Recommendations for 4D applications

Last update: April 9, 2018

Author: Cédric Gareau

Summary

1. Introduction 3
2. 4D applications 3
3. Port number 3
4. Processor 3
5. Operating system 5
6. Memory 6
7. Hard disk 6
8. Backup / Maintenance operations 6
9. Network 7
10. Web 7
11. Physical or virtual machine 7
12. Manufacturer / Model of machine 8
13. Tests / Acceptance / Deployment 8
14. Security 8
15. Server monitoring 11
16. Deploying your compiled applications 12

1. Introduction

This document will be updated regularly and is not meant to be considered as comprehensive. We will be happy to include any contributions you provide us.

2. 4D Applications

Recommended:

- Use Unicode mode
- Compile the database in multi-target (32- and 64-bit)
- Deploy the 64-bit version
- Build a compiled structure of your application

3. Port numbers

Recommended: make sure that the following ports are unique

- mandatory ports:
  - client/server publication port (modifiable in database settings, by default it is 19813)
  - application port (not modifiable, pubication port +1 so by default it is 19814)
  - SQL server port (modifiable in database settings, by default it is 19812; even if you do not use the SQL language, 4D will check whether this port is free on startup)

- optional ports:
  - Web server port(s) (modifiable in database settings, by default it is 80 (HTTP) and 443 (HTTPS))
  - port for PHP interpreter (modifiable in database settings, by default it is 8002 at the address 127.0.0.1)
  - other ports can also be used by 4D, more particularly by the 4D Internet Commands plugin (http://doc.4d.com/4D-Internet-Commands-12.1/Appendixes/Appendix-B-TCP-Port-Numbers.300-506079.en.html)

4. Processor

Recommended:

- several cores
- use preemptive mode (available beginning with 4D version v15 R5 - executed in 64-bit compiled mode only)
- explicitly declare all methods that you want to be started in preemptive mode (check the option in the method properties and verify their eligibility using the compiler)

Since version v11 SQL, 4D benefits from a system of multiple cores (*). The operating system dispatches the "processor time" (total time of all cores) between each application and between the threads of each application. Then it's the application that sets the priority of each of these threads, with each thread working individually.

**: Systems that support multithreading, which allows simulation of 2 logical cores for each core, thus doubling the processing capacity of 4D Server.*

On the other hand, in 4D there are cooperative threads and preemptive threads. 4D uses them automatically (no software or specific preference) according to the 4D executed.

All the cooperative threads function in the main thread unlike preemptive threads which are dispatched by 4D server onto other threads. When run in preemptive mode, a process is dedicated to a CPU. Process management is then delegated to the system, which can allocate each CPU separately on a multi-core machine.

When run in cooperative mode (the only mode available in 4D until 4D v15 R5), all processes are managed by the parent application thread and share the same CPU, even on a multi-core machine.

As a result, in preemptive mode, overall performance of the application is improved, especially on multi-core machines, since multiple processes (threads) can truly run simultaneously. However, actual gains depend on the operations being executed. Basically, code to be run in preemptive threads cannot call parts with external interactions, such as plug-in code or interprocess variables. Accessing data, however, is allowed since the 4D data server supports preemptive execution.

In return, since each thread is independent from the others in preemptive mode, and not managed directly by the application, there are specific constraints applied to methods that you want to be compliant with preemptive use. Additionally, preemptive execution is only available in certain specific contexts.

> *Notes:*
> *• A new type of process, called a Worker, allows you to exchange data with any other process, including preemptive ones.*
> *• The new command "CALL FORM" provides an elegant solution for calling interface objects from a preemptive process.*
>
> *Although they were principally designed to meet needs related to interprocess communication in the context of preemptive processes (accessible in 64-bit versions only), the "CALL WORKER" and "CALL FORM" commands are available in 32-bit versions and can be use with processes in cooperative mode.*

Keep in mind that a 4D Remote non-local process always communicates with two twinned threads on the server: one preemptive thread for DB4D requests (create, save, load, delete, order by, query, etc.) and one cooperative thread for application requests (current date(*), GET PROCESS VARIABLE (-1;..), etc.). A third preemptive thread can also be created if you execute SQL commands (when the "Begin SQL" command is called).

Depending on the command executed, 4D uses a particular thread for establishing communication on the corresponding port:

- publication port for DB4D requests (cooperative thread)
- application port (publication port +1) for application requests (preemptive thread)
- SQL port for SQL requests (preemptive thread)

In conclusion, we strongly recommend having a multi-core processor even if the use of all the cores depends on the 4D commands used in your application keeping in mind that the more preemptive you are, the faster you will be on a multi-core machine.

5. Operating system

Recommended: 64-bit OS certified by 4D

Important: You must deploy 4D on an OS certified by our Quality Assurance department. The certification matrices are available on our web site: http://www.4d.com/support/resources/compatibility.html.

- Under Windows:
  - Starting with 4D version v15, for 4D opt for **Windows 10** and for 4D Server or your TSE servers, opt for **Windows Server 2008 R2** (the 4D v14 product range and higher are not compatible with Windows Server 2008 Standard) or for **Windows Server 2012 R2** (4D Windows versions are not compatible with the 'Server Core' option of Windows Server);
  - Deploy 4D Server on a dedicated machine (under Windows Server, uninstall all the roles, particularly the file server role that is installed by default);
  - Deploy the 64-bit product range:
    - 4D Server is available in a 64-bit version starting with version v12,
    - 4D Developer is available in a 64-bit version starting with version v15 R5;
  - For performance reasons, we also recommend that you disable the local security strategy "Microsoft network client: digitally sign communications (if server agrees)" in the security options, which reduces performance by up to 15% when it is enabled.

- Under Mac:
  - Deploy the 64-bit product range:
    - 4D Server is available in a 64-bit version starting with version v15.1,
    - 4D Developer is available in a 64-bit version starting with version v15 R5.

64-bit versions allow your 4D stand-alone applications, as well as your 4D remote applications, to take full advantage of the power of 64-bit operating systems. The main advantage of 64-bit architecture is that more RAM can be addressed.

Although widely rewritten, 4D 64-bit applications are highly compatible with the current 4D databases. However, since they use the most recent technologies, we needed to update some features, as well as to stop supporting others. All these evolutions are detailed in the Differences in the 64-bit releases section.

On the other hand, implementing the 64-bit architecture provided us with the opportunity to support new, powerful features such as the ability to handle multithreading processes, OS-based printing architecture or cutting-edge Quick Report and Label editors.

6. Memory

Recommended:

- provide enough memory in the machine for the cache and engine memory of 4D Server + the operating system itself
- deploy the 64-bit version of 4D Server (a 32-bit application cannot use more than 4 GB of memory whereas a 64-bit application has 8 TB of theoretical address space!)
- determine the ideal cache value for your database in production by using the "4D_Info_Report" component (http://taow.4d.com/Outil-4D-Info-Report/PS.1938271.en.html)

(Plan to leave some free slots if the data file is likely to grow rapidly so that you can add additional memory if needed)

7. Hard disk

Recommended: 1 high-performance SSD disk to store the 4D database + 1 high-capacity disk for backups

> Warning: you must look at read and write speeds before purchasing the SSD disk. It's better to get a smaller but high-performance disk.
> Ideally, you would be able to store the system, 4D Server and the database on it.

A 2nd high-capacity disk is recommended for storing database backups and log files (and also to have a copy of the database in case the SSD disk crashes).

8. Backup / Maintenance operations

Recommended: RAID + 4D Backup + log file + regular compacting of data file

We recommend using the backup mechanisms of 4D and that you enable the log file.

Performing regular backups of your data is important but when there is an incident, it is does not allow you to recover any data entered since the last backup. In order to meet this need, 4D has a special tool: the log file. This file ensures the permanent security of the data in your database. Moreover, 4D works constantly with a data cache located in memory. Any change made to the data of the database is temporarily stored in the cache before being written to the hard disk. This accelerates the functioning of applications; accessing the memory is much faster than accessing the disk. If an incident occurs in the database before the data in the cache could be written to the disk, you must integrate the current log file in order to fully recover the database.

In addition to 4D backups and the log file, we also recommend that you plan regular maintenance operations to compact your data and index files.

To improve failure tolerance, security and/or overall performance, implementing a RAID system is a very good choice:

- for databases: the best choice is RAID 10 (security and performance)
- for backups: the best choice is RAID 5 (price and security)

Once your backup strategy has been set up, we recommend that you consider the worst case scenario (fire, theft) and consider performing a weekly copy of your database on a removable support and storing it in a separate secure location.

9. Network

If you use the 4D Web server and you want to separate the traffic for safety reasons, we recommend providing 2 network cards: one for 4D Remote users and the other for Web requests.

10. Web

Recommended:

- use 64-bit version of 4D
- use 4D Server or 4D in local local (preemptive mode is not supported by 4D in Remote mode)
- use a compiled database
- having a maximum of database methods and projects methods related to Web being confirmed as thread-safe for 4D Compiler
- in the database preferences for Web server:
    - check the option "Use preemptive processes" to enable this mode
    - check the option "Use the 4D Web cache" and set the cache size to 524 288 Kb
    - set the Inactive Process Timeout to 8 hours and enable the option "Automatic Session Management" for the users be able to reuse the same context during the day (in case you don't manage them by programming)
    - check the option "Use Keep-Alive Connections"

Since 4D v16, the built-in Web Server (64 bits only) for Windows and for MacOS allow to fully take advantage of the multi-core by using the preemptive Web processes in compiled applications. Most of 4D commands related to Web, methods and URLs, are thread-safe and can be used in preemptive mode.You can set up your code related to Web, including HTML 4D tags and Web based methods, in order to simultaneously execute on as many cores as possible.

11. Physical or virtual machine

Recommended: physical machine

Even though we recommend a physical machine for 4D Server, we do have a certain number of clients who work with virtual environments, and their numbers are on the rise. We even have clients

who have virtual TSE servers on blade servers. The advantage of this solution is the flexibility for assigning resources (CPU, memory).
However, we do not have official documents certifying the functioning of 4D in a virtual environment.

Certain recommendations must therefore be taken into account concerning virtualization such as that of ensuring that machine performance is adequate in terms of resources (memory, processor, etc.) or that the virtual operating system be certified with the version of 4D installed.
Generally, CPU and RAM resources must be slightly higher in a virtual environment with respect to a non-virtual solution.
In addition, the performance observed depends largely on how the virtual machine is configured (CPU, memory, etc. but also the network card (characteristics, shared or not, dedicated, correctly configured, etc.)), on the solution used, the version of the solution and the system on which the solution is installed. For this part, we recommend consulting the sites and forums of the virtual solution providers as well as comparative studies.

12. Manufacturer / Model of machine

We have no particular recommendations. However, you must look at the hardware of the machine in detail before purchasing it in order to make sure that its characteristics correspond to your needs as well as to the above recommendations.

13. Tests / Acceptance / Deployment

Do not deploy your database without having tested it beforehand in its future environment or in a similar environment (hardware, identical 4D version, etc.) and especially under its future conditions of use (with the same number of simultaneous users or using test scenarios written by the users). Stressing your database in order to know its limits which may not be detectable by the development team will spare you many problems once you are in production and will allow you to optimize the functions that are used the most.
Tests and acceptance are the keys of a successful deployment!

14. Security

Source : https://blog.4d.com/security-data-protection/

Security is an **important and fundamental topic** for a database or business solution system. This article proposes an overview of how 4D protects your data. In fact, security is about **data protection**. And data protection is a huge area. Data needs to be protected for unwanted access, but also for loss.This is an important fact, as most users think only about protecting for unauthorized users, not about **protection for events** such as power failure, damaged hard disk, accidental data modifications and so on.

Security and data protection is a very wide area: it starts by user authentication, goes to external access *(such as web or SQL)*, unwanted code execution (SQL injection, script inspection attacks), then security updates, backup and more.

4D Server

4D Server is an integrated Client/Server development system, optimized to build robust business applications with an embedded database system. While 4D can send out data (with standards such as HTTP, SOAP, ODBC or OCI) or can be accessed from the outside (with HTTP, SOAP, ODBC/SQL), the main usage is based on the **internal development language "4D"**, using an internal, proprietary network protocol to communicate between the business client and the server.

The network communication supports TLS 1.2 encryption, either using a **predefined key** (no SSL certificate required) or alternatively a **customer provided key file**.

The tight binding between development language and network communication allows a high level build in protection concept, avoiding typical attack scenarios such as SQL injection or buffer overflow.

The **4D language** is a powerful and mature language, perfectly designed to build business application systems. It consists in more than a 1500 commands, covering database operations *(order by, query, creating, transactions and so on)*, printing, communicating with other devices or computers, document management, window or user interface commands and much more. Take a look at the 4D language manual for more details.

The language itself is **tokenized**, even in **interpreted** (development or prototyping) mode, it is never executed as text evaluation. In production mode the language is **compiled**, with automatic range checking protection for buffer overflow attacks.

4D Web Server

4D feature its own build-in HTTP Server, a powerful, multi-threaded server for both static and dynamic content. The tight integration has a **drastic impact** on increased security.

Beside better code security (see below), this concept removes the typical forgotten update problem. As all is integrated, there is **only one software to update** *(see "Software Update Section" for additional details)*. Normal solutions requires a huge amount of software packages to update: from PHP, OpenSSL, Apache, NodeJS and so on… All needs regular updates and it is common that some parts stay unpatched for a long time, especially if used as department solution, without a specialized IT team.

Web requests triggers 4D code, which responds to the request on business application level, not just on database level. The tight integration allows to control every request, using build in authorization or customized implementations, of course **TLS encrypted**.

The build-in HTTP Server also allows fine control justifications, by example for a REST Server.

4D v16 R6 now supports Perfect Forward Secrecy (PFS). This gives you the highest security rate for your communications, by default! Beyond the protection it offers, the support of PFS also inscreases the SSL veririfcation tests results, which is excellent for our customers. In particular, for the customers working with sensitive information.

The default security level of the 4D Web Server has been increased to comply with certain network security functions (App Transport Security (ATS) on iOS for example), and allows to get better results for Web Security verification tests (for ex. SSL Labs).
So we have:

- enabled « Perfect Forward Secrecy » and
- disabled the RC4 algorithm from the ciffer list

Therefore, the 4D Web Server gets an "A" in the SSL Labs rating, with no need of any action!

Perfect Forward Secrecy (PFS) is a key exchange algorithm. It uses Diffie-Hellman (DH) algorithms to generate session keys and only the two parts implied in the communication can get them.4D automatically enables PFS when TLS is activated on the server. For this 4D generates a « dhparams.pem » file -- if it does not exist yet -- containing the private key of your DH server. If you use the 4D standard ciffer list, PFS is ready to use. If you prefer to use a custom cipher list, check that it contains entries with ECDH or DH algorithms.

To know if PFS is enabled on your Web Server, execute the command « WEB Get server info » with the new attribute « perfectForwardSecrecy ». This allows to check if all needed conditions to use PFS are met:

- TLS is enabled
- The cipher list contains at least one ECDH or DH algorithm
- The « Dhparams.pem » file exists and is valide
- All SSL/TLS certificates are present

The RC4 algorithm has had security issues and is now deprecated in 4D Web Server. All RC4 encryptions have been removed from the default ciffer list and the RC4 and the "!RC4" model has been added to this list to explicitly exclude it.

SOAP Server

Similar as the HTTP Server, a SOAP Server is built-in, allowing **detailed access control**, based on business objects (not just database level).

4D SQL Server

While data access for 4D Remote by default goes through a property protocol, SQL access (natively or via ODBC), is supported as well. In addition, there are open source PDO (PHP Data Objects) drivers available. SQL access to the database level can be controlled with password system, SQL schemas and fine controlled using SQL views.

4D's build-in password system

4D’s build-in **user authorization system** can be replaced by 3rd party systems. 4D supports the direct usage of Microsoft Active Directory and LDAP, as well as fully customized systems.

Software update mechanism

Modern software might be a complex combination of software products, database server, middleware, application server, web server and more. It’s easy to forget to keep all the pieces up-to-

date, like an OpenSSL DLL for instance. 4D reduces this problem in many ways, not just helping the admin with his daily life but **reducing the risk by design**.

As **integrated all in one solution**, it is only one folder to replace. Everything is installed in a single folder, it could be even replaced with a drag&drop process. Making it simple avoids the *"I'll do it later"* syndrome. With a single replace, all parts of the business application is updated in one step, **nothing can be missed**.

The server can be updated fully automatically. The update process is neither controlled nor forced by 4D itself, it is **fully in the hand of the developer of the solution**.

Backup and journaling system

4D provides out of the box a transactional-based journaling system. Every single data modification operation is **logged** and can be **rolled back**. In case of an emergency case, the work of the day can be restored – nothing is lost. In case of an interruption, database is automatically checked on restart and missing operations (kept in memory, not stored to disk yet) are restored, to have the database back with all information. Even in case of a total data corruption (bad disk, etc…), the data file is **automatically restored** from last full backup and the journal including the daily work is integrated.

The transaction journal can also be useful in case of accidental deletion (or sabotage record manipulation) as well, both for forensic and data recovery.

Standard backup is part of the 4D product, no additional licensing is required, just an additional hard disk is needed (to protect for disk failures).

In 24/7 environments, 4D supports the usage of **cascaded and/or star mirror systems**. A production, a mirror and a secondary mirror builds a cluster of systems to provide services around the clock. An additional mirror system could be run in another city or cloud to **protect the data even in extreme disasters**.

In parallel to transactional-based journaling 4D supports snapshots of virtual machines as well (VSS Writer).

Additionnal protection

All standard protection concepts, such as **server room protection** or using **encrypted hard disks** (hardware solutions like encrypted SSD or software solution like Bitlocker) are of course recommended as well.

Please also see our Security Guide: https://blog.4d.com/4d-security-guide/

15. Server monitoring

You can use the « **4D_Info_Report** » component to get usefull information :

- regarding the system, hardware and 4D environment
- regarding the databse: structure, data, triggers, indexes, custom settings used,...
- in real time: memory, cache, connected users, processes,...

This component can be used in production: the impact on performances of the monitored application is meaningless.
You can also use the "Monitor" tab of the 4D Server Administration Window to display dynamic information related to the database usage as well as information on the system and the 4D application.
The graph area allows to watch the evolution in real time of several parameters:

- the rate of used processes,
- the network traffic,
- the memory usage.

This information can be returned by programming with the command « **GET ACTIVITY SNAPSHOT** ». This command is used to get a snapshot of the *x* operations that are most time-consuming and/or run most frequently, such as cache writing or the execution of formulas.

By default, **GET ACTIVITY SNAPSHOT** processes operations performed locally (with 4D single-user, 4D Server or 4D in remote mode). However, with 4D in remote mode, you can also get a snapshot of operations performed on the server: you just need to pass the asterisk (*) as the last parameter. In this case, the server data is recovered locally.
The * parameter is ignored when the command is executed on 4D Server or 4D single-user.

An other new command in 4D v16R4 (modified in v16 R5) is « **Get process activity** ».

The **Get process activity** command returns a snapshot of connected user sessions and/or related running processes at a given time. This command returns all processes, including internal processes that were not reachable by the PROCESS PROPERTIES command.

16. Deploying your compiled applications

We recommend to always deploy your 4D applications as compiled applications.

Before generating a compiled version, we advise to check syntax and, in the database properties, to set the following options:

- enable "Range checking"
- enable "Generate the symbol file"
- enable "Generate error file"
- enable "Multi-target compilation (32-bit and 64-bit)"
- select "All variables are typed" for Compilation Path
- select "Long integer" for Numeric
- select "Long integer" for Button