BC & JA

FastCGI for OpenVMS

FastCGI is a protocol for interfacing interactive programs with a web server. FastCGI is a variation on the earlier Common Gateway Interface (CGI); FastCGI's main aim is to reduce the overhead associated with interfacing the web server and CGI programs, allowing a server to handle more web page requests at once.

Brett Cameron, John Apps
2009-08-17

FastCGI for OpenVMS

Brett Cameron (brett.r.cameron@gmail.com) and John Apps (johndapps@gmail.com)

August 2009

Disclaimer: the information in this document is the sole responsibility of the authors. The information contained herein is offered on a best-effort basis. Hewlett-Packard offers no support or warranty on any of the content in this document or the software described in it.

Contents

1. Introduction

Thank you for your interest in this port of FastCGI (http://www.fastcgi.com). This port is being made available on OpenVMS Alpha and Integrity (Itanium) Servers in the form of a software development kit from which we hope to gain experience and feedback from those downloading and using it.

To borrow from Wikipedia (http://en.wikipedia.org/wiki/FastCGI):

> *"FastCGI is a protocol for interfacing interactive programs with a web server. FastCGI is a variation on the earlier Common Gateway Interface (CGI); FastCGI's main aim is to reduce the overhead associated with interfacing the web server and CGI programs, allowing a server to handle more web page requests at once.*
>
> *Instead of creating a new process for every request, FastCGI can use a single persistent process which handles many requests over its lifetime. Processing of multiple requests simultaneously is achieved either by using a single connection with internal multiplexing and/or by using multiple connections. Many such processes can exist, something that can increase stability and scalability. FastCGI also allows programs to get the web server to do certain simple operations, like reading in a file, before the request is handed over. Environment information and page requests are sent from the web server to the process over a TCP/IP connection (for remote processes) or UNIX domain sockets (for local processes)[1]. Responses are returned from the process to the web server over the same connection. The connection may be closed at the end of a response, but the web server and the process are left standing.*

[1] It should be noted that only TCP/IP connections are supported for the OpenVMS port. Other forms of inter-process communication analogous to UNIX domain sockets may be incorporated into future releases; however TCP/IP provides considerable flexibility with regard to deployment and is arguably more than fast enough for most purposes.

> *Many web site administrators and programmers are finding that the separation of web applications from the web server in FastCGI has many advantages over embedded interpreters (`mod_perl`, `mod_php`, and so on). This separation allows server and application processes to be restarted independently – an important consideration for busy web sites. It also facilitates per-application security policies – important for ISPs and web hosting companies."*

This release of FastCGI for OpenVMS includes the FastCGI API, which is required to develop FastCGI applications and the `mod_fastcgi` module for Apache (CSWS), allowing Apache on OpenVMS to be used to serve FastCGI-based applications. Applications developed using the FastCGI API can also be used with the WASD Web server (http://wasd.vsm.com.au/) via the WASD FCGIplus extension.

The following sections briefly describe the installation and configuration of the FastCGI for OpenVMS software. For additional information on FastCGI and for details of how to develop FastCGI applications using the FastCGI API, please refer to http://www.fastcgi.com.

2. Requirements

The kit you are installing has been compiled and built using the operating system and compiler versions listed below. While it is highly likely that you will have no problems installing and using the kit on systems running higher product versions, this may not be the case if your system is running older versions. We shall be pleased to hear from you should you have a requirement for older versions and will do our best to accommodate these wishes.

- OpenVMS 8.3 Integrity Server
- HP TCP/IP Services for OpenVMS Industry Standard 64 Version V5.6
- HP C S7.1-013
- UnZip 5.42 (or similar) for OpenVMS is required to unpack the kit

In addition to the above requirements, it is assumed that the reader has a good knowledge of OpenVMS and of software development in the OpenVMS environment.

2.1. Optional Products

While the FastCGI API is primarily intended to be used to develop applications in C, there is no particular reason why it cannot be used to develop applications using practically any OpenVMS 3GL such as COBOL, BASIC, Pascal, or FORTRAN. However, it should be noted that regardless of your language preferences, it will probably be necessary to write at least some C code.

Both WASD (see http://wasd.vsm.com.au/) and the HP Secure Web Server for OpenVMS (Apache for OpenVMS) can be used on OpenVMS as FastCGI servers. Instructions on using WASD as FastCGI server may be found at http://wasd.vsm.com.au/wasd/, and some additional notes on configuring WASD with FastCGI are provided in this document.

In order to use FastCGI with Apache, it is necessary to install the Apache FastCGI module `mod_fastcgi` as described in Section 7. The FastCGI module for Apache 2.1-1 on OpenVMS is provided as part of the FastCGI for OpenVMS kit (note that the `mod_fastcgi` module supplied with this kit will **not** work with Apache 1.3).

It is of course possible to run the FastCGI server for your FastCGI applications on a separate server or indeed a completely different operating system platform to your applications. A list of Web servers supporting FastCGI may be found at http://www.fastcgi.com/drupal/node/3.

3. Recommended Reading

As noted previously, before getting too carried away, be sure to read (or at least browse) the assorted documentation available via the FastCGI Web site (http://www.fastcgi.com). If you are intending to use FastCGI with WASD, then you should also read through the WASD FCGIplus documentation (http://wasd.vsm.com.au/ht_root/src/fcgi/READMORE.HTML).

4. Contents of the Kit

The kit is provided as a ZIP file (created using Zip 2.3 for OpenVMS), which allows individual users to easily install the software under their own accounts, as opposed to having to negotiate with the system administrator for some sort of system-wide installation.

In addition, it should be noted that the kit includes only compiled code (object libraries and shareable images), include files, and example code. While FastCGI is an Open Source technology, several OpenVMS-specific pieces of code have been added by the authors, requiring internal clarification of what licensing or IP issues may or may not exist. Until such matters are resolved, we will provide a binary distribution only.

The following file listing represents the output from the `unzip -l` command run against the ZIP file kit on OpenVMS I64 (the listing on OpenVMS Alpha would be very similar). As you can see, there is not that much to the distribution; however what is there is pretty much all that is required in order to develop FastCGI applications on OpenVMS using C or any other OpenVMS programming language. It is likely that additional functionality and additional examples will be added to the kit over time.

```
$ unzip -l FCGI-VMS-I64-01.ZIP
Archive:  SYS$SYSDEVICE:[apps]FCGI-VMS-I64-01.ZIP;1
  Length      Date    Time    Name
 --------    ----    ----    ----
        0  07-11-09 08:37   fcgi/bin/
        0  07-11-09 08:37   fcgi/examples/
        0  07-11-09 08:37   fcgi/include/
        0  07-11-09 08:37   fcgi/lib/
   154624  07-20-09 02:07   fcgi/bin/mod_fastcgi.exe
      190  08-09-09 23:09   fcgi/examples/build.com
     1955  07-11-09 08:36   fcgi/examples/echo.c
    18588  07-10-09 06:05   fcgi/include/fcgiapp.h
     5848  07-10-09 06:05   fcgi/include/fcgi_stdio.h
   241664  07-26-09 08:54   fcgi/lib/libfcgi.olb
 --------                   -------
   422869                   10 files
```

5. Installation

Unpacking and installing the ZIP file kit is very straightforward. After copying the supplied ZIP file (`FCGI-VMS-AXP-01.ZIP` for OpenVMS Alpha or `FCGI-VMS-I64-01.ZIP` for OpenVMS I64) to a suitable location, unpack the contents of the relevant ZIP file using the `unzip` command.

For example, for the I64 kit, you would enter the following command:

```
$ unzip FCGI-VMS-I64-01.ZIP
Archive:  SYS$SYSDEVICE:[apps]FCGI-VMS-I64-01.ZIP;1
   creating: [.fcgi.bin]
   creating: [.fcgi.examples]
   creating: [.fcgi.include]
   creating: [.fcgi.lib]
      ...
      ...
```

After unpacking the kit you will have a `[.fcgi]` directory in your current directory that contains files listed above from the `unzip -l` command.

5.1. Post-installation Steps

To complete the installation (regardless of the installation method), you should own define the logical `fcgi$root` to point to your top-level `[.fcgi]` directory. If you are performing a system-wide installation then this logical should be defined using `/system`, otherwise a process-level logical should be defined. For example, something similar to one of the following commands might be used to define the `fcgi$root` logical for a system-wide installation:

```
$ define/sys/trans=conc fcgi$root DORONE$DKA0:[fcgi.]
```

or:

```
$ define/sys/exe/tran=(conc,term) fcgi$root -
"''f$getdvi("DISK$IVMS82","DEVNAM")'[sys0.syscommon.fcgi.]"
```

Remember that this is a concealed device logical name and that all the rules relevant to those names apply here.

Installation and configuration of the `mod_fastcgi` Apache module included with the kit is discussed in Section 7.

5.2. Privileges and Quotas

A moderate level of privilege is currently required in order to run applications developed using the current release of the OpenVMS FastCGI port. The FastCGI API code performs a number of calls to the `setsockopt()` TCP/IP system call to adjust various TCP/IP socket attributes. In order to perform these socket operations, the account in question requires a system `UIC` or `SYSPRV`, `BYPASS`, or `OPER` privilege. Subsequent releases of the FastCGI software may provide facilities negate this requirement.

Generally speaking, there are no special quota requirements for applications developed using FastCGI on OpenVMS, although a reasonably high `BYTLM` is recommended. The authors typically operate with quota settings similar to the following (on I64 or Alpha), which should be more than adequate for most purposes:

```
Maxjobs:          0  Fillm:      4096  Bytlm:      2000000
Maxacctjobs:      0  Shrfillm:       0  Pbytlm:           0
Maxdetach:        0  BIOlm:        900  JTquota:       8192
Prclm:            0  DIOlm:        900  WSdef:         4096
Prio:             4  ASTlm:        900  WSquo:        16384
Queprio:          0  TQElm:        900  WSextent:     32767
CPU:         (none)  Enqlm:       8192  Pgflquo:    2000000
```

6. Sample applications

The directory `fcgi$root:[examples]` presently contains a single example program that serves to illustrate the general structure of a FastCGI application and how to build it. To build the sample application, simply copy the files from `fcgi$root:[examples]` into a local directory, and execute the build procedure (`build.com`).

If you look at the code for the sample application (`echo.c`) you will see that one of the first things the program does is to call the function `FCGI_InitVMS()`, passing it any command line arguments that were supplied to the program. This function is specific to the OpenVMS port of the FastCGI API and must be called before calling any other functions in order to ensure that file descriptors used by the FastCGI environment are set up correctly. A detailed description of this function and the various command line arguments that it accepts is provided in Section 9.

After building the sample application (`echo.exe`) and configuring Apache or WASD as described below, the sample application may be run as follows (assuming that the program was built in the `fcgi$root:[examples]` directory:

```
$ echo :== $fcgi$root:[examples]echo.exe
$ echo –n 5 –p 8485
```

The second command starts the FastCGI `echo` server, instructing it to create and maintain 5 child processes, and to listen for requests on port 8485. Refer to Section 9 for more details regarding the various command line options that can be specified.

7. Configuring and using mod_fastcgi with Apache

`mod_fastcgi` is a module for the Apache Web server that implements the FastCGI protocol, allowing Apache to be used with FastCGI applications. Installation of `mod_fastcgi` is straightforward and equates to little more than copying the module into the Apache modules directory and defining the module in the Apache configuration file. Specifically, the following steps should be performed:

- Assuming that FastCGI for OpenVMS has been installed in accordance with the instructions specified in Section 5, copy `fcgi$root:[bin]mod_fastcgi.exe` to `apache$root:[modules]` and ensure that the ownership and permissions on the copied file are the same as the other modules in the Apache directory
- Modify the Apache configuration file (`apache$root:[conf]httpd.conf`) to include the `mod_fastcgi` module. For example, you might add the following statement which instructs Apache to dynamically load the `mod_fastcgi` module at the bottom of the configuration file:

  ```
  LoadModule fastcgi_module modules/mod_fastcgi.exe
  ```

- Restart Apache to pick up the new service to verify that the module loads correctly

As discussed in the following section, only the `FastCgiExternalServer` directive is currently able to be used with `mod_fastcgi` to configure FastCGI applications with Apache on OpenVMS. Configuring an application with the `FastCgiExternalServer` directive is straightforward, and involves adding a single line to `httpd.conf` (after the `LoadModule` directive) to associate the application with a particular URL; do not forget to rest Apache after you have made the change.

For example, the following entry would associate the alias /calc with a FastCGI application running on the same node as the Web server and listening on port 8485:

```
FastCGIExternalServer "/apache$root/000000/htdocs/calc" -host 127.0.0.1:8485
```

Assuming that Apache is running on node bugs.acme.com, users would access the FastCGI application via the URL http://bugs.acme.com/calc.

Note: your FastCGI application should ideally be started before Apache is started. If it is not, Apache will report an error the first time each Apache instance tries to send a request to the application. However, after this initial failure, Apache processes will establish a connection to the FastCGI application and subsequent requests will succeed.

7.1. Known limitations

Only the `FastCgiExternalServer` directive is currently able to be used with this port of `mod_fastcgi` on OpenVMS. The authors feel this is not too much of a limitation. It is hoped that additional directives will be made available in future releases.

8. Configuring WASD

Applications developed using the FastCGI API may also be used in conjunction with the WASD Web server via the FCGIplus extension. Installation instructions for the FCGIplus extension can be found at http://wasd.vsm.com.au/WASD/fcgiplus101.txt and instructions for configuring applications to work with WASD and FCGI plus may be found at http://wasd.vsm.com.au/ht_root/src/fcgi/READMORE.HTML.

Generally speaking, configuration typically equates to little more than adding a single line to `httpd$map.conf` to specify a mapping for your FastCGI application. For example, the following entry would map /calc to a FastCGI application running on localhost (127.0.0.1) listening on port 8485.

```
script+ /calc* /cgi-bin/fcgiplus* \
            script=nofind script=params=fcgi_connect=127.0.0.1:8485
```

Note that any new `script+` entries should be placed with other such (pre-existing) entries in `httpd$map.conf`, and *not* at the bottom of the file.

9. Notes on Using the FastCGI API on OpenVMS

As noted in Section 6, in order for FastCGI applications on OpenVMS to operate correctly it is vitally important that all applications call the function `FCGI_InitVMS()` **before** doing anything else in order to ensure that the OpenVMS FastCGI environment is correctly initialised. This function ensures that file descriptors are correctly set up for communication with FastCGI servers (such as Apache and WASD), starts one or more child processes to process FastCGI requests, and monitors child processes, restarting them as necessary within the prescribed limits.

The C prototype for this function is as follows:

```
int FCGI_InitVMS(int argc, char **argv)
```

The function is intended to be passed any command line options that were supplied to the program. The following options are specific to the operation of the `FCGI_InitVMS()` function:

Option	Description
`-p <port>`	Specifies the port number to be used by the FastCGI application. A port number must be specified as no default port number is assumed.
`-b <backlog>`	Specifies the number of requests that can be queued up on the wire. If this option is not specified on the command line, a default value of 100 will be used. Specified values must be between 1 and 1024 (inclusive).
`-n <number-of-children>`	Specifies that number of child processes that are to be started. If this option is not specified, a default value of 1 will be used. The number of children must be between 1 and 100 (inclusive).
`-m <scan-interval>`	Specifies the interval at which the parent process will check the status of started child processes. If this option is not specified, a scan interval of 30 seconds will be used. The scan interval must be between 10 and 300 seconds (inclusive).
`-r <maximum-restarts>`	Specifies the maximum number of times that the parent process will restart failed child processes. The default value is 10, and the minimum and maximum permissible values are 0 and 150, respectively.
`-v (verbose)`	If this option is specified, the parent process will perform (slightly) more comprehensive operational logging.
`--`	This option instructs `FCGI_InitVMS()` to stop processing command line options at this point. Any command line options specified after the `--` will be passed to child processes and may be used for application-specific purposes.

Table 1 FCGI_InitVMS Options

If the number of arguments is less than or equal to 1 or `FCGI_InitVMS()` determines that the process is a child, the function returns immediately without performing any initialisation or

tasks. If the process is not a child process and the number of arguments is less than or equal to 1, the assumption is that the program is being run as a plain CGI application.

Upon successful completion (for example if the number of arguments is less than or equal to 1 or the process is determined to be a child), the function `FCGI_InitVMS()` returns a value of 0. However, if an error is encountered while processing command line arguments, starting child processes, or while monitoring child processes, the `FCGI_InitVMS()` function will log details of the error and exit. Consequently, there is little value in application programs checking the return status of the function.