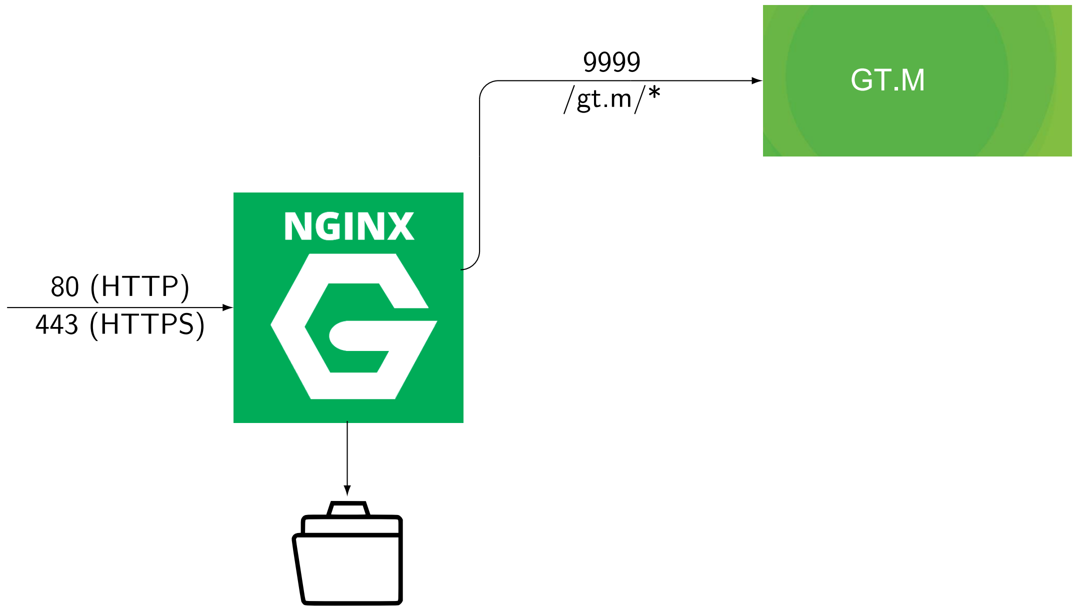
FastCGI for GT.M - Installation and Quick-Start

Winfried Bantel

Aalen University

9. Janur 2019

The nginx-system



Advantages NOT to write a HTTP-Server

- Very very fast HTTP-backend written in native GT.M
- nginx is able to cache - less work for GT.M
- HTTPS supported by nginx
- HTTP/2 supported by nginx
- HTTP/2 with dynamic server push for even faster applications
- Filebased Webserver is done by nginx
- With JSON-Parser ideal backend for Single-Page-Applications (i.e. with AngularJS)
- Supports massive parallel HTTP-requests
- Sensible data can be stored physically on another machine

Test via apache benchmark

Hochschule Aalen

```
>>> ab -n 1000 -c 10 -q "localhost/gt.m/dollarh"
...
Concurrency Level:        10
Time taken for tests:     4.568 seconds
Complete requests:        1000
Failed requests:          0
Total transferred:        178920 bytes
HTML transferred:         13000 bytes
Requests per second:      218.90 [#/sec] (mean)
Time per request:         45.683 [ms] (mean)
Transfer rate:            38.25 [Kbytes/sec] received
...
Percentage of the requests served within a certain time (ms)
  50%      40
  66%      40
  75%      40
  80%      40
  90%      40
  95%      40
  98%      42
  99%     558
 100%     620 (longest request)
```

Installation-Steps

I hope You are firm in GT.M!

1. Install nginx
2. Edit nginx-Config
3. Install fis-gtm
4. Install xinetd
5. Edit GT.M-start-script
6. Edit xinetd-Config-Script
7. Copy FCGI.m
8. Set a global
9. Be happy

User and so on

- In these slides the user is wbantel.
- His home-directory is /home/wbantel/
- If You want another user: adapt!

Step 1: Install nginx

```
>>> sudo apt install nginx
>>> curl localhost
```

Or test from any Computer in WWW / LAN with IP-Address oder DNS

Step 2: Edit nginx-Config

Edit `/etc/nginx/sites-enabled/default`:

- In the global section:

```
upstream gtm_fcgi_backend {
        server 127.0.0.1:9999;
        keepalive 32;
}
```

- In the server-section:

```
         location /gt.m/ {
                fastcgi_pass gtm_fcgi_backend;
                fastcgi_keep_conn on ;
                fastcgi_param   QUERY_STRING        $query_string;
                fastcgi_param   SID                 $cookie_sid;
                fastcgi_param   DOCUMENT_URI        $document_uri;
                fastcgi_param   REQUEST_METHOD      $request_method;
                fastcgi_param   REMOTE_ADDR         $remote_addr;
        }
```

Restart nginx:

```
>>> sudo service nginx restart
```

Step 3: Install fis-gtm

```
>>> sudo apt install fis-gtm
>>>
```

Step 4: Install xinetd

```
>>> sudo apt install xinetd
>>>
```

Step 5: Edit GT.M-start-script

```
>>> cat /home/wbantel/mumps.sh
#!/bin/bash
export gtmdir=/home/wbantel/.fis-gtm
/usr/lib/x86_64-linux-gnu/fis-gtm/V6.2-002A-2build1_x86_64/utf8/gtm $1 $2
>>> chmod +x mumps.sh
>>> ./mumps.sh -run TEST
```

- TEST.m shold be in Your routine-directory `/home/wbantel/.fis-gtm/V.../r/`
- Perhaps the gtm-script is in another directory!?!

  ```
  >>> find / -name "gtm"
  ```

  and grap the one You want!

Step 6: Edit xinetd-Config-Script

```
>>>  cat /etc/xinetd.d/gtm-fastcgi
service gtm-fastcgi
{
        protocol        = tcp
        port            = 9999
        type            = UNLISTED
        socket_type     = stream
        wait            = no
        user            = wbantel
        group           = wbantel
        server          = /home/wbantel/mumps.sh
        server_args     = -run FCGI
        disable         = no
}
>>> sudo service xinetd restart
>>>
```

Step 7: Copy FCGI.m

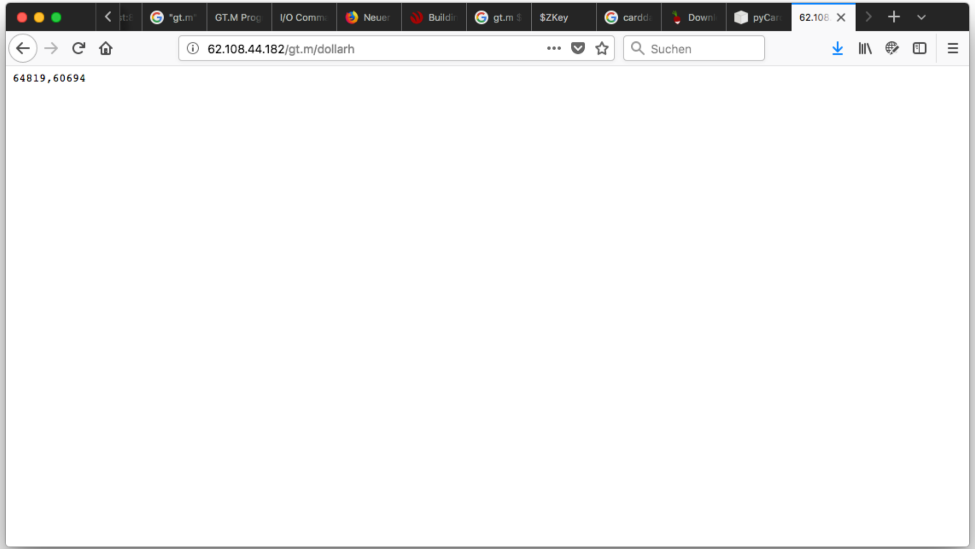
```
>>> cp /from/somewhere/FCGI.m /home/wbantel/.fis-gtm/V.../r/
>>>
```

Step 8: Set a global

```
>>>  /home/wbantel/mumps.sh
GTM> SET ^FCGI("DOCUMENT_URI","/gt.m/dollarh")="DOLLARH"
GTM>
>>>
```

Step 9: Be happy

Hochschule Aalen



```
64819,60694
```

More than one System

You need another GTM-System, perhaps for development an production, totally different?

- Create another user with a GT.M-Start-Script
- Create another xinetd-Config with another TCP/IP-Port and another name
- Create another Upstream-Part with the correct Port in nginx-Config
- Create another Location-Part with another URI an the correct Upstream in nginx-Config

Some configuration

```
^FCGI("PRM","ZLINK")
```

0 (or killed or not 1)	The called routine will be called without ZLINK
1	The called routine will be ZLINKed before called

Use this parameter for developing (set to 1) so when you edit a routine and save it the changes will have an effect. Otherwise kill the global and it will run a little bit faster...

How to call an M-routine

- FastCGI examines `$PIECE(uri,"/",1,3)`
  Attention, first piece is alway empty
- Second piece has to be the location from nginx-config-file (usually gt.m)
- Third piece is variable and used for distribute to application-routine
- Set an Indirection-Global for Your app (see step 8)

How to generate Output

Several ways for backend-routine to generate output

1. Write to device %fcgi
2. Give a global-name
3. Give a filename
4. Set a single variable
5. Set an array variable

Don't mix it up, use only exactly one way!

Generate output - device %fcgi

- Easiest way to generate Output

```
EXOUTPUT1     ; Generates output using %fcgi
          ; On start %5cgi is open and used!!!
          w "<html><head></head><body>",$H,"</body></html>"
```

Generate output - globalbased

- Ideal in case of the global already exists

```
EXOUTPUT2    ; Generates output using global
        s ^dummy="<html><head></head><body>"_$H_"</body></html>"
        s %fcgi("o","global")="^dummy"
```

Generate output - filebased

Hochschule Aalen

- Ideal in case of the file already exists

```
EXOUTPUT3    ; Generates output using file / filename
        s f="/tmp/"_$j_".html"
        u f w "<html><head></head><body>"_$H_"</body></html>" c f
        s %fcgi("o","file")=f
```

Generate output using local variable

```
EXOUTPUT4     ; using local variable
        s %fcgi("o","stdout")"<html><head></head><body>"_$H_"</body><
```

Generate output using local array

```
EXOUTPUT5     ; Generate output using array
        s %fcgi("o","stdout",1)="<html>"
        s %fcgi("o","stdout",2)="<head></head>"
        s %fcgi("o","stdout",3)="<body>"_$H_"</body>"
        s %fcgi("o","stdout",4)="</html>"
```

Output-Header

- For Content-Type, Redirect and so on

```
EXSETHEADER    ; Generates output using %fcgi
       s %fcgi("o","header","Content-Type")="application/json"
       w "{""$H"":"""_$H_""",""$J"":"""_$J_"""}"
```

```
>>> curl -i "localhost:8080/gt.m/EXSETHEADER"
HTTP/1.1 200 OK
Server: nginx/1.14.0
Date: Wed, 09 Jan 2019 14:07:03 GMT
Content-Type: application/json
Content-Length: 32
Connection: keep-alive
X-job: 2483
X-nr: 1

{"$H":"65022,54423","$J":"2483"}
```

Session-ID is the new $JOB

- Session-tracking ist forced calling `SID^FCGI`
- Stored in `%fcgi("i","header","SID")`
- Two Comma-separated integers:
  1. 64-bit random-int which ist constant for your session
  2. Counter auto-incrementing with each HTTP-request
- Is done by a temporary (non-perstiant) cookie
- Ideal for storing session-specific data

```
EXSID    ; Generates output using %fcgi
         q:'$$SID^FCGI()  s sid=%fcgi("i","header","SID")
         w "<html><head></head><body>"
         w "Your Session-ID is ",+sid,"<br>",!
         w "Your Session-count is ",$P(sid,",",2),"<br>",!
         w "Your last visit ($H) was: ",$G(^dummy(+sid)),"<br>",!
         s h=$H w "Now $H is: ",h,"<br>",!
         s ^dummy(+sid)=h
         w "<br>Feel free to reload!"
         w "<br><a href=""javascript:location.reload()"">Reload</a>"
         w "</body></html>"
```

HTTP-GET-vars

- Easiest way to get data from Webclient

```
EXGETVAR    ;
        w "<html><head></head><body>"
        i $G(%fcgi("i","_GET","name"))="" d
        . w "You did't enter a name"
        e  w "Hello ",%fcgi("i","_GET","name"),"!"
        w "<form method=""POST"">",!
        w "<input type=""text"" name=""name"">",!
        w "<input type=""submit"" value=""Submit"">",!
        w "</form></body></html>"
```

HTTP-POST-vars

```
EXPOSTVAR       ;
        w "<html><head></head><body>"
        i $G(%fcgi("i","_GET","name"))="" d
        . w "You did't enter a name"
        e  w "Hello ",%fcgi("i","_POST","name"),"!"
        w "<form method=""POST"">",!
        w "<input type=""text"" name=""name"">",!
        w "<input type=""submit"" value=""Submit"">",!
        w "</form></body></html>"
```

HTTP-BODY

- Suitable for JSON-data, File-Uploads and so on

```
EXSTDIN     ;
          ; > curl ip-address:port/gt.m/EXPOSTVAR -d "Hallo Welt!"
          ; > curl ip-address:port/gt.m/EXPOSTVAR -d @file.txt
          ; Or a Browser-form with method post:
          ; <form action="/gt.m/EXPOSTVAR" method="POST">...</form>
         w "<html><head></head><body>Your Post-Data is<pre>"
         w $G(%fcgi("i","stdin"))
         w "</pre></body></html>",!
```

```
>>> curl -i "localhost:8080/gt.m/EXSTDIN" -d '{"NN":"Bantel"}'
HTTP/1.1 200 OK
Server: nginx/1.14.0
Date: Wed, 09 Jan 2019 14:13:28 GMT
Content-Length: 83
Connection: keep-alive
X-job: 2699
X-nr: 2

<html><head></head><body>Your Post-Data is<pre>{"NN":"Bantel"}</pre></body
```

Complete HTTP-Info

- The complete info is stored in %fcgi

```
EXHTTPINFO    ;;
        s %fcgi("o","header","Content-Type")="text/plain"
        zwr %fcgi
```

```
>>>  curl "localhost:8080/gt.m/EXHTTPINFO?test=1"  -d '{"NN":"Bantel"}'
%fcgi="/tmp/fcgi-fifo-4011" ;*
%fcgi("i","FCGI_KEEP_CONN")=1
%fcgi("i","_GET","test")=1
%fcgi("i","_POST","{""NN"":""Bantel""}")=""
%fcgi("i","header","DOCUMENT_URI")="/gt.m/EXHTTPINFO"
%fcgi("i","header","HTTP_ACCEPT")="*/*"
%fcgi("i","header","HTTP_CONTENT_LENGTH")=15
%fcgi("i","header","HTTP_CONTENT_TYPE")="application/x-www-form-urlencoded
%fcgi("i","header","HTTP_HOST")="localhost:8080"
%fcgi("i","header","HTTP_USER_AGENT")="curl/7.51.0"
%fcgi("i","header","QUERY_STRING")="test=1"
%fcgi("i","header","REMOTE_ADDR")="10.0.2.2"
%fcgi("i","header","REQUEST_METHOD")="POST"
%fcgi("i","header","SID")=""
%fcgi("i","stdin")="{""NN"":""Bantel""}"
%fcgi("internal","entryRef")="^EXHTTPINFO"
```