LLaMA.cpp HTTP Server（LLaMA.cpp项目中的HTTP Server）

Fast, lightweight, pure C/C++ HTTP server based on httplib, nlohmann::json and **llama.cpp**.

- 基于[httplib]、nlohmann::json、(https://github.com/yhirose/cpp-httplib)、**llama.cpp**快速的、轻量级的、纯C/C++的HTTP server

Set of LLM REST APIs and a simple web front end to interact with llama.cpp.

- 一组 LLM REST API 和一个用于与 llama.cpp 交互的简单 Web 前端。

Features:

- LLM inference of F16 and quantized models on GPU and CPU（GPU 和 CPU 上 F16 和量化模型的 LLM 推理）
- OpenAI API compatible chat completions and embeddings routes（OpenAI API 兼容聊天完成和嵌入路线）
- Parallel decoding with multi-user support（支持多用户并行解码）
- Continuous batching
- Multimodal (wip)
- Monitoring endpoints（监控端点）
- Schema-constrained JSON response format（受架构约束的 JSON 响应格式）

The project is under active development, and we are looking for feedback and contributors.

- 该项目正在积极开发中，我们正在[寻求反馈和贡献者]（https://github.com/ggerganov/llama.cpp/issues/4216）。

Usage（使用：）

```
# 下列内容是如何通过参数控制.llama-server的使用
usage: ./llama-server [options]

general:

  -h,    --help, --usage          print usage and exit
         --version                show version and build info
  -v,    --verbose                print verbose information
         --verbosity N            set specific verbosity level (default: 0)
         --verbose-prompt         print a verbose prompt before generation
(default: false)
         --no-display-prompt      don't print prompt at generation (default:
false)
  -co,   --color                  colorise output to distinguish prompt and user
input from generations (default: false)
  -s,    --seed SEED              RNG seed (default: -1, use random seed for < 0)
  -t,    --threads N              number of threads to use during generation
```

```
(default: 8)
  -tb,   --threads-batch N        number of threads to use during batch and prompt
processing (default: same as --threads)
  -td,   --threads-draft N        number of threads to use during generation
(default: same as --threads)
  -tbd,  --threads-batch-draft N  number of threads to use during batch and prompt
processing (default: same as --threads-draft)
         --draft N                number of tokens to draft for speculative
decoding (default: 5)
  -ps,   --p-split N              speculative decoding split probability (default:
0.1)
  -lcs,  --lookup-cache-static FNAME
                                  path to static lookup cache to use for lookup
decoding (not updated by generation)
  -lcd,  --lookup-cache-dynamic FNAME
                                  path to dynamic lookup cache to use for lookup
decoding (updated by generation)
  -c,    --ctx-size N             size of the prompt context (default: 0, 0 =
loaded from model)
  -n,    --predict N              number of tokens to predict (default: -1, -1 =
infinity, -2 = until context filled)
  -b,    --batch-size N           logical maximum batch size (default: 2048)
  -ub,   --ubatch-size N          physical maximum batch size (default: 512)
         --keep N                 number of tokens to keep from the initial prompt
(default: 0, -1 = all)
         --chunks N               max number of chunks to process (default: -1, -1
= all)
  -fa,   --flash-attn             enable Flash Attention (default: disabled)
  -p,    --prompt PROMPT          prompt to start generation with
                                  in conversation mode, this will be used as
system prompt
                                  (default: '')
  -f,    --file FNAME             a file containing the prompt (default: none)
         --in-file FNAME          an input file (repeat to specify multiple files)
  -bf,   --binary-file FNAME      binary file containing the prompt (default:
none)
  -e,    --escape                 process escapes sequences (\n, \r, \t, \', \",
\\) (default: true)
         --no-escape              do not process escape sequences
  -ptc,  --print-token-count N    print token count every N tokens (default: -1)
         --prompt-cache FNAME     file to cache prompt state for faster startup
(default: none)
         --prompt-cache-all       if specified, saves user input and generations
to cache as well
                                  not supported with --interactive or other
interactive options
         --prompt-cache-ro        if specified, uses the prompt cache but does not
update it
  -r,    --reverse-prompt PROMPT  halt generation at PROMPT, return control in
interactive mode
                                  can be specified more than once for multiple
prompts
  -sp,   --special                special tokens output enabled (default: false)
  -cnv,  --conversation           run in conversation mode, does not print special
```

```
tokens and suffix/prefix
                                  if suffix/prefix are not specified, default chat
template will be used
                                  (default: false)
  -i,    --interactive            run in interactive mode (default: false)
  -if,   --interactive-first      run in interactive mode and wait for input right
away (default: false)
  -mli,  --multiline-input        allows you to write or paste multiple lines
without ending each in '\'
         --in-prefix-bos          prefix BOS to user inputs, preceding the `--in-
prefix` string
         --in-prefix STRING       string to prefix user inputs with (default:
empty)
         --in-suffix STRING       string to suffix after user inputs with
(default: empty)
         --spm-infill             use Suffix/Prefix/Middle pattern for infill
(instead of Prefix/Suffix/Middle) as some models prefer this. (default: disabled)

sampling:

         --samplers SAMPLERS      samplers that will be used for generation in the
order, separated by ';'
                                  (default:
top_k;tfs_z;typical_p;top_p;min_p;temperature)
         --sampling-seq SEQUENCE  simplified sequence for samplers that will be
used (default: kfypmt)
         --ignore-eos             ignore end of stream token and continue
generating (implies --logit-bias EOS-inf)
         --penalize-nl            penalize newline tokens (default: false)
         --temp N                 temperature (default: 0.8)
         --top-k N                top-k sampling (default: 40, 0 = disabled)
         --top-p N                top-p sampling (default: 0.9, 1.0 = disabled)
         --min-p N                min-p sampling (default: 0.1, 0.0 = disabled)
         --tfs N                  tail free sampling, parameter z (default: 1.0,
1.0 = disabled)
         --typical N              locally typical sampling, parameter p (default:
1.0, 1.0 = disabled)
         --repeat-last-n N        last n tokens to consider for penalize (default:
64, 0 = disabled, -1 = ctx_size)
         --repeat-penalty N       penalize repeat sequence of tokens (default:
1.0, 1.0 = disabled)
         --presence-penalty N     repeat alpha presence penalty (default: 0.0, 0.0
= disabled)
         --frequency-penalty N    repeat alpha frequency penalty (default: 0.0,
0.0 = disabled)
         --dynatemp-range N       dynamic temperature range (default: 0.0, 0.0 =
disabled)
         --dynatemp-exp N         dynamic temperature exponent (default: 1.0)
         --mirostat N             use Mirostat sampling.
                                  Top K, Nucleus, Tail Free and Locally Typical
samplers are ignored if used.
                                  (default: 0, 0 = disabled, 1 = Mirostat, 2 =
Mirostat 2.0)
         --mirostat-lr N          Mirostat learning rate, parameter eta (default:
```

```
0.1)
         --mirostat-ent N         Mirostat target entropy, parameter tau (default:
5.0)
         -l TOKEN_ID(+/-)BIAS     modifies the likelihood of token appearing in
the completion,
                                  i.e. `--logit-bias 15043+1` to increase
likelihood of token ' Hello',
                                  or `--logit-bias 15043-1` to decrease likelihood
of token ' Hello'
         --cfg-negative-prompt PROMPT
                                  negative prompt to use for guidance (default:
'')
         --cfg-negative-prompt-file FNAME
                                  negative prompt file to use for guidance
         --cfg-scale N            strength of guidance (default: 1.0, 1.0 =
disable)
         --chat-template JINJA_TEMPLATE
                                  set custom jinja chat template (default:
template taken from model's metadata)
                                  if suffix/prefix are specified, template will be
disabled
                                  only commonly used templates are accepted:

https://github.com/ggerganov/llama.cpp/wiki/Templates-supported-by-
llama_chat_apply_template

grammar:

         --grammar GRAMMAR        BNF-like grammar to constrain generations (see
samples in grammars/ dir) (default: '')
         --grammar-file FNAME     file to read grammar from
  -j,    --json-schema SCHEMA     JSON schema to constrain generations
(https://json-schema.org/), e.g. `{}` for any JSON object
                                  For schemas w/ external $refs, use --grammar +
example/json_schema_to_grammar.py instead

embedding:

         --pooling {none,mean,cls,last}
                                  pooling type for embeddings, use model default
if unspecified
         --attention {causal,non-causal}
                                  attention type for embeddings, use model default
if unspecified

context hacking:

         --rope-scaling {none,linear,yarn}
                                  RoPE frequency scaling method, defaults to
linear unless specified by the model
         --rope-scale N           RoPE context scaling factor, expands context by
a factor of N
         --rope-freq-base N       RoPE base frequency, used by NTK-aware scaling
(default: loaded from model)
```

```
         --rope-freq-scale N      RoPE frequency scaling factor, expands context
by a factor of 1/N
         --yarn-orig-ctx N        YaRN: original context size of model (default: 0
= model training context size)
         --yarn-ext-factor N      YaRN: extrapolation mix factor (default: -1.0,
0.0 = full interpolation)
         --yarn-attn-factor N     YaRN: scale sqrt(t) or attention magnitude
(default: 1.0)
         --yarn-beta-slow N       YaRN: high correction dim or alpha (default:
1.0)
         --yarn-beta-fast N       YaRN: low correction dim or beta (default: 32.0)
  -gan,  --grp-attn-n N           group-attention factor (default: 1)
  -gaw,  --grp-attn-w N           group-attention width (default: 512.0)
  -dkvc, --dump-kv-cache          verbose print of the KV cache
  -nkvo, --no-kv-offload          disable KV offload
  -ctk,  --cache-type-k TYPE      KV cache data type for K (default: f16)
  -ctv,  --cache-type-v TYPE      KV cache data type for V (default: f16)

perplexity:

         --all-logits             return logits for all tokens in the batch
(default: false)
         --hellaswag              compute HellaSwag score over random tasks from
datafile supplied with -f
         --hellaswag-tasks N      number of tasks to use when computing the
HellaSwag score (default: 400)
         --winogrande             compute Winogrande score over random tasks from
datafile supplied with -f
         --winogrande-tasks N     number of tasks to use when computing the
Winogrande score (default: 0)
         --multiple-choice        compute multiple choice score over random tasks
from datafile supplied with -f
         --multiple-choice-tasks N
                                  number of tasks to use when computing the
multiple choice score (default: 0)
         --kl-divergence          computes KL-divergence to logits provided via --
kl-divergence-base
         --ppl-stride N           stride for perplexity calculation (default: 0)
         --ppl-output-type {0,1}  output type for perplexity calculation (default:
0)

parallel:

  -dt,   --defrag-thold N         KV cache defragmentation threshold (default:
-1.0, < 0 - disabled)
  -np,   --parallel N             number of parallel sequences to decode (default:
1)
  -ns,   --sequences N            number of sequences to decode (default: 1)
  -cb,   --cont-batching          enable continuous batching (a.k.a dynamic
batching) (default: enabled)

multi-modality:

         --mmproj FILE            path to a multimodal projector file for LLaVA.
```

```
see examples/llava/README.md
         --image FILE             path to an image file. use with multimodal
models. Specify multiple times for batching

backend:

         --rpc SERVERS            comma separated list of RPC servers
         --mlock                  force system to keep model in RAM rather than
swapping or compressing
         --no-mmap                do not memory-map model (slower load but may
reduce pageouts if not using mlock)
         --numa TYPE              attempt optimizations that help on some NUMA
systems
                                    - distribute: spread execution evenly over all
nodes
                                    - isolate: only spawn threads on CPUs on the
node that execution started on
                                    - numactl: use the CPU map provided by numactl
                                  if run without this previously, it is
recommended to drop the system page cache before using this
                                  see
https://github.com/ggerganov/llama.cpp/issues/1437

model:

         --check-tensors          check model tensor data for invalid values
(default: false)
         --override-kv KEY=TYPE:VALUE
                                  advanced option to override model metadata by
key. may be specified multiple times.
                                  types: int, float, bool, str. example: --
override-kv tokenizer.ggml.add_bos_token=bool:false
         --lora FNAME             apply LoRA adapter (implies --no-mmap)
         --lora-scaled FNAME S    apply LoRA adapter with user defined scaling S
(implies --no-mmap)
         --lora-base FNAME        optional model to use as a base for the layers
modified by the LoRA adapter
         --control-vector FNAME   add a control vector
                                  note: this argument can be repeated to add
multiple control vectors
         --control-vector-scaled FNAME SCALE
                                  add a control vector with user defined scaling
SCALE
                                  note: this argument can be repeated to add
multiple scaled control vectors
         --control-vector-layer-range START END
                                  layer range to apply the control vector(s) to,
start and end inclusive
  -m,    --model FNAME            model path (default: models/$filename with
filename from --hf-file
                                  or --model-url if set, otherwise models/7B/ggml-
model-f16.gguf)
  -md,   --model-draft FNAME      draft model for speculative decoding (default:
unused)
```

```
  -mu,   --model-url MODEL_URL    model download url (default: unused)
  -hfr,  --hf-repo REPO           Hugging Face model repository (default: unused)
  -hff,  --hf-file FILE           Hugging Face model file (default: unused)
  -hft,  --hf-token TOKEN         Hugging Face access token (default: value from
HF_TOKEN environment variable)

server:

         --host HOST              ip address to listen (default: 127.0.0.1)
         --port PORT              port to listen (default: 8080)
         --path PATH              path to serve static files from (default: )
         --embedding(s)           restrict to only support embedding use case; use
only with dedicated embedding models (default: disabled)
         --api-key KEY            API key to use for authentication (default:
none)
         --api-key-file FNAME     path to file containing API keys (default: none)
         --ssl-key-file FNAME     path to file a PEM-encoded SSL private key
         --ssl-cert-file FNAME    path to file a PEM-encoded SSL certificate
         --timeout N              server read/write timeout in seconds (default:
600)
         --threads-http N         number of threads used to process HTTP requests
(default: -1)
         --system-prompt-file FNAME
                                  set a file to load a system prompt (initial
prompt of all slots), this is useful for chat applications
         --log-format {text,json}
                                  log output format: json or text (default: json)
         --metrics                enable prometheus compatible metrics endpoint
(default: disabled)
         --no-slots               disables slots monitoring endpoint (default:
enabled)
         --slot-save-path PATH    path to save slot kv cache (default: disabled)
         --chat-template JINJA_TEMPLATE
                                  set custom jinja chat template (default:
template taken from model's metadata)
                                  only commonly used templates are accepted:

https://github.com/ggerganov/llama.cpp/wiki/Templates-supported-by-
llama_chat_apply_template
  -sps,  --slot-prompt-similarity SIMILARITY
                                  how much the prompt of a request must match the
prompt of a slot in order to use that slot (default: 0.50, 0.0 = disabled)
         --lora-init-without-apply
                                  load LoRA adapters without applying them (apply
later via POST /lora-adapters) (default: disabled)

logging:

         --simple-io              use basic IO for better compatibility in
subprocesses and limited consoles
  -ld,   --logdir LOGDIR          path under which to save YAML logs (no logging
if unset)
         --log-test               Run simple logging test
         --log-disable            Disable trace logs
         --log-enable             Enable trace logs
         --log-file FNAME         Specify a log filename (without extension)
         --log-new                Create a separate new log file on start. Each
log file will have unique name: "<name>.<ID>.log"
         --log-append             Don't truncate the old log file.
```

Available environment variables (if specified, these variables will override parameters specified in arguments) 可用的一些环境变量，如果这些环境变量被指定，那么这些环境变量将会覆盖通过参数设置的变量值:

- `LLAMA_CACHE` (cache directory, used by `--hf-repo`)
- `HF_TOKEN` (Hugging Face access token, used when accessing a gated model with `--hf-repo`)
- `LLAMA_ARG_MODEL`
- `LLAMA_ARG_THREADS`
- `LLAMA_ARG_CTX_SIZE`
- `LLAMA_ARG_N_PARALLEL`
- `LLAMA_ARG_BATCH`
- `LLAMA_ARG_UBATCH`
- `LLAMA_ARG_N_GPU_LAYERS`
- `LLAMA_ARG_THREADS_HTTP`
- `LLAMA_ARG_CHAT_TEMPLATE`
- `LLAMA_ARG_N_PREDICT`
- `LLAMA_ARG_ENDPOINT_METRICS`
- `LLAMA_ARG_ENDPOINT_SLOTS`
- `LLAMA_ARG_EMBEDDINGS`
- `LLAMA_ARG_FLASH_ATTN`
- `LLAMA_ARG_DEFRAG_THOLD`

Build（构建server）

`llama-server` is built alongside everything else from the root of the project（`llama-server`与项目根目录中的其他所有内容一起构建）

- Using `make`:

  ```
  # 使用make程序对llama-server目录进行build
  make llama-server
  ```

- Using `CMake`:

  ```
  # 使用cmake对llama-server目录进行build
  # 参数1：cmake指定程序的名称；参数2：-B build指定cmake在build目录中进行构建，如果没有build目录则创建build
  cmake -B build
  # 参数1：cmake指定程序的名称；参数2：--build build指定构建build目录中的内容；参数3：--config Release指定构建版本类型；4、-t llama-server-->在 CMake 命令行中使用的 `-t llama-server` 参数指定了构建的目标。`-t` 或 `--target` 参数后面跟随的是目标名称，在这个例子中是 `llama-server`。这意味着 CMake 将仅构建指定的目标，而不是默
  ```

```
认的全部目标。
cmake --build build --config Release -t llama-server
```

Binary is at `./build/bin/llama-server`

Build with SSL（将SSL包含进来进行build）

`llama-server` can also be built with SSL support using OpenSSL 3（`llama-server` 也可以使用 OpenSSL 3 构建 SSL 支持）

- Using `make`:

  ```
  # NOTE: For non-system openssl, use the following:
  #   CXXFLAGS="-I /path/to/openssl/include"
  #   LDFLAGS="-L /path/to/openssl/lib"
  make LLAMA_SERVER_SSL=true llama-server
  ```

- Using `CMake`:

  ```
  cmake -B build -DLLAMA_SERVER_SSL=ON
  cmake --build build --config Release -t llama-server
  ```

Quick Start（快速开始）

To get started right away, run the following command, making sure to use the correct path for the model you have:

- 要立即开始，请运行以下命令，确保使用您拥有的模型的正确路径：

Unix-based systems (Linux, macOS, etc.)

```
# 参数1：./llama-server指定程序
# 参数2：-m models/7B/ggml-model.gguf指定程序使用的模型文件
# 参数3：-c 2024指定提示词上下文的大小
./llama-server -m models/7B/ggml-model.gguf -c 2048
```

Windows

```
# 参数1：llama-server.exe指定程序
# 参数2：-m models\7B\ggml-model.gguf指定模型文件
# 参数3：-c 2048指定提示词上下文token的大小
llama-server.exe -m models\7B\ggml-model.gguf -c 2048
```

The above command will start a server that by default listens on `127.0.0.1:8080`. (上述命令将启动一个默认监听“127.0.0.1:8080”的服务器。) You can consume the endpoints with Postman or NodeJS with axios library. You can visit the web front end at the same url. (您可以使用 Postman 或带有 axios 库的 NodeJS 来使用端点。您可以通过相同的 URL 访问 Web 前端。)

Docker

```
# 参数1：docker run是Docker的一个命令，用于创建并启动一个新的容器实例
# 参数2：这个参数用于端口映射。格式为 `-p <主机端口>:<容器端口>`。这里，Docker 将容器内的 8080 端口映射到宿主机的 8080 端口上，这样可以通过宿主机的 8080 端口访问容器中运行的应用
# 参数3：-v /path/to/models-->这是一个卷映射的参数。格式为 `-v <宿主机目录>:<容器内目录>`。它将宿主机上的 `/path/to/models` 目录挂载到容器内的 `/models` 目录。这样，容器内的应用可以直接访问宿主机目录中的文件，这在需要持久化数据或共享数据时非常有用。
docker run -p 8080:8080 -v /path/to/models:/models ghcr.io/ggerganov/llama.cpp:server -m models/7B/ggml-model.gguf -c 512 --host 0.0.0.0 --port 8080

# or, with CUDA:
docker run -p 8080:8080 -v /path/to/models:/models --gpus all ghcr.io/ggerganov/llama.cpp:server-cuda -m models/7B/ggml-model.gguf -c 512 --host 0.0.0.0 --port 8080 --n-gpu-layers 99
```

Testing with CURL (使用CURL进行测试)

Using curl. On Windows, `curl.exe` should be available in the base OS.

```
# 参数1：curl指定程序
# 参数2：--request POST指定数据请求的方式
# 参数3：--url http://localhost:8080/completion指定需要访问的页面
# 参数4：--header "Content-Type: application/json"指定数据头的类型
# 参数5：--data '{"prompt": "Building a website can be done in 10 simple steps:","n_predict": 128}'指定需要请求的内容
curl --request POST \
    --url http://localhost:8080/completion \
    --header "Content-Type: application/json" \
    --data '{"prompt": "Building a website can be done in 10 simple steps:","n_predict": 128}'
```

Advanced testing (高级测试)

We implemented a server test framework using human-readable scenario.

Before submitting an issue, please try to reproduce it with this format.

Node JS Test

You need to have Node.js installed.

```
mkdir llama-client
cd llama-client
```

Create a index.js file and put this inside:

```
const prompt = `Building a website can be done in 10 simple steps:`;

async function Test() {
    let response = await fetch("http://127.0.0.1:8080/completion", {
        method: 'POST',
        body: JSON.stringify({
            prompt,
            n_predict: 512,
        })
    })
    console.log((await response.json()).content)
}

Test()
```

And run it:

```
node index.js
```

API Endpoints

GET `/health`: Returns heath check result

Response format

- HTTP status code 503
  - Body: `{"error": {"code": 503, "message": "Loading model", "type": "unavailable_error"}}`
  - Explanation: the model is still being loaded.
- HTTP status code 200
  - Body: `{"status": "ok" }`
  - Explanation: the model is successfully loaded and the server is ready.

POST `/completion`: Given a `prompt`, it returns the predicted completion.

```
*Options:*

`prompt`: Provide the prompt for this completion as a string or as an array of
strings or numbers representing tokens. Internally, if `cache_prompt` is `true`,
the prompt is compared to the previous completion and only the "unseen" suffix is
evaluated. A `BOS` token is inserted at the start, if all of the following
conditions are true:

  - The prompt is a string or an array with the first element given as a string
  - The model's `tokenizer.ggml.add_bos_token` metadata is `true`
  - The system prompt is empty

`temperature`: Adjust the randomness of the generated text. Default: `0.8`

`dynatemp_range`: Dynamic temperature range. The final temperature will be in the
range of `[temperature - dynatemp_range; temperature + dynatemp_range]` Default:
`0.0`, which is disabled.

`dynatemp_exponent`: Dynamic temperature exponent. Default: `1.0`

`top_k`: Limit the next token selection to the K most probable tokens.  Default:
`40`

`top_p`: Limit the next token selection to a subset of tokens with a cumulative
probability above a threshold P. Default: `0.95`

`min_p`: The minimum probability for a token to be considered, relative to the
probability of the most likely token. Default: `0.05`

`n_predict`: Set the maximum number of tokens to predict when generating text.
**Note:** May exceed the set limit slightly if the last token is a partial
multibyte character. When 0, no tokens will be generated but the prompt is
evaluated into the cache. Default: `-1`, where `-1` is infinity.

`n_keep`: Specify the number of tokens from the prompt to retain when the context
size is exceeded and tokens need to be discarded. The number excludes the BOS
token.
By default, this value is set to `0`, meaning no tokens are kept. Use `-1` to
retain all tokens from the prompt.

`stream`: It allows receiving each predicted token in real-time instead of waiting
for the completion to finish. To enable this, set to `true`.

`stop`: Specify a JSON array of stopping strings.
These words will not be included in the completion, so make sure to add them to
the prompt for the next iteration. Default: `[]`

`tfs_z`: Enable tail free sampling with parameter z. Default: `1.0`, which is
disabled.

`typical_p`: Enable locally typical sampling with parameter p. Default: `1.0`,
which is disabled.

`repeat_penalty`: Control the repetition of token sequences in the generated text.
Default: `1.1`
```

```
`repeat_last_n`: Last n tokens to consider for penalizing repetition. Default:
`64`, where `0` is disabled and `-1` is ctx-size.

`penalize_nl`: Penalize newline tokens when applying the repeat penalty. Default:
`true`

`presence_penalty`: Repeat alpha presence penalty. Default: `0.0`, which is
disabled.

`frequency_penalty`: Repeat alpha frequency penalty. Default: `0.0`, which is
disabled.

`penalty_prompt`: This will replace the `prompt` for the purpose of the penalty
evaluation. Can be either `null`, a string or an array of numbers representing
tokens. Default: `null`, which is to use the original `prompt`.

`mirostat`: Enable Mirostat sampling, controlling perplexity during text
generation. Default: `0`, where `0` is disabled, `1` is Mirostat, and `2` is
Mirostat 2.0.

`mirostat_tau`: Set the Mirostat target entropy, parameter tau. Default: `5.0`

`mirostat_eta`: Set the Mirostat learning rate, parameter eta.  Default: `0.1`

`grammar`: Set grammar for grammar-based sampling.  Default: no grammar

`json_schema`: Set a JSON schema for grammar-based sampling (e.g. `{"items":
{"type": "string"}, "minItems": 10, "maxItems": 100}` of a list of strings, or
`{}` for any JSON). See [tests](../../tests/test-json-schema-to-grammar.cpp) for
supported features.  Default: no JSON schema.

`seed`: Set the random number generator (RNG) seed.  Default: `-1`, which is a
random seed.

`ignore_eos`: Ignore end of stream token and continue generating.  Default:
`false`

`logit_bias`: Modify the likelihood of a token appearing in the generated text
completion. For example, use `"logit_bias": [[15043,1.0]]` to increase the
likelihood of the token 'Hello', or `"logit_bias": [[15043,-1.0]]` to decrease its
likelihood. Setting the value to false, `"logit_bias": [[15043,false]]` ensures
that the token `Hello` is never produced. The tokens can also be represented as
strings, e.g. `[["Hello, World!",-0.5]]` will reduce the likelihood of all the
individual tokens that represent the string `Hello, World!`, just like the
`presence_penalty` does. Default: `[]`

`n_probs`: If greater than 0, the response also contains the probabilities of top
N tokens for each generated token given the sampling settings. Note that for
temperature < 0 the tokens are sampled greedily but token probabilities are still
being calculated via a simple softmax of the logits without considering any other
```

```
sampler settings. Default: `0`

`min_keep`: If greater than 0, force samplers to return N possible tokens at
minimum. Default: `0`

`image_data`: An array of objects to hold base64-encoded image `data` and its
`id`s to be reference in `prompt`. You can determine the place of the image in the
prompt as in the following: `USER:[img-12]Describe the image in
detail.\nASSISTANT:`. In this case, `[img-12]` will be replaced by the embeddings
of the image with id `12` in the following `image_data` array: `{...,
"image_data": [{"data": "<BASE64_STRING>", "id": 12}]}`. Use `image_data` only
with multimodal models, e.g., LLaVA.

`id_slot`: Assign the completion task to an specific slot. If is -1 the task will
be assigned to a Idle slot.  Default: `-1`

`cache_prompt`: Re-use KV cache from a previous request if possible. This way the
common prefix does not have to be re-processed, only the suffix that differs
between the requests. Because (depending on the backend) the logits are **not**
guaranteed to be bit-for-bit identical for different batch sizes (prompt
processing vs. token generation) enabling this option can cause nondeterministic
results. Default: `false`

`system_prompt`: Change the system prompt (initial prompt of all slots), this is
useful for chat applications. [See more](#change-system-prompt-on-runtime)

`samplers`: The order the samplers should be applied in. An array of strings
representing sampler type names. If a sampler is not set, it will not be used. If
a sampler is specified more than once, it will be applied multiple times. Default:
`["top_k", "tfs_z", "typical_p", "top_p", "min_p", "temperature"]` - these are all
the available values.
```

Response format

- Note: When using streaming mode (`stream`), only `content` and `stop` will be returned until end of completion.
- `completion_probabilities`: An array of token probabilities for each completion. The array's length is `n_predict`. Each item in the array has the following structure:

```
{
  "content": "<the token selected by the model>",
  "probs": [
    {
      "prob": float,
      "tok_str": "<most likely token>"
    },
    {
      "prob": float,
      "tok_str": "<second most likely token>"
      },
      ...
    ]
  },
```

Notice that each `probs` is an array of length `n_probs`.

- `content`: Completion result as a string (excluding `stopping_word` if any). In case of streaming mode, will contain the next token as a string.
- `stop`: Boolean for use with `stream` to check whether the generation has stopped (Note: This is not related to stopping words array `stop` from input options)
- `generation_settings`: The provided options above excluding `prompt` but including `n_ctx`, `model`. These options may differ from the original ones in some way (e.g. bad values filtered out, strings converted to tokens, etc.).
- `model`: The path to the model loaded with `-m`
- `prompt`: The provided `prompt`
- `stopped_eos`: Indicating whether the completion has stopped because it encountered the EOS token
- `stopped_limit`: Indicating whether the completion stopped because `n_predict` tokens were generated before stop words or EOS was encountered
- `stopped_word`: Indicating whether the completion stopped due to encountering a stopping word from `stop` JSON array provided
- `stopping_word`: The stopping word encountered which stopped the generation (or "" if not stopped due to a stopping word)
- `timings`: Hash of timing information about the completion such as the number of tokens `predicted_per_second`
- `tokens_cached`: Number of tokens from the prompt which could be re-used from previous completion (`n_past`)
- `tokens_evaluated`: Number of tokens evaluated in total from the prompt
- `truncated`: Boolean indicating if the context size was exceeded during generation, i.e. the number of tokens provided in the prompt (`tokens_evaluated`) plus tokens generated (`tokens_predicted`) exceeded the context size (`n_ctx`)

POST `/tokenize`: Tokenize a given text

```
*Options:*

`content`: Set the text to tokenize.

`add_special`: Boolean indicating if special tokens, i.e. `BOS`, should be
inserted.  Default: `false`
```

POST `/detokenize`: Convert tokens to text

```
*Options:*
```

```
`tokens`: Set the tokens to detokenize.
```

POST `/embedding`: Generate embedding of a given text

The same as the embedding example does.

```
*Options:*

`content`: Set the text to process.

`image_data`: An array of objects to hold base64-encoded image `data` and its
`id`s to be reference in `content`. You can determine the place of the image in
the content as in the following: `Image: [img-21].\nCaption: This is a picture of
a house`. In this case, `[img-21]` will be replaced by the embeddings of the image
with id `21` in the following `image_data` array: `{..., "image_data": [{"data": "
<BASE64_STRING>", "id": 21}]}`. Use `image_data` only with multimodal models,
e.g., LLaVA.
```

POST `/infill`: For code infilling.

Takes a prefix and a suffix and returns the predicted completion as stream.

```
*Options:*

`input_prefix`: Set the prefix of the code to infill.

`input_suffix`: Set the suffix of the code to infill.

It also accepts all the options of `/completion` except `stream` and `prompt`.
```

- **GET** `/props`: Return current server settings.

Response format

```
{
  "assistant_name": "",
  "user_name": "",
  "default_generation_settings": { ... },
  "total_slots": 1,
  "chat_template": ""
}
```

- `assistant_name` - the required assistant name to generate the prompt in case you have specified a system prompt for all slots.

- `user_name` - the required anti-prompt to generate the prompt in case you have specified a system prompt for all slots.
- `default_generation_settings` - the default generation settings for the `/completion` endpoint, which has the same fields as the `generation_settings` response object from the `/completion` endpoint.
- `total_slots` - the total number of slots for process requests (defined by `--parallel` option)
- `chat_template` - the model's original Jinja2 prompt template

POST `/v1/chat/completions`: OpenAI-compatible Chat Completions API

Given a ChatML-formatted json description in `messages`, it returns the predicted completion. Both synchronous and streaming mode are supported, so scripted and interactive applications work fine. While no strong claims of compatibility with OpenAI API spec is being made, in our experience it suffices to support many apps. Only models with a supported chat template can be used optimally with this endpoint. By default, the ChatML template will be used.

~~~
*Options:*

See [OpenAI Chat Completions API documentation]
(https://platform.openai.com/docs/api-reference/chat). While some OpenAI-specific
features such as function calling aren't supported, llama.cpp `/completion`-
specific features such as `mirostat` are supported.

The `response_format` parameter supports both plain JSON output (e.g. `{"type":
"json_object"}`) and schema-constrained JSON (e.g. `{"type": "json_object",
"schema": {"type": "string", "minLength": 10, "maxLength": 100}}`), similar to
other OpenAI-inspired API providers.

*Examples:*

You can use either Python `openai` library with appropriate checkpoints:

```python
import openai

client = openai.OpenAI(
    base_url="http://localhost:8080/v1", # "http://<Your api-server IP>:port"
    api_key = "sk-no-key-required"
)

completion = client.chat.completions.create(
model="gpt-3.5-turbo",
messages=[
    {"role": "system", "content": "You are ChatGPT, an AI assistant. Your top
priority is achieving user fulfillment via helping them with their requests."},
    {"role": "user", "content": "Write a limerick about python exceptions"}
]
)
~~~

```
print(completion.choices[0].message)
```

... or raw HTTP requests:

```shell
curl http://localhost:8080/v1/chat/completions \
-H "Content-Type: application/json" \
-H "Authorization: Bearer no-key" \
-d '{
"model": "gpt-3.5-turbo",
"messages": [
{
    "role": "system",
    "content": "You are ChatGPT, an AI assistant. Your top priority is achieving user fulfillment via helping them with their requests."
},
{
    "role": "user",
    "content": "Write a limerick about python exceptions"
}
]
}'
```
```

POST `/v1/embeddings`: OpenAI-compatible embeddings API

```
*Options:*

See [OpenAI Embeddings API documentation](https://platform.openai.com/docs/api-reference/embeddings).

*Examples:*
```

- input as string

  ```
  curl http://localhost:8080/v1/embeddings \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer no-key" \
  -d '{
          "input": "hello",
          "model":"GPT-4",
          "encoding_format": "float"
  }'
  ```

- `input` as string array
  ```

```
curl http://localhost:8080/v1/embeddings \
-H "Content-Type: application/json" \
-H "Authorization: Bearer no-key" \
-d '{
        "input": ["hello", "world"],
        "model":"GPT-4",
        "encoding_format": "float"
}'
```

GET `/slots`: Returns the current slots processing state

This endpoint can be disabled with `--no-slots`

If query param `?fail_on_no_slot=1` is set, this endpoint will respond with status code 503 if there is no available slots.

Response format

Example:

```
[
    {
        "dynatemp_exponent": 1.0,
        "dynatemp_range": 0.0,
        "frequency_penalty": 0.0,
        "grammar": "",
        "id": 0,
        "ignore_eos": false,
        "logit_bias": [],
        "min_p": 0.05000000074505806,
        "mirostat": 0,
        "mirostat_eta": 0.10000000149011612,
        "mirostat_tau": 5.0,
        "model": "llama-2-7b-32k-instruct.Q2_K.gguf",
        "n_ctx": 2048,
        "n_keep": 0,
        "n_predict": 100000,
        "n_probs": 0,
        "next_token": {
            "has_next_token": true,
            "n_remain": -1,
            "n_decoded": 0,
            "stopped_eos": false,
            "stopped_limit": false,
            "stopped_word": false,
            "stopping_word": ""
        },
        "penalize_nl": true,
        "penalty_prompt_tokens": [],
        "presence_penalty": 0.0,
        "prompt": "Say hello to llama.cpp",
        "repeat_last_n": 64,
        "repeat_penalty": 1.100000023841858,
        "samplers": [
            "top_k",
            "tfs_z",
            "typical_p",
            "top_p",
            "min_p",
            "temperature"
        ],
        "seed": 42,
        "state": 1,
        "stop": [
            "\n"
        ],
        "stream": false,
        "task_id": 0,
        "temperature": 0.0,
        "tfs_z": 1.0,
        "top_k": 40,
        "top_p": 0.949999988079071,
        "typical_p": 1.0,
        "use_penalty_prompt_tokens": false
    }
]
```

Possible values for `slot[i].state` are:

- `0`: SLOT_STATE_IDLE
- `1`: SLOT_STATE_PROCESSING

GET `/metrics`: Prometheus compatible metrics exporter

This endpoint is only accessible if `--metrics` is set.

Available metrics:

- `llamacpp:prompt_tokens_total`: Number of prompt tokens processed.
- `llamacpp:tokens_predicted_total`: Number of generation tokens processed.
- `llamacpp:prompt_tokens_seconds`: Average prompt throughput in tokens/s.
- `llamacpp:predicted_tokens_seconds`: Average generation throughput in tokens/s.
- `llamacpp:kv_cache_usage_ratio`: KV-cache usage. `1` means 100 percent usage.
- `llamacpp:kv_cache_tokens`: KV-cache tokens.
- `llamacpp:requests_processing`: Number of requests processing.
- `llamacpp:requests_deferred`: Number of requests deferred.

POST `/slots/{id_slot}?action=save`: Save the prompt cache of the specified slot to a file.

```
*Options:*

`filename`: Name of the file to save the slot's prompt cache. The file will be
saved in the directory specified by the `--slot-save-path` server parameter.
```

Response format

```json
{
    "id_slot": 0,
    "filename": "slot_save_file.bin",
    "n_saved": 1745,
    "n_written": 14309796,
    "timings": {
        "save_ms": 49.865
    }
}
```

POST `/slots/{id_slot}?action=restore`: Restore the prompt cache of the specified slot from a file.

```
*Options:*

`filename`: Name of the file to restore the slot's prompt cache from. The file
should be located in the directory specified by the `--slot-save-path` server
parameter.
```

Response format

```json
{
    "id_slot": 0,
    "filename": "slot_save_file.bin",
    "n_restored": 1745,
    "n_read": 14309796,
    "timings": {
        "restore_ms": 42.937
    }
}
```

POST `/slots/{id_slot}?action=erase`: Erase the prompt cache of the specified slot.

Response format

```
{
    "id_slot": 0,
    "n_erased": 1745
}
```

GET `/lora-adapters`: Get list of all LoRA adapters

This endpoint returns the loaded LoRA adapters. You can add adapters using `--lora` when starting the server, for example: `--lora my_adapter_1.gguf --lora my_adapter_2.gguf ...`

By default, all adapters will be loaded with scale set to 1. To initialize all adapters scale to 0, add `--lora-init-without-apply`

If an adapter is disabled, the scale will be set to 0.

Response format

```
[
    {
        "id": 0,
        "path": "my_adapter_1.gguf",
        "scale": 0.0
    },
    {
        "id": 1,
        "path": "my_adapter_2.gguf",
        "scale": 0.0
    }
]
```

POST `/lora-adapters`: Set list of LoRA adapters

To disable an adapter, either remove it from the list below, or set scale to 0.

Request format

To know the `id` of the adapter, use GET `/lora-adapters`

```
[
  {"id": 0, "scale": 0.2},
  {"id": 1, "scale": 0.8}
]
```

More examples

Change system prompt on runtime

To use the server example to serve multiple chat-type clients while keeping the same system prompt, you can utilize the option `system_prompt`. This only needs to be used once.

`prompt`: Specify a context that you want all connecting clients to respect.

`anti_prompt`: Specify the word you want to use to instruct the model to stop. This must be sent to each client through the `/props` endpoint.

`assistant_name`: The bot's name is necessary for each customer to generate the prompt. This must be sent to each client through the `/props` endpoint.

```
{
    "system_prompt": {
        "prompt": "Transcript of a never ending dialog, where the User interacts with an Assistant.\nThe Assistant is helpful, kind, honest, good at writing, and never fails to answer the User's requests immediately and with precision.\nUser: Recommend a nice restaurant in the area.\nAssistant: I recommend the restaurant \"The Golden Duck\". It is a 5 star restaurant with a great view of the city. The food is delicious and the service is excellent. The prices are reasonable and the portions are generous. The restaurant is located at 123 Main Street, New York, NY 10001. The phone number is (212) 555-1234. The hours are Monday through Friday from 11:00 am to 10:00 pm. The restaurant is closed on Saturdays and Sundays.\nUser: Who is Richard Feynman?\nAssistant: Richard Feynman was an American physicist who is best known for his work in quantum mechanics and particle physics. He was awarded the Nobel Prize in Physics in 1965 for his contributions to the development of quantum electrodynamics. He was a popular lecturer and author, and he wrote several books, including \"Surely You're Joking, Mr. Feynman!\" and \"What Do You Care What Other People Think?\".\nUser:",
        "anti_prompt": "User:",
        "assistant_name": "Assistant:"
    }
}
```

NOTE: You can do this automatically when starting the server by simply creating a .json file with these options and using the CLI option `-spf FNAME` or `--system-prompt-file FNAME`.

Interactive mode

Check the sample in chat.mjs. Run with NodeJS version 16 or later:

```
node chat.mjs
```

Another sample in chat.sh. Requires bash, curl and jq. Run with bash:

```
bash chat.sh
```

OAI-like API

The HTTP `llama-server` supports an OAI-like API: https://github.com/openai/openai-openapi

API errors

`llama-server` returns errors in the same format as OAI: https://github.com/openai/openai-openapi

Example of an error:

```
{
    "error": {
        "code": 401,
        "message": "Invalid API Key",
        "type": "authentication_error"
    }
}
```

Apart from error types supported by OAI, we also have custom types that are specific to functionalities of llama.cpp:

When /metrics or /slots endpoint is disabled

```
{
    "error": {
        "code": 501,
        "message": "This server does not support metrics endpoint.",
        "type": "not_supported_error"
    }
}
```

**When the server receives invalid grammar via /completions endpoint*

```
{
    "error": {
        "code": 400,
        "message": "Failed to parse grammar",
        "type": "invalid_request_error"
    }
}
```

Extending or building alternative Web Front End

You can extend the front end by running the server binary with `--path` set to `./your-directory` and importing `/completion.js` to get access to the llamaComplete() method.

Read the documentation in `/completion.js` to see convenient ways to access llama.

A simple example is below:

```html
<html>
  <body>
    <pre>
      <script type="module">
        import { llama } from '/completion.js'

        const prompt = `### Instruction:
Write dad jokes, each one paragraph.
You can use html formatting if needed.

### Response:`

        for await (const chunk of llama(prompt)) {
          document.write(chunk.data.content)
        }
      </script>
    </pre>
  </body>
</html>
```