yaoweibin / nginx_ajp_module

support AJP protocol proxy with Nginx http://github.com/yaoweibin/nginx_ajp_module

131 commits | 4 branches | 9 releases | 5 contributors

branch: master nginx_ajp_module / +

Merge pull request #28 from itpp16/patch-5 ...

yaoweibin authored on Apr 29, 2014 latest commit 12c852ce7b

test	Merge branch 'master' into development	a year ago
util	fix docs and tidy the source	4 years ago
README	added the tag v0.3	a year ago
README.markdown	added the tag v0.3	a year ago
README.wiki	added the tag v0.3	a year ago
config	Update config	10 months ago
ngx_http_ajp.c	Update ngx_http_ajp.c	10 months ago
ngx_http_ajp.h	fixed coding style	a year ago
ngx_http_ajp_handler.c	Update ngx_http_ajp_handler.c	10 months ago
ngx_http_ajp_handler.h	add state machine for input filter	4 years ago
ngx_http_ajp_module.c	fixed coding style	a year ago
ngx_http_ajp_module.h	fixed coding style	2 years ago
ngx_http_ajp_msg.c	Update ngx_http_ajp_msg.c	10 months ago

README.markdown

Name

nginx_ajp_module - support AJP protocol proxy with Nginx

Synopsis

```
http {
        upstream tomcats {
                server 127.0.0.1:8009;
                keepalive 10;
        }

        server {

                listen 80;

                location / {
                        ajp_keep_conn on;
                        ajp_pass tomcats;
                }
        }
}
```

Description

With this module, Nginx can connect to AJP port directly. The motivation of writing these modules is

Nginx's high performance and robustness.

Directives

ajp_buffers

syntax: *ajp_buffers the_number is_size;*

default: *ajp_buffers 8 4k/8k;*

context: *http, server, location*

This directive specifies the number and the size of buffers, into which will be read the response, obtained from the AJP server. By default, the size of one buffer is equal to the size of a page. Depending on platform this is either 4K, 8K or 16K.

ajp_buffer_size

syntax: *ajp_buffer_size the_size;*

default: *ajp_buffer_size 4k/8k;*

context: *http, server, location*

This directive sets the buffer size, into which will be read the first part of the response, obtained from the AJP server.

In this part of response the small response-header is located, as a rule.

By default, the buffersize is equal to the size of one buffer in directive `ajp_buffers`; however, it is

possible to set it to less.

ajp_cache

syntax: *ajp_cache zone;*

default: *off*

context: *http, server, location*

The directive specifies the area which actually is the share memory's name for caching. The same area can be used in several places. You must set the `ajp_cache_path` first.

ajp_cache_key

syntax: *ajp_cache_key line;*

default: *none*

context: *http, server, location*

The directive specifies what information is included in the key for caching, for example

```
    ajp_cache_key "$host$request_uri$cookie_user";
```

Note that by default, the hostname of the server is not included in the cache key. If you are using subdomains for different locations on your website, you need to include it, e.g. by changing the cache key to something like

```
    ajp_cache_key "$scheme$host$request_uri";
```

ajp_cache_methods

syntax: *ajp_cache_methods [GET HEAD POST];*

default: *ajp_cache_methods GET HEAD;*

context: *main,http,location*

GET/HEAD is syntax sugar, i.e. you can not disable GET/HEAD even if you set just

```
    ajp_cache_methods  POST;
```

ajp_cache_min_uses

syntax: *ajp_cache_min_uses n;*

default: *ajp_cache_min_uses 1;*

context: *http, server, location*

Sets the number of requests after which the response will be cached.

ajp_cache_path

syntax: *ajp_cache_path /path/to/cache [levels=m:n keys_zone=name:time inactive=time clean_time=time];*

default: *none*

context: *http, server, location*

This directive sets the cache path and other cache parameters. Cached data stored in files. Key and filename in cache is md5 of proxied URL. **Levels** parameter set number of subdirectories in cache, for example for:

```
    ajp_cache_path  /data/nginx/cache  levels=1:2   keys_zone=one:10m;
```

file names will be like:

/data/nginx/cache/c/29/b7f54b2df7773722d382f4809d65029c

ajp_cache_use_stale

syntax: *ajp_cache_use_stale [updating|error|timeout|invalid_header|http_500];*

default: *ajp_cache_use_stale off;*

context: *http, server, location*

If an error occurs while working with the AJP server it is possible to use a stale cached response. This directives determines in which cases it is permitted. The directive's parameters match those of the `ajp_next_upstream` directive.

Additionally, the updating parameter permits to use a stale cached response if it is currently being updated. This allows to minimize the number of accesses to AJP servers when updating cached data.

ajp_cache_valid

syntax: *ajp_cache_valid [http_error_code|time];*

default: *none*

context: *http, server, location*

Sets caching time for different response codes. For example, the following directives

```
ajp_cache_valid 200 302 10m;
ajp_cache_valid 404      1m;
```

set 10 minutes of caching for responses with codes 200 and 302, and 1 minute for responses with code 404.

If only caching time is specified

```
ajp_cache_valid 5m;
```

then only 200, 301, and 302 responses are cached.

In addition, it can be specified to cache any responses using the any parameter:

```
ajp_cache_valid 200 302 10m;
ajp_cache_valid 301      1h;
ajp_cache_valid any      1m;
```

Parameters of caching can also be set directly in the response header. This has a higher precedence than setting of caching time using the directive. The “X-Accel-Expires” header field sets caching time of a response in seconds. The value 0 disables to cache a response. If a value starts with the prefix @, it sets an absolute time in seconds since Epoch, up to which the response may be cached. If header does not include the “X-Accel-Expires” field, parameters of caching may be set in the header fields “Expires” or “Cache-Control”. If a header includes the “Set-Cookie” field, such a response will not be cached. Processing of one or more of these response header fields can be disabled using the `ajp_ignore_headers` directive.

ajp_connect_timeout

syntax: *ajp_connect_timeout time;*

default: *ajp_connect_timeout 60s;*

context: *http, server, location*

This directive assigns a timeout for the connection to the upstream server. It is necessary to keep in mind that this time out cannot be more than 75 seconds.

This is not the time until the server returns the pages, this is the ajp_read_timeout statement. If your upstream server is up, but hanging (e.g. it does not have enough threads to process your request so it puts you in the pool of connections to deal with later), then this statement will not help as the connection to the server has been made.

ajp_header_packet_buffer_size

syntax: *ajp_header packet_buffer_size;*

default: *ajp_header_packet_buffer_size 8k;*

context: *http, server, location*

Set the buffer size of Forward Request packet. The range is (0, 2^16).

ajp_hide_header

syntax: *ajp_hide_header name;*

context: *http, server, location*

By default, Nginx does not pass headers "Status" and "X-Accel-..." from the AJP process back to the client. This directive can be used to hide other headers as well.

If the headers "Status" and "X-Accel-..." must be provided, then it is necessary to use directive ajp_pass_header to force them to be returned to the client.

ajp_ignore_headers

syntax: *ajp_ignore_headers name [name ...];*

default: *none*

context: *http, server, location*

This directive(0.7.54+) prohibits the processing of the header lines from the proxy server's response.

It can specify the string as "X-Accel-Redirect", "X-Accel-Expires", "Expires" or "Cache-Control".

ajp_ignore_client_abort

syntax: *ajp_ignore_client_abort on|off;*

default: *ajp_ignore_client_abort off;*

context: *http, server, location*

This directive determines if current request to the AJP-server must be aborted in case the client aborts the request to the server.

ajp_intercept_errors

syntax: *ajp_intercept_errors on|off;*

default: *ajp_intercept_errors off;*

context: *http, server, location*

This directive determines whether or not to transfer 4xx and 5xx errors back to the client or to allow Nginx to answer with directive error_page.

Note: You need to explicitly define the error_page handler for this for it to be useful. As Igor says, "nginx does not intercept an error if there is no custom handler for it it does not show its default pages. This allows to intercept some errors, while passing others as are."

ajp_keep_conn

syntax: *ajp_keep_conn on|off;*

default: *ajp_keep_conn off;*

context: *http, server, location*

This directive determines whether or not to keep the connection alive with backend server.

ajp_next_upstream

syntax: *ajp_next_upstream [error|timeout|invalid_header|http_500|http_502|http_503|http_504|http_404|off];*

default: *ajp_next_upstream error timeout;*

context: *http, server, location*

Directive determines, in what cases the request will be transmitted to the next server:

- error — an error has occurred while connecting to the server, sending a request to it, or reading its response;
- timeout — occurred timeout during the connection with the server, transfer the request or while reading response from the server;
- invalid_header — server returned a empty or incorrect answer;
- http_500 — server returned answer with code 500;
- http_502 — server returned answer with code 502;
- http_503 — server returned answer with code 503;
- http_504 — server returned answer with code 504;
- http_404 — server returned answer with code 404;
- off — it forbids the request transfer to the next server Transferring the request to the next server is only possible when nothing has been transferred to the client -- that is, if an error or timeout arises in the middle of the transfer of the request, then it is not possible to retry the current request on a different server.

ajp_max_data_packet_size

syntax: *ajp_max_data_packet_size size;*

default: *ajp_max_data_packet_size 8k;*

context: *http, server, location*

Set the maximum size of AJP's Data packet. The range is [8k, 2^16];

ajp_max_temp_file_size

syntax: *ajp_max_temp_file_size size;*

default: *ajp_max_temp_file_size 1G;*

context: *http, server, location, if*

The maximum size of a temporary file when the content is larger than the proxy buffer. If file is larger than this size, it will be served synchronously from upstream server rather than buffered to disk.

If ajp_max_temp_file_size is equal to zero, temporary files usage will be disabled.

ajp_pass

syntax: *ajp_pass ajp-server*

default: *none*

context: *location, if in location*

Directive assigns the port or socket on which the AJP-server is listening. Port can be indicated by itself or as an address and port, for example:

```
    ajp_pass   localhost:9000;
```

using a Unix domain socket:

```
    ajp_pass   unix:/tmp/ajp.socket;
```

You may also use an upstream block.

```
    upstream backend  {
            server   localhost:1234;
    }
```

```
    ajp_pass   backend;
```

ajp_pass_header

syntax: *ajp_pass_header name;*

context: *http, server, location*

Permits to pass specific header fields from the AJP server to a client.

ajp_pass_request_headers

syntax: *ajp_pass_request_headers [on | off];*

default: *ajp_pass_request_headers on;*

context: *http, server, location*

Permits to pass request header fields from the client to server.

ajp_pass_request_body

syntax: *ajp_pass_request_body [on | off] ;*

default: *ajp_pass_request_body on;*

context: *http, server, location*

Permits to pass request body from the client to server.

ajp_read_timeout

syntax: *ajp_read_timeout time;*

default: *ajp_read_timeout_time 60*

context: *http, server, location*

Directive sets the amount of time for upstream to wait for a AJP process to send data. Change this directive if you have long running AJP processes that do not produce output until they have finished processing. If you are seeing an upstream timed out error in the error log, then increase this parameter to something more appropriate.

ajp_send_lowat

syntax: *ajp_send_lowat [on | off];*

default: *ajp_send_lowat off;*

context: *http, server, location, if*

This directive set SO_SNDLOWAT. This directive is only available on FreeBSD

ajp_send_timeout

syntax: *ajp_send_timeout time;*

default: *ajp_send_timeout 60;*

context: *http, server, location*

This directive assigns timeout with the transfer of request to the upstream server. Timeout is

established not on entire transfer of request, but only between two write operations. If after this time the upstream server will not take new data, then nginx is shutdown the connection.

ajp_store

syntax: *ajp_store [on | off | path] ;*

default: *ajp_store off;*

context: *http, server, location*

This directive sets the path in which upstream files are stored. The parameter "on" preserves files in accordance with path specified in directives *alias* or *root*. The parameter "off" forbids storing. Furthermore, the name of the path can be clearly assigned with the aid of the line with the variables:

```
    ajp_store   /data/www$original_uri;
```

The time of modification for the file will be set to the date of "Last-Modified" header in the response. To be able to safe files in this directory it is necessary that the path is under the directory with temporary files, given by directive `ajp_temp_path` for the data location.

This directive can be used for creating the local copies for dynamic output of the backend which is not very often changed, for example:

```
    location /images/ {
            root                /data/www;
            error_page          404 = @fetch;
    }

    location @fetch {
            internal;
```

```
            ajp_pass            backend;
            ajp_store           on;
            ajp_store_access    user:rw  group:rw  all:r;
            ajp_temp_path       /data/temp;

            root                /data/www;
    }
```

To be clear ajp_store is not a cache, it's rather mirror on demand.

ajp_store_access

syntax: *ajp_store_access users:permissions [users:permission ...];*

default: *ajp_store_access user:rw;*

context: *http, server, location*

This directive assigns the permissions for the created files and directories, for example:

```
    ajp_store_access  user:rw  group:rw  all:r;
```

If any rights for groups or all are assigned, then it is not necessary to assign rights for user:

```
    ajp_store_access  group:rw  all:r;
```

ajp_temp_path

syntax: *ajp_temp_path dir-path [level1 [level2 [level3]]] ;*

default: *$NGX_PREFIX/ajp_temp*

context: *http, server, location*

This directive works like client_body_temp_path to specify a location to buffer large proxied requests to the filesystem.

ajp_temp_file_write_size

syntax: *ajp_temp_file_write_size size;*

default: *ajp_temp_file_write_size ["#ajp buffer size"] * 2;*

context: *http, server, location, if*

Sets the amount of data that will be flushed to the ajp_temp_path when writing. It may be used to prevent a worker process blocking for too long while spooling data.

Installation

Download the latest version of the release tarball of this module from github (http://github.com/yaoweibin/nginx_ajp_module)

Grab the nginx source code from nginx.org (http://nginx.org/), for example, the version 1.2.0 (see nginx compatibility), and then build the source with this module:

```
$ wget 'http://nginx.org/download/nginx-1.4.4.tar.gz'
$ tar -xzvf nginx-1.4.4.tar.gz
$ cd nginx-1.4.4/
$ ./configure --add-module=/path/to/nginx_ajp_module

$ make
```

```
$ make install
```

Compatibility

- The master branch is for Nginx-1.1.4+
- If you want to use it with Nginx-1.0.x, you can use this nginx-1.0 (https://github.com/yaoweibin/nginx_ajp_module/tree/nginx-1.0) branch.

TODO

- SSL

Known Issues

*

Changelogs

v0.3

- remove the jvm_route and keepalive module

v0.2

- bugfix

v0.1

- first release

Authors

- Weibin Yao(姚伟斌) *yaoweibin AT gmail DOT com*
- Jinti Shen(路奇) *jinti.shen AT gmail DOT com*
- Joshua Zhu(叔度) *zhuzhaoyuan AT gmail DOT com*
- Simon Liu(雕梁) *simohayha.bobo AT gmail DOT com*
- Matthew Ma(东坡) *mj19821214 AT gmail DOT com*

Acknowledgments

- Thanks 李金虎(beagem@163.com) to improve the keepalive feature with this module.

License

This README template is from agentzh (http://github.com/agentzh).

I borrowed a lot of codes from Fastcgi module of Nginx. This part of code is copyrighted by Igor Sysoev. And the design of apache's mod_ajp_proxy (http://httpd.apache.org/docs/trunk/mod/mod_proxy_ajp.html). Thanks for their hard work.

This module is licensed under the BSD license.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

POD ERRORS

Hey! **The above document had some coding errors, which are explained below:**

- Around line 212:

  L<> starts or ends with whitespace