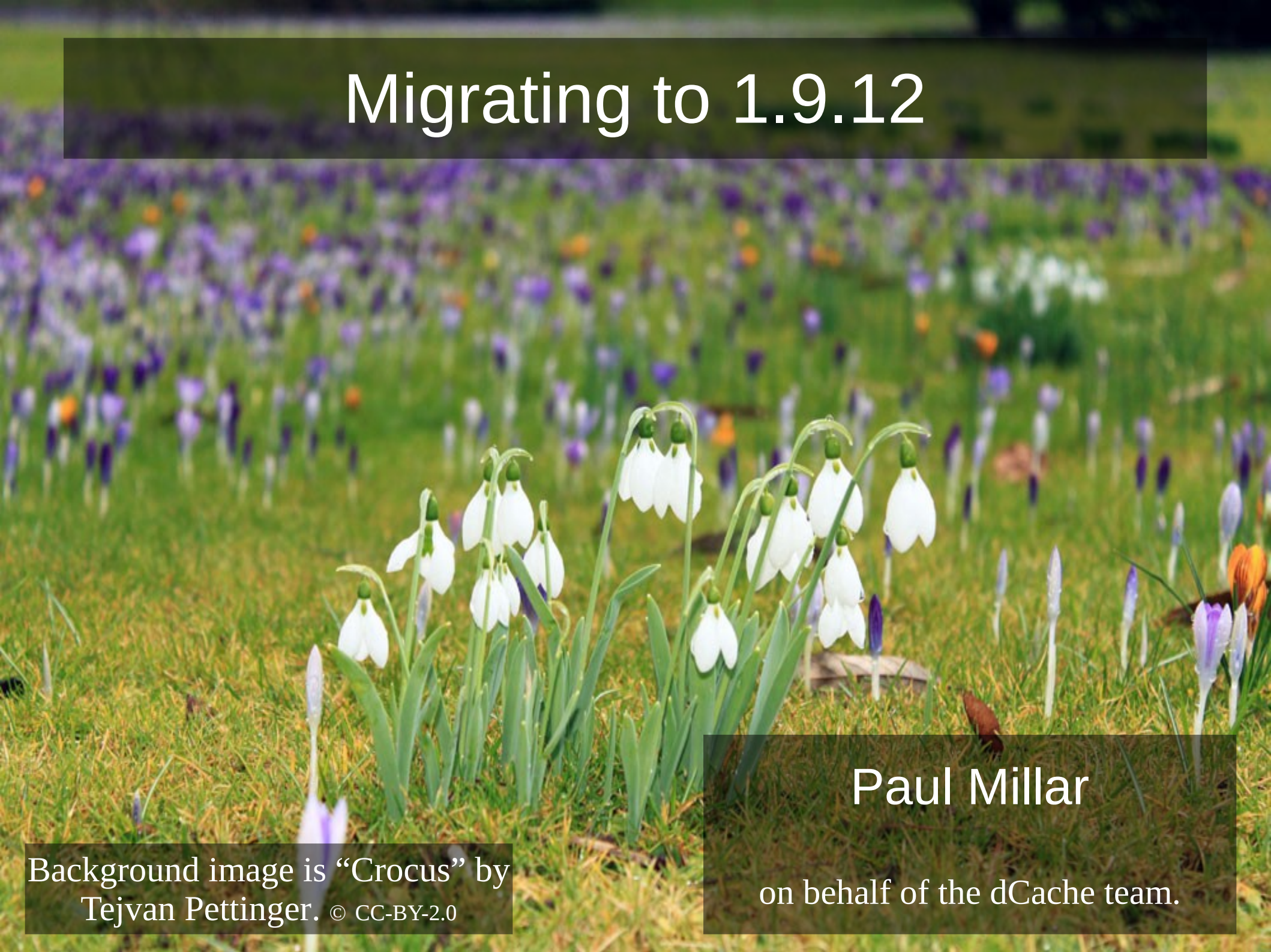
Migrating to 1.9.12



Paul Millar

on behalf of the dCache team.

Overview

The new configuration system

How to migrate

One final thing...

New ideas and concepts

- **Service**: a group of one or more cells
  - Smallest unit that can be deployed
  - Examples: pool, dcapdoor, gsiftpdoor, poolmanager, gplazma.
- **Domain**: a JVM instance (as before)
  - Domains can have any name
    - With 1.9.12 we don't have a naming convention
  - Domain can host any number of services
    - Run a complete dCache (SRM and everything) in a single JVM!

Advantages of new config. system

- More **flexible**:
  - Multiple doors on the same node,
  - Per-domain and per-service configuration,
  - Easy to move services between domains.
- Configuration is shorter and so easier to understand.
- All nodes can share the **same set of configuration files**
- An authoritative source of **default values**
- Allows us to manage the life-cycle of a property

Default values

- No template files any more!
- Files located in `/opt/d-cache/share/defaults`
- These files provide:
  - Descriptions for each property.
  - Authoritative statement on default values.
  - Annotations describing how property should be used.

Do not edit files in **`defaults`** directory. Your changes ***will be lost*** when you upgrade!

Fragment from a defaults file

```
keyStore=${dcache.paths.etc}/hostcert.p12

#  ---- Password for SSL server certificate
#
#   This parameter specifies the password with which the PKCS12 encoded
#   server certificate is encrypted.
#
keyStorePassword=dcache

#  ---- Trusted SSL CA certificates
#
#   This parameter specifies the path
#   the the trusted CA certicates used for
#   in /etc/grid-security/certificates/ ne
#   Java Keystore file before they can be
#   'bin/dcache import cacerts' command to
#   This is used in Webadmin and WebDAV.
#
#   Notice that for GSI the CA cetificates in
#   /etc/grid-security/certificates/ are used directly.
#
trustStore=${dcache.paths.etc}/certificates.jks

#  ---- Password for trusted SSL CA certificates
#
#   This parameter specifies the pass
#   containing the trusted CA cert
#
trustStorePassword=dcache
```

Comments describing this property

Property `keyStorePassword` has a default value `dcache`. A site may alter this value with a similar line in dcache.conf

This default value contains a reference to property `dcache.paths.etc`. Changing `dcache.paths.etc` will automatically adjust `trustStore`.

dcache.conf

- Most properties are configured in this file
- It replaces `dCacheSetup`
- We recommend that this file is identical on all nodes in a dCache instance.
- There's no template file:
  - an empty `dcache.conf` is valid
- Most of the old parameters work as before
- The structure is *almost* the same as `dCacheSetup`

Layout file

- Each host uses a **single** layout file.
- Layout file is located in **`/opt/d-cache/etc/layouts`**
  - If **`dcache.conf`** has **`dcache.layout=foo`** then the file is **`/opt/d-cache/etc/layouts/foo.conf`**
  - Recommend using hostname for layout:

    `dcache.layout=${host.name}`
- A layout file contains:
  - Which **domains** should be started on this host
  - Which **services** should run in these domains
  - (optionally) updated defaults for this host
  - (optionally) domain-specific configuration
  - (optionally) service-specific configuration

Example layout file

```
[doors]

[dCacheDomain]

[srm]
```

Define the domain **doors**

Define the domain **dCacheDomain**

Define the domain **srm**

Defining a domain

Each domain declaration has form:

[<domain>**]**

The domain is called <domain>, which must be a unique name within the dCache instance. The name can be almost anything, but use only alphanumerical characters to be safe.

Example layout file

```
[doors]
[doors/dcap]

[doors/gsidcap]
[doors/gridftp]
[doors/webdav]
[doors/xrootd]

[dCacheDomain]




[srm]
```

There are five services that should run inside domain doors: **dcap**, **gsidcap**, **gridftp**, **webdav** and **xrootd**.

Defining a service

Each service declaration has form:

[<domain> **/** <service>**]**

This says that ***a*** service of type <service> is started in domain <domain>. This may be repeated to start multiple instances of the same service.

Example layout file

```
[doors]
[doors/dcap]

[doors/gsidcap]
[doors/gridftp]
[doors/webdav]
[doors/xrootd]

[dCacheDomain]

[dCacheDomain/poolmanager]
[dCacheDomain/pinmanager]
[dCacheDomain/cleaner]
[dCacheDomain/pnfsmanager]
[dCacheDomain/gplazma]

[srm]
[srm/srm]
[srm/spacemanager]
```

The services that should run in domain dCacheDomain

Define which services should run inside the srm domain

Example layout file

```
dcache.java.memory.heap = 512M

[doors]
[doors/dcap]

[doors/gsidcap]
[doors/gridftp]
[doors/webdav]
[doors/xrootd]

[dCacheDomain]

[dCacheDomain/poolmanager]
[dCacheDomain/pinmanager]
[dCacheDomain/cleaner]
[dCacheDomain/pnfsmanager]
[dCacheDomain/gplazma]

[srm]
[srm/srm]
[srm/spacemanager]
```

New default property values for this host

Updated default values for a node

Declarations at the beginning of are like a mini `dcache.conf` file. They declarations adjust the default values or those configured in `dcache.conf`. The new values will be used in all domains and all services running inside those domains.

Example layout file

```
dcache.java.memory.heap = 512M

[doors]
[doors/dcap]

[doors/gsidcap]
[doors/
[doors/
[doors/

[dCacheDomain]
  dcache.java.memory.heap = 2048M
[dCacheDomain/poolmanager]
[dCacheDomain/pinmanager]
[dCacheDomain/cleaner]
[dCacheDomain/pnfsmanager]
[dCacheDomain/gplazma]

[srm]
[srm/srm]
[srm/spacemanager]
```

Updated property value that apply only to this domain

Updated default values for a domain

Declarations immediately after a domain declaration affect only that domain and all services inside that domain. Other domains (and services running in those domains) are unaffected by the new configuration.

Example layout file

```
dcache.java.memory.heap = 512M

[doors]
[doors/dcap]
  port = 22126
[doors/gsidcap]
[doors/gridftp]
[doors/webdav]
[doors/xrootd]

[dCacheDomain]
  dcache.java.memory.heap = 2048M
[dCacheDomain/poolmanager]
[dCacheDomain/pinmanager]
[dCacheDomain/cleaner]
[dCacheDomain/pnfsmanager]
[dCacheDomain/gplazma]

[srm]
[srm/srm]
[srm/spacemanager]
```

Updated property value for this service only.

Updated default values for a service

Declarations immediately after a service declaration affect only that service. The domain is unaffected by this configuration, as are all other domains and services running in those domains.

Two GridFTP doors

Parameterised domain name

```
[gridftp-${host.name}-1]
[gridftp-${host.name}-1/gridftp]
  cell.name = gridftp-${host.name}-1
  port = 2811

[gridftp-${host.name}-2]
[gridftp-${host.name}-2/gridftp]
  cell.name = gridftp-${host.name}-2
  port = 2812
```

Need to ensure cell names are distinct

Need to specify distinct ports because two doors can't share the same port

Default cell names

Cell names have default values, like all other properties. The default cell name depends on the service. For the **`gridftp`** service, the default cell name is `gridftp-${host.name}`. This is unique provided only one GridFTP door is running on a host.

Two GridFTP doors, one domain

```
[gridftp-${host.name}-1]
[gridftp-${host.name}-1/gridftp]
  cell.name = gridftp-${host.name}-1
  port = 2811

[gridftp-${host.name}-2]
[gridftp-${host.name}-2/gridftp]
  cell.name = gridftp-${host.name}-2
  port = 2812
```

```
[gridftp-${host.name}-1]
[gridftp-${host.name}-1/gridftp]
  cell.name = gridftp-${host.name}-1
  port = 2811

[gridftp-${host.name}-1/gridftp]
  cell.name = gridftp-${host.name}-2
  port = 2812
```

Remove the domain declaration and place second door in the first domain

Pitfalls to watch out for

Duplicate declarations:

```
property.name = value 1
property.name = value 2
```

```
# property.name = value 1
property.name = value 2
```

Reference without braces:

```
cell.name = dcap-$host.name
```

```
cell.name = dcap-${host.name}
```

Recursive references:

```
cell.name = ${cell.name}-1
```

In defaults file:

```
cell.name = dcap-${host.name}
```

In layouts file:

```
cell.name = dcap-${host.name}-1
```

Service before domain:

```
[domain/service]
[domain]
```

```
[domain]
[domain/service]
```

Annotations

- A property's default value declaration (in a defaults file) also says how it may be used:

```
srmPort = 8443
(obsolete)waitForRepositoryReady =
(deprecated,not-for-services)logArea =
```

Annotation	Description	Effect when configured
deprecated	A property that will be retired in a future major version of dCache	Warning message is printed in log file.
obsolete	A property that is no longer supported, but isn't critical to dCache	Warning message is printed in log file.
forbidden	A property that is no longer support and was critical to dCache	Error message is printed and dCache refuses to start.
not-for-services	A property that has no effect if configured for a service.	Warning message is printed in log file if used in a service

check-config

- New command added in 1.9.12:

  `/opt/d-cache/bin/dcache check-config`

- Checks:
  - the `dcache.conf` and layout file is structured correctly.
  - properties are used according to annotations.
- Two types of message: warning and error
  - Warning: a problem but dCache will start
  - Error: a problem that prevents dCache from starting.
- Check-config will list all warnings and errors.
  - Starting dCache will also list warnings and errors, but will stop after the first error.

The migration process

- How to migrate a single node.
- Migrating a dCache instance.
- What next?

How to migrate a node

- Practice on a test machine first!
- Shutdown dCache
- If running Chimera NFS server:
  - Shutdown Chimera NFS server.
  - Remove the init-script.
- Upgrade dCache RPM
- Run the migration script
- Fix any remaining problems
- Start dCache

The migration script

- **`/opt/d-cache/libexec/migrate-from-1.9.5.sh`**
- Provides 1.9.12 configuration:
  - Reads **`dCacheSetup`**, **`node_config`** and **`*.poollist`** files
  - Writes **`dcache.conf`** and **`layouts/`**<hostname>**`.conf`**
- Handles many (but not all) complications:
  - Multiple definitions: the migration script comments out all but the last assignment of a property
  - Filtering out configuration if the property value is the same as **`dCacheSetup.template`**
  - Migrates configuration from old property names to the new names

Known limitations of migration script

- Doesn't handle if <domain>`Setup` isn't a symbolic link to `dCacheSetup`.
  - See worked example in hands-on session
- Doesn't handle modified batch files.
  - Our advice was not to modify batch files.
  - Some sites found they had to (due to our previous, inflexible configuration system)
  - With 1.9.12, sites shouldn't need to modify batch files
- Some property configurations require manual attention ...

Complication #1: java property

- In 1.9.5, the `java` property must be defined
- In 1.9.12, the `java` property must not be defined
  - By default, dCache searches for the java binary in $PATH.
  - This may be overridden by setting `JAVA_HOME` or `JAVA` environment variables in either `/etc/dcache.env` or `/etc/defaults/dcache`.
- Currently, migration script doesn't fix this problem, manual intervention is needed.
- Future version of the migration script will handle this automatically.

Complication #2: java_options

- `dCacheSetup.template` file contains `java_options` property
  - Sites running 1.9.5 have the Java command-line hard-coded in their `dCacheSetup` file.
  - Changing memory usage requires adjusting this property
- With 1.9.12, this is now `dcache.java.options`
  - Don't configure this property (use other properties)
  - `java_options` property is ignored
- Migration script currently doesn't fix this problem; you need to convert `java_options` value into `dcache.java.*` properties.

Migrating a dCache instance

- Smaller sites can upgrade all nodes in one go.
- Larger sites can do a 'rolling upgrade'
  - Update head nodes:
    - Go into “down time”
    - Update all doors and central nodes.
    - Verify working ok.
    - Come out of “down time”.
  - Go into “at risk” and migrate each pool, one at a time.

What can we do, after migrating?

- **Consolidate domains**: run services in the same domain.
  - Think which services need to be restarted independently.
- per-Domains **adjust the required memory**
  - Free up memory for system IO cache.
- **Common configuration** deployment
  - e.g., sites share the same set of config files across nodes using AFS, NFS, or simply rsync.

One final thing...

New info-provider

- Configuration has moved:
  - **`etc/info-provider.xml`** new location
  - File contains only your configuration
  - No template file: valid file supplied with the RPM
  - **`etc/glue-1.3.xml`** is ignored (please delete it)
- Transformation information is now separate:
  - Located in **`share/info-provider`** directory
  - **Changes to these files will be lost when upgrading**
  - If you need to edit these files, let us know.