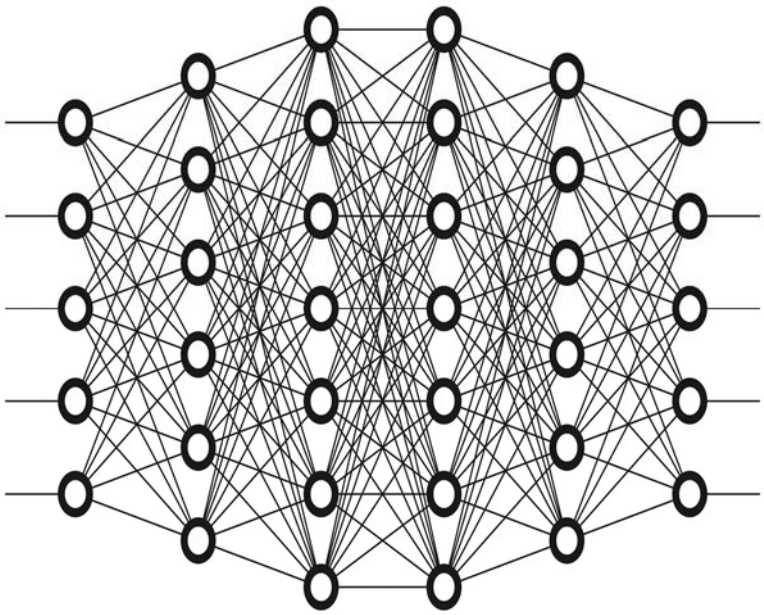




Déploiement d'une machine learning LLM chatbot dans un environnement de cloud computing



Déploiement d'une machine learning LLM chatbot dans un environnement de cloud computing

il s'agit d'un réseau neuronal transformer-based. Ces modèles de base utilisent l'IA générative (et plus précisément l'apprentissage profond) pour le traitement du langage naturel et la génération de langage naturel.

les LLMs parcourent de vastes ensembles de données textuelles afin de s'entrainer à vous répondre.

On utilisera dans le cadre de ce développement un modèle LLM llama 2 de la société méta et un modèle de la société mistral AI. Pour limiter les ressources consommées on adoptera un modèle orienté chatbot. Mistral 7B et Llama2 apparaissent comme deux des modèles de langage local (LLM) open source les plus performants et gratuits.

les modèles sont spécifiquement pré-entraînés pour les tâches orientées chat, ce qui en fait un choix naturel pour la création d'applications d'IA conversationnelle

Pour Meta on a choisi le modèle llama-2-7b-chat.Q4_K_M.gguf

GGUF est un nouveau format introduit par le workframe llama.cpp. Il remplace GGML, qui n'est plus pris en charge par llama.cpp. GGUF offre de nombreux avantages par rapport à GGML, tels qu'une meilleure tokenisation (manière de fournirdu texte au LLM) et la prise en charge de tokens spéciaux. Il prend également en charge les métadonnées et est conçu pour être extensible.

Pour Mistral AI on a choisi le modèle mistral-7b-v0.1.Q4_K_M.gguf. Ces deux modèles sont équivalents en terme de paramètres (7 millions pour chacun).

La première partie permet d'élaborer une image d'un conteneur grâce à un fichier DockerFile (une connexion internet est nécessaire pour télécharger les outils nécessaires).

Ce fichier Dockerfile mettra en place le framework llama.cpp sous python dans un environnement d'exécution sous CPU.

L'image de base sera une distribution débian disposant de nombreux outils de développement pour python. Elle sera nommé python et se trouve sur le dockerhub

il chargera le framework flask (serveur web développé sous python) afin de proposer une inférence vers le chatbot LLM grâce au port 5000

```
EXPOSE 5000
```

Le fichier llama_cpu_server.py permettra grâce à Flask le déploiement du LLM conteneurisé

```
CMD ["python", "llama_cpu_server.py"]
```

Les deux fichiers le modèle llama-2-7b-chat.Q4_K_M.gguf et le programme de déploiement seront copiés dans le conteneur à l'emplacement suivant dans le système de fichier de l'image de base /app

Le répertoire de travail du pod sera /app

Proposez le contenu du fichier Dockerfile

Créer une image à partir du dockerfile

```
docker build -t llama-cpu-server .
```

Dans un premier temps sous notre nœud master on peut mettre en place un conteneur sous docker

```
docker run -p 5000:5000 llama-cpu-server
```

L'inférence vers le modèle LLM s'effectue par l'intermédiaire de requête Post (API Restfull) en JSON. Tapez la commande suivante grâce à curl sous le master

```
curl -X POST -H "Content-Type: application/json" -d '{ "system_message": "You are a helpful assistant", "user_message": "Generate a list of 5 funny dog names", "max_tokens": 100 }' http://172.42.42.100:5000/llama
```

Attendre ainsi la réponse du LLM (un Tocken correpond approximativement à 4 caratères)

Enregistrez notre image sous un fichier compressé .tar

docker save llama-cpu-server:latest -o /tmp/ /tmp/ llama-cpu-server.tar

On souhaite sous Openshift déployer ce modèle

Lancez sous Vcenter ou ESXI8 la VM ocgitlab

Copiez dans le répertoire /tmp le fichier /tmp/ llama-cpu-server.tar

chargez notre image dans le cache de podman

```
podman load -i /tmp/ llama-cpu-server.tar
```

taggez notre image pour le registre privée quay

```
podman tag llama-cpu-server:latest ocbastion.poolst.gouv.fr:8443 /mirror/oc-mirror-metadata/ llama-cpu-server:latest
```

Puis poussez notre image dans le registre privé quay

```
podman push ocbastion.poolst.gouv.fr:8443 /mirror/oc-mirror-metadata/ llama-cpu-server:latest
```

La deuxième partie consiste à déployer un pod sous openshift qui nous permet de mettre en place ce serveur LLM

Lancez sous Vcenter ou ESXI8 la VM snocluster1 (openshift)

On doit créer le fichier manifest pour la mise en place de ce pod

notre fichier manifest yaml doit répondre aux exigences suivantes :

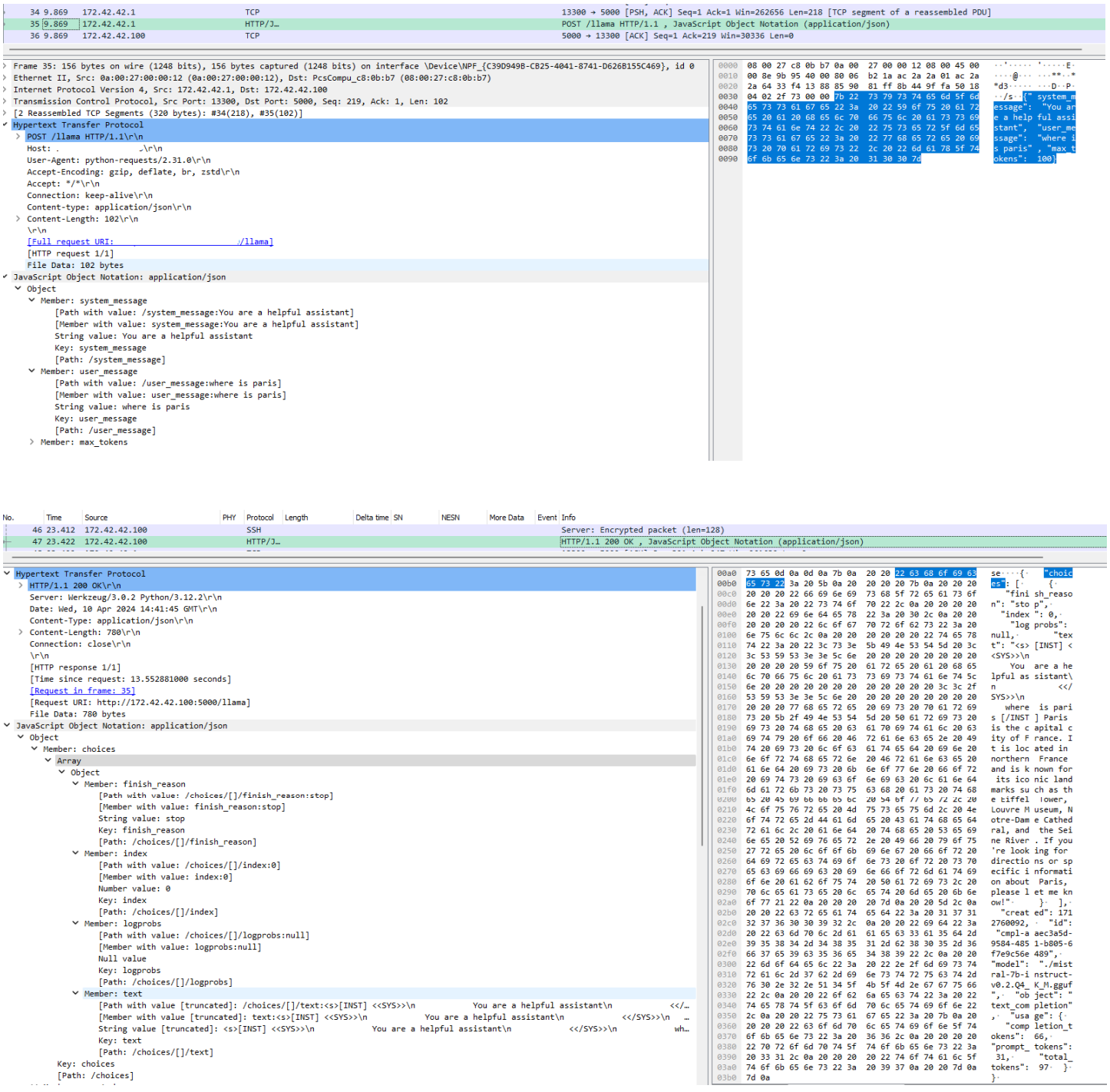
On souhaite un replica de 1 pour les tests sur le master
L'image se trouve sur le registre privée ocbastion.poolst.gouv.fr:8443 /mirror/oc-mirror-metadata/ llama-cpu-server:latest
Il n'y aura aucune donnée persistante
Le serveur flask LLM dispose d'un service sur le port 5000 pour l'inférence Ce service utilise les requêtes https post restfull. Mettez en place un service Nodeport de façon à joindre ce service respectivement sur le port 30050

Fournir le fichier yaml et Déployez le soit en ligne de commande sur la VM ocgitlab ou directement sous openshift :

La troisième partie nous permet de tester ce service grâce à un chabot développé maison LLMJson.exe:

Lancez ce programme et relevez sous wireshark les échanges requête réponse :




34 9.869 172.42.42.1 TCP 13300 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=262656 Len=218 [TCP segment of a reassembled PDU]
35 9.869 172.42.42.1 HTTP/J… POST /llama HTTP/1.1 , JavaScript Object Notation (application/json)
36 9.869 172.42.42.100 TCP 5000 → 13300 [ACK] Seq=1 Ack=219 Win=30336 Len=0
Frame 35: 156 bytes on wire (1248 bits), 156 bytes captured (1248 bits) on interface \Device\NPF_{C39D949B-CB25-4041-8741-D626B155C469}, id 0
Ethernet II, Src: 0a:00:27:00:00:12 (0a:00:27:00:00:12), Dst: PcsCompu_c8:0b:b7 (08:00:27:c8:0b:b7)
Internet Protocol Version 4, Src: 172.42.42.1, Dst: 172.42.42.100
Transmission Control Protocol, Src Port: 13300, Dst Port: 5000, Seq: 219, Ack: 1, Len: 102
[2 Reassembled TCP Segments (320 bytes): #34(218), #35(102)]
Hypertext Transfer Protocol
POST /llama HTTP/1.1\r\n
Host:
User-Agent: python-requests/2.31.0\r\n
Accept-Encoding: gzip, deflate, br, zstd\r\n
Accept: */*\r\n
Connection: keep-alive\r\n
Content-type: application/json\r\n
Content-Length: 102\r\n
\r\n
[Full request URI: /llama]
[HTTP request 1/1]
File Data: 102 bytes
JavaScript Object Notation: application/json
Object
Member: system_message
[Path with value: /system_message:You are a helpful assistant]
[Member with value: system_message:You are a helpful assistant]
String value: You are a helpful assistant
Key: system_message
[Path: /system_message]
Member: user_message
[Path with value: /user_message:where is paris]
[Member with value: user_message:where is paris]
String value: where is paris
Key: user_message
[Path: /user_message]
Member: max_tokens
0000 08 00 27 c8 0b b7 0a 00 27 00 00 12 08 00 45 00
0010 00 8e 9b 95 40 00 80 06 b2 1a ac 2a 2a 01 ac 2a
0020 2a 64 33 f4 13 88 85 90 81 ff 8b 44 9f fa 50 18
0030 04 02 2f 73 00 00 7b 22 73 79 73 74 65 6d 5f 6d
0040 65 73 73 61 67 65 22 3a 20 22 59 6f 75 20 61 72
0050 65 20 61 20 68 65 6c 70 66 75 6c 20 61 73 73 69
0060 73 74 61 6e 74 22 2c 20 22 75 73 65 72 5f 6d 65
0070 73 73 61 67 65 22 3a 20 22 77 68 65 72 65 20 69
0080 73 20 70 61 72 69 73 22 2c 20 22 6d 61 78 5f 74
0090 6f 6b 65 6e 73 22 3a 20 31 30 30 7d

No. Time Source PHY Protocol Length Delta time SN NESN More Data Event Info
46 23.412 172.42.42.100 SSH Server: Encrypted packet (len=128)
47 23.422 172.42.42.100 HTTP/J… HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Server: Werkzeug/3.0.2 Python/3.12.2\r\n
Date: Wed, 10 Apr 2024 14:41:45 GMT\r\n
Content-Type: application/json\r\n
Content-Length: 780\r\n
Connection: close\r\n
\r\n
[HTTP response 1/1]
[Time since request: 13.552881000 seconds]
[Request in frame: 35]
[Request URI: http://172.42.42.100:5000/llama]
File Data: 780 bytes
JavaScript Object Notation: application/json
Object
Member: choices
Array
Object
Member: finish_reason
[Path with value: /choices/[]/finish_reason:stop]
[Member with value: finish_reason:stop]
String value: stop
Key: finish_reason
[Path: /choices/[]/finish_reason]
Member: index
[Path with value: /choices/[]/index:0]
[Member with value: index:0]
Number value: 0
Key: index
[Path: /choices/[]/index]
Member: logprobs
[Path with value: /choices/[]/logprobs:null]
[Member with value: logprobs:null]
Null value
Key: logprobs
[Path: /choices/[]/logprobs]
Member: text
[Path with value [truncated]: /choices/[]/text:<s>[INST] <<SYS>>\n You are a helpful assistant\n <</…
[Member with value [truncated]: text:<s>[INST] <<SYS>>\n You are a helpful assistant\n <</SYS>>\n …
String value [truncated]: <s>[INST] <<SYS>>\n You are a helpful assistant\n <</SYS>>\n wh…
Key: text
[Path: /choices/[]/text]
Key: choices
[Path: /choices]
00a0 73 65 0d 0a 0d 0a 7b 0a 20 20 22 63 68 6f 69 63
00b0 65 73 22 3a 20 5b 0a 20 20 20 20 7b 0a 20 20 20
00c0 20 20 20 22 66 69 6e 69 73 68 5f 72 65 61 73 6f
00d0 6e 22 3a 20 22 73 74 6f 70 22 2c 0a 20 20 20 20
00e0 20 20 22 69 6e 64 65 78 22 3a 20 30 2c 0a 20 20
00f0 20 20 20 22 6c 6f 67 70 72 6f 62 73 22 3a 20
0100 6e 75 6c 6c 2c 0a 20 20 20 20 20 22 74 65 78
0110 74 22 3a 20 22 3c 73 3e 5b 49 4e 53 54 5d 20 3c
0120 3c 53 59 53 3e 3e 5c 6e 20 20 20 20 20 20 20
0130 20 20 20 20 59 6f 75 20 61 72 65 20 61 20 68 65
0140 6c 70 66 75 6c 20 61 73 73 69 73 74 61 6e 74 5c
0150 6e 20 20 20 20 20 20 20 20 20 20 20 20 3c 3c 2f
0160 53 59 53 3e 3e 5c 6e 20 20 20 20 20 20 20 20
0170 20 20 20 77 68 65 72 65 20 69 73 20 70 61 72 69
0180 73 20 5b 2f 49 4e 53 54 5d 20 50 61 72 69 73 20
0190 69 73 20 74 68 65 20 63 61 70 69 74 61 6c 20 63
01a0 69 74 79 20 6f 66 20 46 72 61 6e 63 65 2e 20 49
01b0 74 20 69 73 20 6c 6f 63 61 74 65 64 20 69 6e 20
01c0 6e 6f 72 74 68 65 72 6e 20 46 72 61 6e 63 65 20
01d0 61 6e 64 20 69 73 20 6b 6e 6f 77 6e 20 66 6f 72
01e0 20 69 74 73 20 69 63 6f 6e 69 63 20 6c 61 6e 64
01f0 6d 61 72 6b 73 20 73 75 63 68 20 61 73 20 74 68
0200 65 20 45 69 66 66 65 6c 20 54 6f 77 65 72 2c 20
0210 4c 6f 75 76 72 65 20 4d 75 73 65 75 6d 2c 20 4e
0220 6f 74 72 65 2d 44 61 6d 65 20 43 61 74 68 65 64
0230 72 61 6c 2c 20 61 6e 64 20 74 68 65 20 53 65 69
0240 6e 65 20 52 69 76 65 72 2e 20 49 66 20 79 6f 75
0250 27 72 65 20 6c 6f 6f 6b 69 6e 67 20 66 6f 72 20
0260 64 69 72 65 63 74 69 6f 6e 73 20 6f 72 20 73 70
0270 65 63 69 66 69 63 20 69 6e 66 6f 72 6d 61 74 69
0280 6f 6e 20 61 62 6f 75 74 20 50 61 72 69 73 2c 20
0290 70 6c 65 61 73 65 20 6c 65 74 20 6d 65 20 6b 6e
02a0 6f 77 21 22 0a 20 20 20 20 7d 0a 20 20 5d 2c 0a
02b0 20 20 22 63 72 65 61 74 65 64 22 3a 20 31 37 31
02c0 32 37 36 30 30 39 32 2c 0a 20 20 22 69 64 22 3a
02d0 20 22 63 6d 70 6c 2d 61 61 65 63 33 61 35 64 2d
02e0 39 35 38 34 2d 34 38 35 31 2d 62 38 30 35 2d 36
02f0 66 37 65 39 63 35 36 65 34 38 39 22 2c 0a 20 20
0300 22 6d 6f 64 65 6c 22 3a 20 22 2e 2f 6d 69 73 74
0310 72 61 6c 2d 37 62 2d 69 6e 73 74 72 75 63 74 2d
0320 76 30 2e 32 2e 51 34 5f 4b 5f 4d 2e 67 67 75 66
0330 22 2c 0a 20 20 22 6f 62 6a 65 63 74 22 3a 20 22
0340 74 65 78 74 5f 63 6f 6d 70 6c 65 74 69 6f 6e 22
0350 2c 0a 20 20 22 75 73 61 67 65 22 3a 20 7b 0a 20
0360 20 20 20 22 63 6f 6d 70 6c 65 74 69 6f 6e 5f 74
0370 6f 6b 65 6e 73 22 3a 20 36 36 2c 0a 20 20 20 20
0380 22 70 72 6f 6d 70 74 5f 74 6f 6b 65 6e 73 22 3a
0390 20 33 31 2c 0a 20 20 20 20 22 74 6f 74 61 6c 5f
03a0 74 6f 6b 65 6e 73 22 3a 20 39 37 0a 20 20 7d 0a
03b0 7d 0a