日日是Oracle APEX

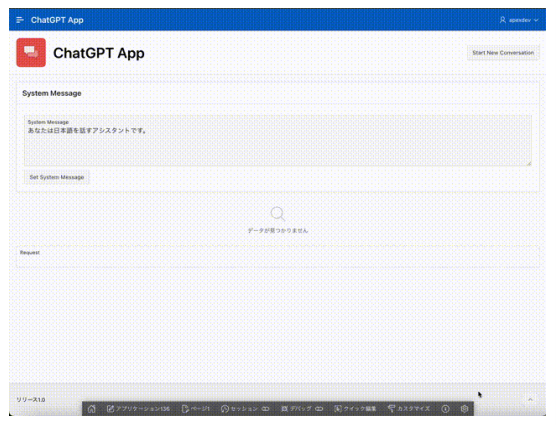
Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2023年7月21日金曜日

llama_cpp.serverをAmpere A1のインスタンス上で動かしてみる

先の記事でLlama.cppを使ってAmpere A1でLlama 2を動かしてみたのですが、Llama.cppにはllama_cpp.serverというOpenAI互換のサーバーが含まれていました。

OpenAIについては、以前にAPIを呼び出す簡単なAPEXアプリケーションを作っているので（こちらの記事）、llama_cpp.serverをAmpere A1のインスタンスで実行して、APEXアプリケーションからアクセスしてみました。



APIのエンドポイントのホストをapi.openai.comからllama_cpp.serverを実行しているサーバーに変えるだけで、APEXアプリケーションはそのまま動作しました。

ただし、扱うトークン数の最大値が64かその近辺になっているようで、回答が短すぎました。ドキュメントに記載は見つけられませんが、llama_cpp.serverのリクエストに属性としてmax_tokensを含めることで、長い回答を得ることができました。

APEXアプリケーションのエクスポートは以下です。
https://github.com/ujnak/apexapps/blob/master/exports/chatgpt-app.zip

このアプリケーションに以下の変更を加えます。

アプリケーション定義の**置換**に**置換文字列**として**G_SERVER**を定義します。**置換値**はllama_cpp.serverが稼働しているインスタンスの（**https://**で始まる）**ホスト名**になります。

ボタンSEND_USER_MESSAGEをクリックしたときに実行されるPL/SQLコードを変更します。変更は３箇所です。

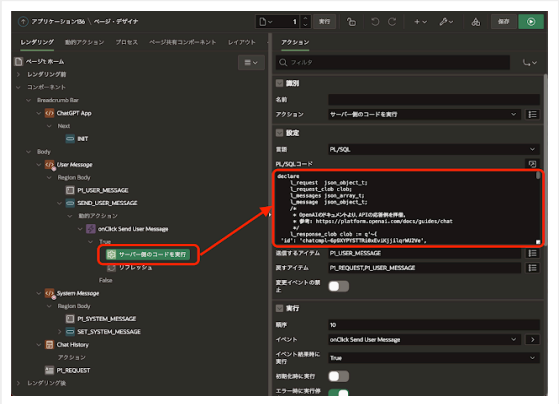
1. 属性max_tokensの追加。
2. API呼び出しのURLに直書きされたapi.openai.comを、置換文字列G_SERVERに置き換える。
3. Web資格証明の指定を除く。

```
declare
    l_request  json_object_t;
    l_request_clob clob;
    l_messages json_array_t;
    l_message  json_object_t;
    /*
     * OpenAIのドキュメントより、APIの応答例を拝借。
     * 参考: https://platform.openai.com/docs/guides/chat
     */
    l_response_clob clob := q'~{
 'id': 'chatcmpl-6p9XYPYSTTRi0xEviKjjilqrWU2Ve',
 'object': 'chat.completion',
 'created': 1677649420,
 'model': 'gpt-3.5-turbo',
 'usage': {'prompt_tokens': 56, 'completion_tokens': 31, 'total_tokens': 87},
 'choices': [
   {
    'message': {
      'role': 'assistant',
      'content': 'The 2020 World Series was played in Arlington, Texas at the Globe Life Field,
    'finish_reason': 'stop',
    'index': 0
   }
  ]
}~';
    l_response json_object_t;
    l_choices json_array_t;
    l_role    varchar2(80);
    l_content clob;
    l_usage json_object_t;
begin
    /*
     * ユーザーによるメッセージをコレクションに追記する。
```

```
 */
apex_collection.add_member(
    p_collection_name => 'CHATGPT'
    ,p_c001 => 'user'
    ,p_clob001 => :P1_USER_MESSAGE
);
/*
 * ChatGPTのAPIに送信するメッセージを作成する。
 */
/*
 * APEXコレクションCHATGPTより、作成時刻の昇順でメッセージの配列にする。
 */
l_messages := json_array_t();
for r in (select c001, clob001 from apex_collections where collection_name = 'CHATGPT' orde
loop
    l_message := json_object_t();
    l_message.put('role'   ,r.c001);
    l_message.put('content',r.clob001);
    l_messages.append(l_message);
end loop;
l_request := json_object_t();
l_request.put('model','gpt-3.5-turbo');
/* llama_cpp.server向けにmax_tokensを追加 */
l_request.put('max_tokens', 256);
l_request.put('messages', l_messages);
/*
 * temprature, top_pなどのパラメータを設定するとしたら、ここでputする。
 */
l_request_clob := l_request.to_clob();
/* デバッグのため、送信するメッセージをP1_REQUESTに書き込む。 */
:P1_REQUEST := l_request_clob;
/*
 * OpenAIのChatGPTのAPIを呼び出す。今の所、コメントアウト。
 *
 * 参照: https://openai.com/blog/introducing-chatgpt-and-whisper-apis
 * API Ref: https://platform.openai.com/docs/api-reference/chat
*/
apex_web_service.clear_request_headers;
apex_web_service.set_request_headers('Content-Type','application/json',p_reset => false);
l_response_clob := apex_web_service.make_rest_request(
    /* 直書きされたapi.openai.comの部分を置換文字列に変更。 */
    p_url => :G_SERVER || '/v1/chat/completions'
    ,p_http_method => 'POST'
    ,p_body => l_request_clob
    /* Web資格証明の指定は不要 */
    -- ,p_credential_static_id => 'OPENAI_API_KEY'
);
```

```
    /* ドキュメントに記載されているレスポンスで処理を継続する。 */
    l_response := json_object_t(l_response_clob);
    l_choices := l_response.get_array('choices');
    l_message := treat(l_choices.get(0) as json_object_t).get_object('message');
    l_role    := l_message.get_string('role');
    l_content := l_message.get_clob('content');
    /* usageも取り出す。 */
    l_usage := l_response.get_object('usage');
    /*
     * ChatGPTからの応答をAPEXコレクションに追記する。
     */
    apex_collection.add_member(
        p_collection_name => 'CHATGPT'
        ,p_c001 => l_role
        ,p_clob001 => l_content
        ,p_n001 => l_usage.get_number('prompt_tokens')
        ,p_n002 => l_usage.get_number('completion_tokens')
        ,p_n003 => l_usage.get_number('total_tokens')
    );
    /* ユーザーのメッセージを消去する */
    :P1_USER_MESSAGE := '';
end;
```

llama_cpp_server_call.sql hosted with ❤ by GitHub view raw



これより、Llama_cpp.serverを動かすために実施した作業を記載します。

Oracle Cloudで動かすHTTPサーバーは基本的にHTTPS化する必要があります。Llama_cpp.serverはHTTPSで動かす方法は提供されていないので、Nginxによりリクエストを受け付けるようにします。

最初にNginxをインストールします。

sudo apt install nginx

（以下はNginxがインストール済みなので、これといった作業は行われていません。）

```
ubuntu@mywhisper2:~$ sudo apt install nginx
Reading package lists... Done
```

```
Building dependency tree
Reading state information... Done
nginx is already the newest version (1.18.0-0ubuntu1.4).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@mywhisper2:~$
```

Llama_cpp.serverを動かすために必要なパッケージをインストールします。

pip install llama-cpp-python[server]

（以下もパッケージはインストール済みなので、これといった作業は行われていません。）

```
ubuntu@mywhisper2:~$ pip install llama-cpp-python[server]
Requirement already satisfied: llama-cpp-python[server] in
./.local/lib/python3.8/site-packages (0.1.73)
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.8/dist-
packages (from llama-cpp-python[server]) (1.23.5)
Requirement already satisfied: diskcache>=5.6.1 in ./.local/lib/python3.8/site-
packages (from llama-cpp-python[server]) (5.6.1)
Requirement already satisfied: typing-extensions>=4.5.0 in
./.local/lib/python3.8/site-packages (from llama-cpp-python[server]) (4.7.1)
Requirement already satisfied: uvicorn>=0.22.0; extra == "server" in
./.local/lib/python3.8/site-packages (from llama-cpp-python[server]) (0.23.1)
Requirement already satisfied: fastapi>=0.100.0; extra == "server" in
./.local/lib/python3.8/site-packages (from llama-cpp-python[server]) (0.100.0)
Requirement already satisfied: pydantic-settings>=2.0.1; extra == "server" in
./.local/lib/python3.8/site-packages (from llama-cpp-python[server]) (2.0.2)
Requirement already satisfied: sse-starlette>=1.6.1; extra == "server" in
./.local/lib/python3.8/site-packages (from llama-cpp-python[server]) (1.6.1)
Requirement already satisfied: h11>=0.8 in ./.local/lib/python3.8/site-packages
(from uvicorn>=0.22.0; extra == "server"->llama-cpp-python[server]) (0.14.0)
Requirement already satisfied: click>=7.0 in ./.local/lib/python3.8/site-packages
(from uvicorn>=0.22.0; extra == "server"->llama-cpp-python[server]) (8.1.3)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4
in ./.local/lib/python3.8/site-packages (from fastapi>=0.100.0; extra == "server"-
>llama-cpp-python[server]) (2.0.3)
Requirement already satisfied: starlette<0.28.0,>=0.27.0 in
./.local/lib/python3.8/site-packages (from fastapi>=0.100.0; extra == "server"-
>llama-cpp-python[server]) (0.27.0)
Requirement already satisfied: python-dotenv>=0.21.0 in
./.local/lib/python3.8/site-packages (from pydantic-settings>=2.0.1; extra ==
"server"->llama-cpp-python[server]) (1.0.0)
Requirement already satisfied: annotated-types>=0.4.0 in
./.local/lib/python3.8/site-packages (from pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,
<3.0.0,>=1.7.4->fastapi>=0.100.0; extra == "server"->llama-cpp-python[server])
(0.5.0)
Requirement already satisfied: pydantic-core==2.3.0 in ./.local/lib/python3.8/site-
packages (from pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4-
>fastapi>=0.100.0; extra == "server"->llama-cpp-python[server]) (2.3.0)
Requirement already satisfied: anyio<5,>=3.4.0 in /usr/local/lib/python3.8/dist-
packages (from starlette<0.28.0,>=0.27.0->fastapi>=0.100.0; extra == "server"-
>llama-cpp-python[server]) (3.6.2)
Requirement already satisfied: idna>=2.8 in /usr/lib/python3/dist-packages (from
anyio<5,>=3.4.0->starlette<0.28.0,>=0.27.0->fastapi>=0.100.0; extra == "server"-
>llama-cpp-python[server]) (2.8)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.8/dist-
packages (from anyio<5,>=3.4.0->starlette<0.28.0,>=0.27.0->fastapi>=0.100.0; extra
== "server"->llama-cpp-python[server]) (1.3.0)
ubuntu@mywhisper2:~$
```

Nginxは、HTTPSの標準ポートである443で接続を待ち受けるように構成します。そのため、firewalldで443から8443へのポート・フォワードが設定されている場合、その設定を削除します。

firewall-cmd --remove-forward-port=port=443:proto=tcp:toport=8443

Llama_cpp.serverはポート8000（デフォルト）で接続を待ち受けるため、firewalldにポート8000の接続許可を与えます。

firewall-cmd --add-port=8000/tcp

変更を永続化します。

firewall-cmd --runtime-to-permanent

最終的に以下の設定になります。

firewall-cmd --list-all

```
root@mywhisper2:/home/ubuntu# firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client http https ssh
  ports: 8000/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

root@mywhisper2:/home/ubuntu#
```

Nginxの構成ファイルを/etc/nginx/conf.d/server.confとして作成します。内容は以下になります。

```
server {
    listen 443 ssl;
    ssl_certificate     /etc/letsencrypt/live/ホスト名/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/ホスト名/privkey.pem;
    server_name ホスト名;
    root /usr/share/nginx/html;
    index index.html;

    location / {
        proxy_pass http://localhost:8000/;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        proxy_redirect off;
    }
}
```

server.conf hosted with ❤️ by GitHub　　view raw

Let's encryptのcertbotを使って証明書を取得済みとします。Nginxはrootで動作させるため、**/etc/letsencrypt/live/ホスト名**以下に作成される**fullchain.pem**、**privkey.pem**を直接参照しています。

以上でNginxの構成は完了です。Nginxを起動します。

systemctl start nginx

```
root@mywhisper2:~# systemctl start nginx
root@mywhisper2:~# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:
enabled)
     Active: active (running) since Fri 2023-07-21 03:03:36 UTC; 18s ago
       Docs: man:nginx(8)
    Process: 139267 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process
on; (c>
    Process: 139268 ExecStart=/usr/sbin/nginx -g daemon on; master_process on;
(code=exite>
   Main PID: 139269 (nginx)
      Tasks: 5 (limit: 28696)
     Memory: 4.4M
     CGroup: /system.slice/nginx.service
             ├─139269 nginx: master process /usr/sbin/nginx -g daemon on;
master_process o>
             ├─139270 nginx: worker process
             ├─139271 nginx: worker process
             ├─139272 nginx: worker process
             └─139273 nginx: worker process

Jul 21 03:03:36 mywhisper2 systemd[1]: Starting A high performance web server and a
revers>
Jul 21 03:03:36 mywhisper2 systemd[1]: Started A high performance web server and a
reverse>
lines 1-18/18 (END)
```

あとはllama_cpp.serverを起動するだけなのですが、llama_cpp.serverを起動すると以下の警告が発生します。

```
warning: failed to mlock 174080000-byte buffer (after previously locking 0 bytes):
Cannot allocate memory
Try increasing RLIMIT_MLOCK ('ulimit -l' as root).
llama_new_context_with_model: kv self size  = 1600.00 MB
```

無償枠のAmpere A1、4OCPU、24GBメモリのインスタンスを使っているため、メモリは足りていると思っていたのですが、mlockとmmapのシステム・コールが失敗しています。これは、ulimitでmax locked memoryが65536に制限されているためでした。

```
ubuntu@mywhisper2:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 95653
max locked memory       (kbytes, -l) 65536
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
```

```
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 95653
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
ubuntu@mywhisper2:~$
```

一般ユーザーのubuntuにてmax locked memoryの制限を外すため、**/etc/security/limits.conf**に以下の記述を追加します。

ubuntu hard memlock unlimited

```
#<domain>      <type>  <item>         <value>
#

#*               soft    core            0
#root            hard    core            100000
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#ftp             -       chroot          /ftp
#@student        -       maxlogins       4
ubuntu           hard    memlock         unlimited
```

ユーザーubuntuでログインし直した後から、max locked memoryを変更できるようになります。

ulimit -l unlimited

その後、llama_cpp.serverを起動します。このインスタンスは4OCPUなので、n_threadsに4を指定します。(デフォルトは2)

python3 -m llama_cpp.server --model llama-2-13b-chat.ggmlv3.q8_0.bin --n_threads 4

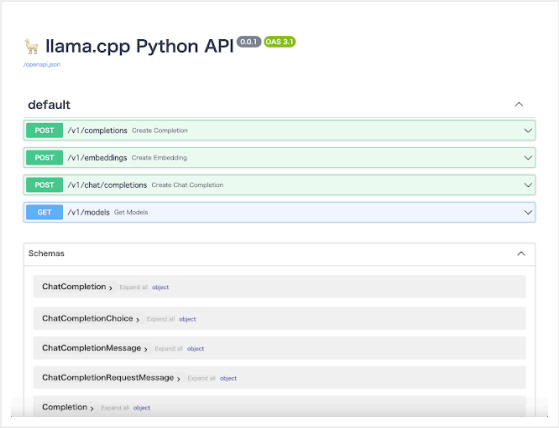
```
ubuntu@mywhisper2:~$ ulimit -l
65536
ubuntu@mywhisper2:~$ ulimit -l unlimited
ubuntu@mywhisper2:~$ python3 -m llama_cpp.server --model llama-2-13b-
chat.ggmlv3.q8_0.bin --n_threads 4
/home/ubuntu/.local/lib/python3.8/site-packages/pydantic/_internal/_fields.py:126:
UserWarning: Field "model_alias" has conflict with protected namespace "model_".

You may be able to resolve this warning by setting
`model_config['protected_namespaces'] = ('settings_',)`.
  warnings.warn(
llama.cpp: loading model from llama-2-13b-chat.ggmlv3.q8_0.bin
llama_model_load_internal: format     = ggjt v3 (latest)
llama_model_load_internal: n_vocab    = 32000
llama_model_load_internal: n_ctx      = 2048
llama_model_load_internal: n_embd     = 5120
llama_model_load_internal: n_mult     = 256
llama_model_load_internal: n_head     = 40
llama_model_load_internal: n_layer    = 40
llama_model_load_internal: n_rot      = 128
llama_model_load_internal: freq_base  = 10000.0
llama_model_load_internal: freq_scale = 1
llama_model_load_internal: ftype      = 7 (mostly Q8_0)
llama_model_load_internal: n_ff       = 13824
```

```
llama_model_load_internal: model size = 13B
llama_model_load_internal: ggml ctx size =    0.09 MB
llama_model_load_internal: mem required  = 15159.96 MB (+ 1608.00 MB per state)
llama_new_context_with_model: kv self size  = 1600.00 MB
AVX = 0 | AVX2 = 0 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | FMA = 0 |
NEON = 1 | ARM_FMA = 1 | F16C = 0 | FP16_VA = 1 | WASM_SIMD = 0 | BLAS = 0 | SSE3 =
0 | VSX = 0 |
INFO:     Started server process [139695]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://localhost:8000 (Press CTRL+C to quit)
```

以上でllama_cpp.serverにアクセスできるようになります。

ブラウザより以下のURLにアクセスします。

https://ホスト名/docs



まだまだ、チューニングの余地はあると思いますが、作業の紹介は以上になります。

完

Yuji N. 時刻: 12:29

共有