INTRODUCTION

dconf is a key/value storage system with expressive types, a very simple API, and can be accessed through the D-Bus inter-process communication system.

dconf
lightweight configuration storage

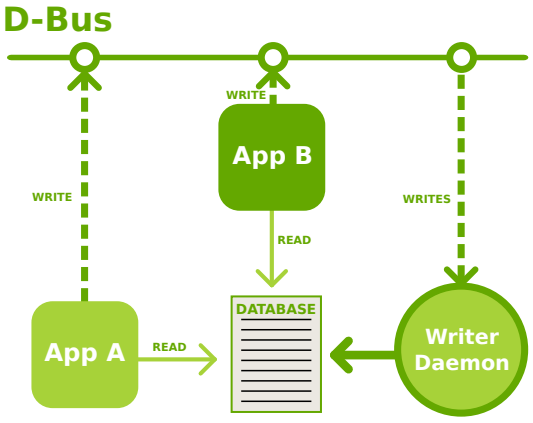
With dconf you can centralize your system and user applications configuration into a database allowing easier administration and configuration backups. Applications can subscribe to changes in any key in the database, allowing applications to react in real time to changes in the configuration.

KEY FEATURES

* Licensed under the LGPL version 2.1
* Simple to use API
* Key change notifications through D-Bus
* Low memory footprint
* High performance key reads and writes
* Reduced disk access for optimized energy consumption
* Support for different access levels (system, read-only, mandatory...)
* Schema based high-level API for applications (GSettings)

ARCHITECTURE

dconf stores its configuration database in three different files; a system wide database, a defaults database and a per user database. Read operations are performed by directly accessing the database file which has been designed to allow multiple reads even in the presence of a writer. This achieves very high performance for read operations.



The writes are performed through a 'writer' service that is started on demand when a process needs to write to the configuration database. Configuration change notifications are sent by the writer which exits after a period of inactivity and can be resumed later without any checkpointing.

GSettings

GSettings is an abstract interface to settings storage that is on track to become the standard interface for all GNOME applications wishing to store settings. GSettings will soon be part of the GLib platform.

On Free Desktop systems, GSettings uses dconf as the backend in which to store application settings, on Windows, GSettings uses the registry as the backend and a native backend is planned for Mac OS.

GSettings has a very strong schemas system. Schemas are mandatory and provide very strong guarantees to the application. With GSettings, the schema is not stored in the configuration database but is deployed as part of the application. Application writers write one schema file and it provides a consistent settings interface to the application regardless of the platform on which it is running.

GSettings uses a local caching mechanism to allow for both reads and writes to be synchronous and non-blocking, and without mainloop reentrancy. This, coupled with its schema system, makes GSettings a much nicer interface to program to than using dconf directly. It is for this reason that, even in light of the GConf compatibility layer, any new application development should target GSettings rather than the GConf API. Some existing applications are even being ported to take advantage of the new API.

A COMPARISON OF dconf AND GConf

For many years GConf has been seen as one of the cruftier APIs in GNOME. A redesign has been talked about for many years, including by the authors of GConf. GConf has meanwhile fallen into a relative lack of maintenance.

GConf was designed to use the ORBit remote object system, which in turn was based on CORBA. These technologies have been marked for deprecation in the GNOME stack for quite some time, and are now in the process of being actively removed from use as part of the GNOME 3.0 plans.

A port of GConf to use D-Bus was done some time ago, but the authors of this port have stated that it isn't suitable for use in general and it is not to be considered for upstream use without a considerable rewrite.

The current plans for the GNOME 3.0 platform include dconf as a drop-in replacement for GConf. An API compatible replacement for libgconf using dconf is being prepared for this purpose.

One of the strangest concepts that people approaching GConf are faced with learning is its schema system where XML files must be hand-written and are rather hard to deploy, many programmers avoid the use of schemas entirely. dconf has no schemas, it is purely a database that stores key/value pairs.

PERFORMANCE

dconf exceeds the performance of GConf by an order of magnitude or more in most ways related to the reading of configuration data.

From a warm cache, dconf can enumerate the contents of the entire database and load all the keys in approximately 10ms (0.01 seconds). The equivalent operation in GConf takes approximately 482ms. This is an increase of approximately 50 times. The reason for this is that reads from the dconf database don't involve going out of process and making a round-trip to another process.

For dconf to read the entire database from a laptop hard drive with a cold cache takes 0.2 seconds. The equivalent operation in GConf on a reasonably fresh install takes approximately 3 seconds. This length of time increases with the age of the user's home directory since the GConf database tends to become more fragmented over time. Anecdotally, 10 seconds or more has been witnessed.

It is not possible to fairly compare write operations between dconf and GConf as they operate on different writing models. With dconf all writes are written atomically to disk and fully synced to the harddrive before the request is finished. With GConf writes are grouped and periodically flushed. It is possible to write many keys at once in dconf whereas transaction support in GConf exists in the API but is not actually implemented as such.

MEMORY USAGE

The dconf writer service uses approximately 216kB of writable memory compared to the 3.2MB used by gconfd. Additionally, the client-side dconf library is very slim in terms of per-process memory consumption. The majority of its memory usage comes from D-Bus and,as D-Bus is already used by many components on a standard Linux system, this overhead does not have a significant impact in most cases.

Perhaps the greatest benefit in memory use that comes with dconf is that the entire configuration database is stored in a binary file format that is memory-mapped and used directly. This means that your applications are sharing the kernel's disk cache memory without making a duplicate copy.

An additional benefit is that the dconf writer service need not be running at most times. During startup, when nobody is making changes to settings, the writer isn't running. Only when someone modifies a setting is the writer activated (by D-Bus). The writer is stateless, so it can easily be programmed to exit after an inactivity timeout. This reduces the number of processes that need to be running.

OVERALL COMPARAISON

	GConf	dconf
Warm cache read time	482ms	**10ms**
Cold cache read time	3s	**0.2s**
Daemon writeable memory size	3.2MB	**216kB (or zero)**

**** All perfomance tests were made against a typical desktop configuration database containin 2428 keys. The keys were identical in both databases. The tests were made on a Core 2 Duo laptop running at 1.87GHz and all software was compiled using GCC 4.4 with -O2.*

GConf API BRIDGE

Work is currently beginning on an API compatibility bridge between dconf and GConf. This bridge will provide a GConf API and possibly ABI compatible interface to existing applications that want to use dconf with little or no porting effort.

This bridge will essentially allow dconf to be used as a drop-in replacement for GConf on any desktop or device.

FUTURE PLANS

* Improve network stored database support (NFS, CIFS...).
* Cross platform database migration.
* Support running dconf apps in the absence of D-Bus.
* Database snapshots and rollbacks for backup purposes.

FURTHER DEVELOPMENT

Codethink Ltd. is the main developer and consultant of the dconf project. If your organisation needs assistance on the deployment and integration of dconf in any of your applications, platforms or devices, our engineers can help you to get the best out of this innovative configuration management system.