

一、QwQ-32B 概述

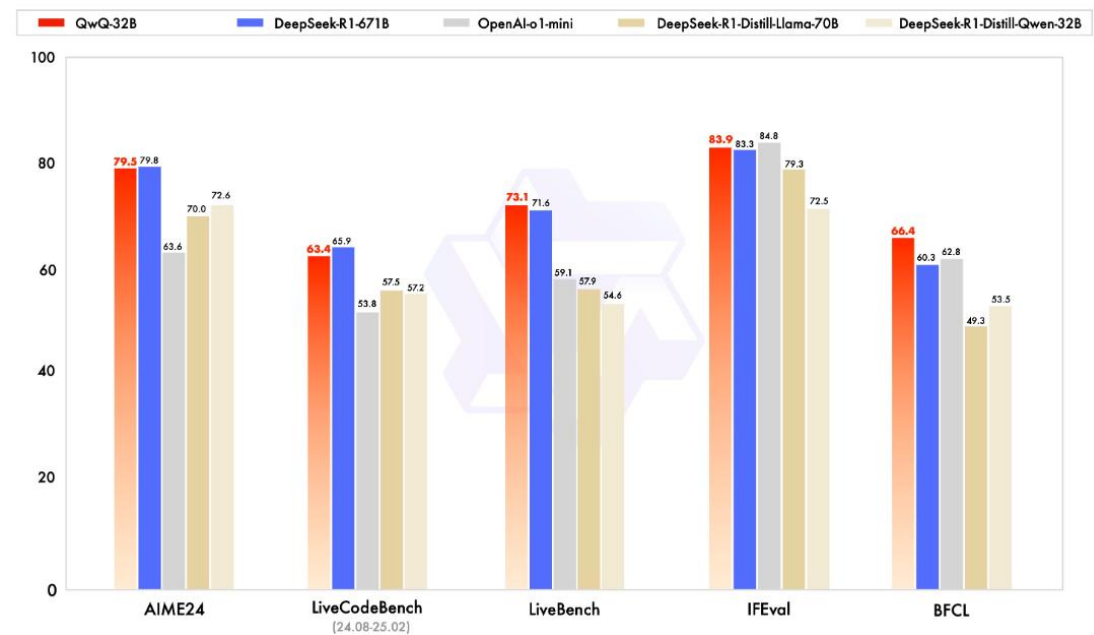
1. QwQ-32B 基本信息

发布时间：2025 年 3 月 6 日，由阿里巴巴集团旗下 Qwen 团队正式发布。

定位：一款轻量化、高效推理的开源大语言模型，对标 DeepSeek-R1 等大规模模型，目标是通过技术创新实现“小参数、高性能”。

开源生态：采用 Apache 2.0 开源协议，模型权重已在 Hugging Face 和 ModelScope 平台发布，支持 Ollama 等工具快速部署，适配消费级显卡（如双 Nvidia RTX 4090），显存需求较同类模型降低 70%。

QwQ-32B 在一系列基准测试中进行了评估，测试了数学推理、编程能力和通用能力。以下结果展示了 QwQ-32B 与其他领先模型的性能对比，包括 DeepSeek-R1-Distilled-Qwen-32B、DeepSeek-R1-Distilled-Llama-70B、OpenAI-o1-mini 以及原始的 DeepSeek-R1。



2. QwQ-32B 的主要功能

强大的推理能力：在数学推理、编程任务和通用能力测试中表现出色，性能媲美更大规模参数数量的模型。

智能体（Agent）能力：支持进行批判性思考，根据环境反馈调整推理过程，适用于复杂任务的动态决策。

多领域适应性：基于强化学习训练，模型在数学、编程和通用能力上均有显

著提升。

3. QwQ-32B 的技术原理

强化学习训练：模型针对数学和编程任务进行 RL 训练。数学任务基于校验答案正确性提供反馈，编程任务基于代码执行结果评估反馈。随后，模型进入通用能力训练阶段，用通用奖励模型和基于规则的验证器进一步提升性能。

预训练基础模型：QwQ-32B 基于强大的预训练模型（如 Qwen2.5-32B），大规模预训练获得广泛的语言和逻辑能力。强化学习在此基础上进一步优化模型的推理能力，让模型在特定任务上表现更优。

智能体集成：模型集成智能体能力，根据环境反馈动态调整推理策略，实现更复杂的任务处理。

4. QwQ-32B 项目

在线体验：<https://modelscope.cn/studios/Qwen/QwQ-32B-Demo>

下载地址：<https://modelscope.cn/collections/QwQ-32B-0f1806b8a8514a>

二、QwQ-32B 本地部署

1. 硬件配置要求：

1) GPU 显存

量化版本：若使用 4bit 量化（如 32b-q4_K_M），24GB 显存的显卡（如 Nvidia RTX3090、4090）即可支持推理。

原版模型（FP16）：需更高显存（约 30GB 以上），建议使用 A100 40GB 或 H100 80GB 等更高配置显卡。

优化特性：QwQ-32B 通过强化学习优化，参数量仅为 DeepSeek-R1 的 1/20，显著降低显存占用。

2) CPU 与内存

CPU：建议采用多核高性能处理器（如 Intel i9 或 AMD Ryzen 9 系列），以支持模型加载与并行计算。

内存：至少 64GB DDR4，推荐 128GB 以上，以处理长上下文窗口（131072 tokens）。

3) 存储

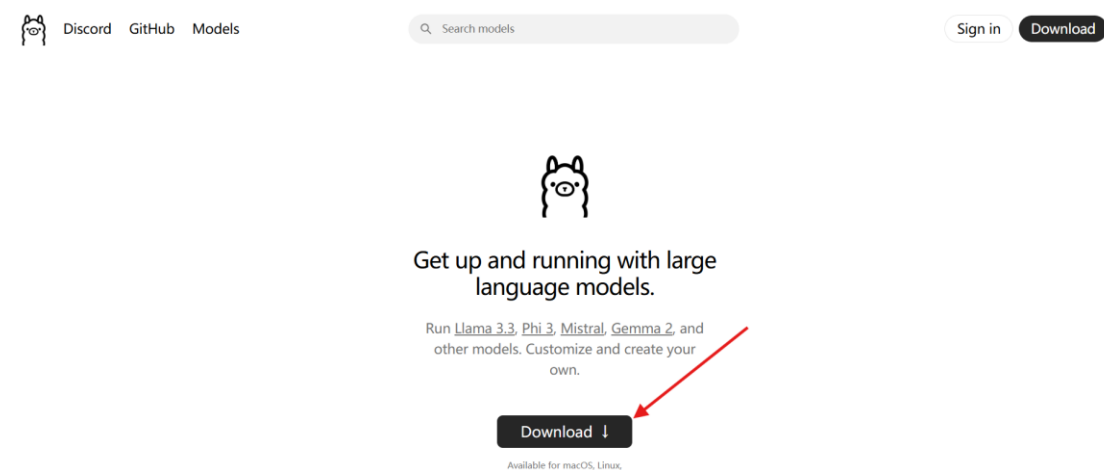
模型文件大小约 60-120GB（视量化版本而定），需预留充足存储空间。

2. Ollama 部署 QwQ-32B。

1) 硬件配置。

测试过程采用的服务器配置为：CPU 英特尔至强 Max 9468 * 2、GPU H GX H20(96GB) * 8、内存 64GB * 32、存储 3.84T Nvme * 4。

2) 打开 Ollama 官方网站：<https://ollama.com/>，从官网下载对应操作系统安装包（后续过程以 Linux 版本为例）。



3) 下载 Linux 版本 Ollama 工具。打开终端命令窗口，运行：`curl -fsSL https://ollama.com/install.sh | sh`



如果 Ollama 工具下载过程出现缓慢或本地网络无法访问情况，可以转至 GitHub（<https://github.com/ollama/ollama/releases>）上下载最新 Release 版本。

▼ Assets 14

ollama-darwin.tgz	21.2 MB	2 days ago
Ollama-darwin.zip	175 MB	2 days ago
ollama-linux-amd64-rocm.tgz	1.19 GB	2 days ago
ollama-linux-amd64.tgz	1.6 GB	2 days ago
ollama-linux-arm64-jetpack5.tgz	451 MB	2 days ago
ollama-linux-arm64-jetpack6.tgz	343 MB	2 days ago
ollama-linux-arm64.tgz	1.6 GB	2 days ago
ollama-windows-amd64-rocm.zip	368 MB	2 days ago
ollama-windows-amd64.zip	1.59 GB	2 days ago
ollama-windows-arm64.zip	20.4 MB	2 days ago
OllamaSetup.exe	1000 MB	2 days ago
sha256sum.txt	1018 Bytes	2 days ago
Source code (zip)		2 days ago

4) 解压安装。

`sudo tar -zxvf ollama-linux-amd64.tgz -C /usr/local`, 其中/usr/local 为解压安装目录（若安装至其它目录下，注意环境变量设置）。

创建服务配置文件：`/etc/systemd/system/ollama.service`，直接在终端命令窗口中运行命令：`sudo vi /etc/systemd/system/ollama.service`，输入以下内容。

```
[Unit]
Description=Ollama Service
After=network-online.target

[Service]
Environment="OLLAMA_HOST=0.0.0.0:11434"
ExecStart=/usr/local/bin/ollama serve
User=root
Group=root
Restart=always
RestartSec=3
# 指定模型保存路径
Environment="OLLAMA_MODELS=/home/GongHang/ollama/models"

[Install]
WantedBy=default.target
```

保存 ollama.service 成功后，执行以下命令，使 Ollama 服务生效。

```
GongHang@server03:~$ sudo systemctl daemon-reload
GongHang@server03:~$ sudo systemctl enable ollama
GongHang@server03:~$ sudo systemctl start ollama
GongHang@server03:~$ ollama -v
ollama version is 0.6.2
GongHang@server03:~$
```

5) 下载并运行 QwQ-32B 模型。

qwq

QwQ is the reasoning model of the Qwen series.

tools 32b

↓ 1M Pulls ⌚ Updated 10 days ago

32b ▾

🏷 8 Tags

ollama run qwq 

Updated 10 days ago		009cb3f08d74 · 20GB
model	arch qwen2 · parameters 32.8B · quantization Q4_K_M	20GB
params	{ "repeat_penalty": 1, "stop": ["< im_start >", "< im_end >"], ...	120B

QwQ-32B 模型参数约 32.8B，Ollama 默认下载 Q4_K_M 量化版本，模型文件约 20GB。

```
GongHang@server03:~$ ollama run qwq
pulling manifest
pulling 7ccc6415b2c7... 100% ██████████ 19 GB
pulling 41196096a061... 100% ██████████ 1.2 KB
pulling d18a5cc71b84... 100% ██████████ 11 KB
pulling e5229acc2492... 100% ██████████ 120 B
pulling 1972e9972e2e... 100% ██████████ 488 B
verifying sha256 digest
writing manifest
success
>>> 你是谁?
<think>
嗯，用户问“你是谁？”，我需要先确定他们想知道什么。可能他们刚接触我，或者想确认我的身份。首先，我应该明确我的身份是阿里巴巴集团旗下的通义实验室自主研发的超大规模语言模型，名字叫通义千问，英文名Qwen。然后，我需要说明我的主要功能，比如回答问题、创作文字、逻辑推理、编程等。还要提到我支持多语言，这样用户知道可以使用不同语言交流。

接下来，用户可能想知道我的应用场景，比如写故事、写公文、写邮件、写剧本、逻辑推理、编程等等。这些例子能帮助他们理解我的能力范围。另外，可能需要提到我的训练数据截止时间，这样用户知道我的知识更新到什么时候，比如2024年12月。

还要注意语气要友好，避免使用过于技术化的术语，保持简洁易懂。可能用户需要进一步的帮助，所以最后可以邀请他们提出具体问题，这样能更好地协助他们。要确保回答结构清晰，信息全面，同时自然流畅，不让用户感到信息过载。

另外，用户可能有隐藏的需求，比如他们可能想测试我的能力，或者寻找特定帮助。所以回答中覆盖多个方面，展示我的多功能性，这样用户能更快找到他们需要的部分。同时，避免使用Markdown格式，保持口语化，让回答更亲切。

</think>
你好！我是通义千问，英文名Qwen，是阿里巴巴集团旗下的通义实验室自主研发的超大规模语言模型。我能够帮助你：

- **回答问题**：无论是常识、专业知识，还是具体问题，我都会尽力为你解答。
- **创作文字**：写故事、公文、邮件、剧本等，我都可以帮你完成。
- **逻辑推理**：分析问题、提供逻辑建议，或帮你解决复杂问题。
- **编程**：我可以写代码、解释编程概念，甚至帮你调试程序。
- **多语言支持**：除了中文，我还支持英文、德语、法语、西班牙语等多种语言。

我的训练数据截止时间是2024年12月，这意味着我对2024年之前的信息有较好的了解，但可能对之后发生的事件或数据不太熟悉。

有什么需要帮助的吗？随时告诉我！

>>> █ end a message (/? for help)
```

3. vLLM 部署 QwQ-32B。

1) 硬件配置。

测试过程采用的服务器配置为：CPU 英特尔至强 Max 9468 * 2、GPU HG X H20(96GB) * 8、内存 64GB * 32、存储 3.84T Nvme * 4。

2) vLLM 安装。

- 通过 pip 指令安装，命令行：pip install vllm，建议在 python 虚拟环境中安装，避免破坏系统环境。
- 采用 docker 镜像安装，在 Linux 终端中执行命令：sudo docker pull vllm /vllm-openai:latest，拉取镜像。

3) 拉取镜像文件结束后，创建和启动一个 vllm-openai 容器。例：

```
sudo docker run -itd --ipc=host --gpus all --name vllm-openai -p 8000:8000 --rm -v modelscope/models:/workspace/models --entrypoint /bin/bash vllm/vllm-openai:latest
```

4) vLLM 部署 QwQ-32B。

a) 进入容器 bash。

运行 `sudo docker ps` 指令，找到 vllm-openai 容器 id；接着通过 vllm-openai 容器 id 进入 vllm-openai 的 bash 窗口。

```
sudo docker ps # 获取 vllm-openai 容器 id
```

```
sudo docker exec -it vllm-openai_id bash # 进入 vllm-openai 的 bash 窗口
```

b) 启动 vLLM 服务。

```
vllm serve model_tag --host 0.0.0.0 --port 8000 --api-key 1234567890 --gpu-memory-utilization 0.9 --tensor-parallel-size 4 --served-model-name QwQ_32B --disable-log-requests > vllm.log 2>&1 &
```

其中：

model_tag: 用于指定要加载的模型，可以是 Hugging Face 模型仓库中的模型名称，也可以是本地路径。

api-key: 启用 API 访问控制，客户端访问需提供此密钥。

gpu-memory-utilization: 指定 GPU 内存利用率，值为 0-1 的小数，默认为 0.9。

tensor-parallel-size: 设置 tensor 并行的数量（单机多 GPU 并行推理，指定 GPU 数量）。

served-model-name: 加载模型的别名（自定义）。

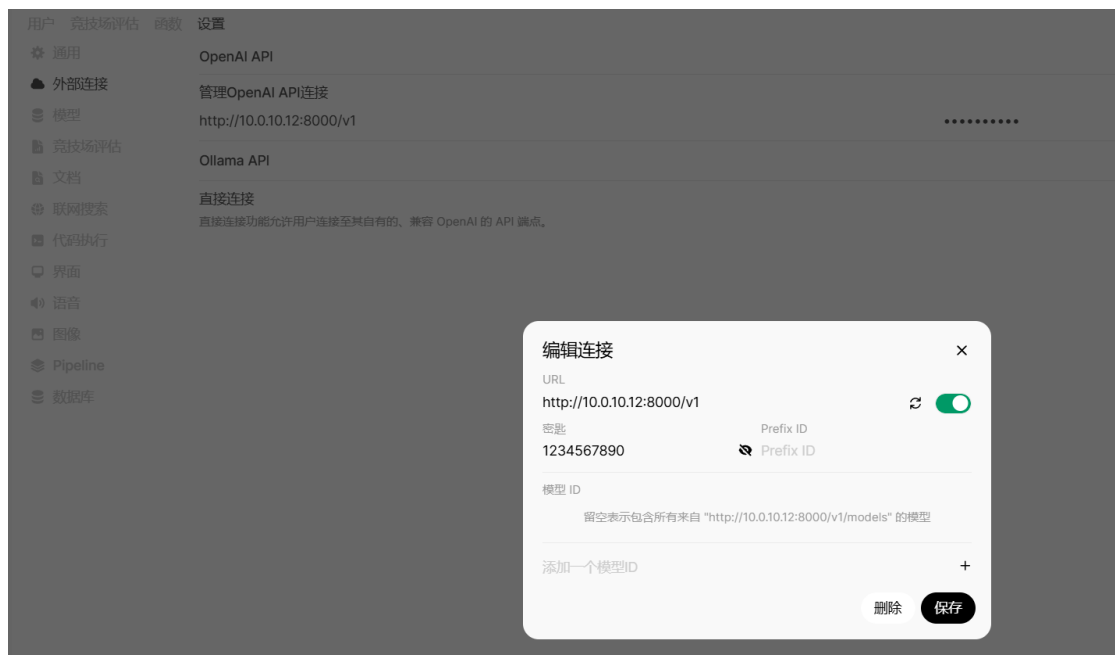
```
DeepSeek-R1-Distill-Llama-70B QwQ-32B
root@server02:/mnt# vllm serve QwQ-32B --host 0.0.0.0 --port 8000 --api-key 1234567890 --gpu-memory-utilization 0.9 --tensor-parallel-size 4 --served-model-name QwQ_32B --disable-log-requests
INFO 03-17 18:52:33 __init__.py:207] Automatically detected platform cuda.
INFO 03-17 18:52:33 api_server.py:912] vLLM API server version 0.7.3
```

```

(VllmWorkerProcess pid=853) INFO 03-17 18:53:47 custom_all_reduce.py:226] Registering 4515 cuda graph addresses
(VllmWorkerProcess pid=854) INFO 03-17 18:53:47 custom_all_reduce.py:226] Registering 4515 cuda graph addresses
(VllmWorkerProcess pid=854) INFO 03-17 18:53:48 model_runner.py:1562] Graph capturing finished in 18 secs, took 0.42 GiB
(VllmWorkerProcess pid=853) INFO 03-17 18:53:48 model_runner.py:1562] Graph capturing finished in 18 secs, took 0.42 GiB
(VllmWorkerProcess pid=852) INFO 03-17 18:53:48 model_runner.py:1562] Graph capturing finished in 18 secs, took 0.42 GiB
INFO 03-17 18:53:48 model_runner.py:1562] Graph capturing finished in 18 secs, took 0.42 GiB
INFO 03-17 18:53:48 llm_engine.py:436] init engine (profile, create kv cache, warmup model) took 26.76 seconds
INFO 03-17 18:53:48 api_server.py:958] Starting vLLM API server on http://0.0.0.0:8000
INFO 03-17 18:53:48 launcher.py:23] Available routes are:
INFO 03-17 18:53:48 launcher.py:31] Route: /openapi.json, Methods: GET, HEAD
INFO 03-17 18:53:48 launcher.py:31] Route: /docs, Methods: GET, HEAD
INFO 03-17 18:53:48 launcher.py:31] Route: /docs/oauth2-redirect, Methods: GET, HEAD
INFO 03-17 18:53:48 launcher.py:31] Route: /redoc, Methods: GET, HEAD
INFO 03-17 18:53:48 launcher.py:31] Route: /health, Methods: GET
INFO 03-17 18:53:48 launcher.py:31] Route: /ping, Methods: POST, GET
INFO 03-17 18:53:48 launcher.py:31] Route: /tokenize, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /detokenize, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /v1/models, Methods: GET
INFO 03-17 18:53:48 launcher.py:31] Route: /version, Methods: GET
INFO 03-17 18:53:48 launcher.py:31] Route: /v1/chat/completions, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /v1/completions, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /v1/embeddings, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /pooling, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /score, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /v1/score, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /v1/audio/transcriptions, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /rerank, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /v1/rerank, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /v2/rerank, Methods: POST
INFO 03-17 18:53:48 launcher.py:31] Route: /invocations, Methods: POST
INFO: Started server process [510]
INFO: Waiting for application startup.
INFO: Application startup complete.

```

5) 在 Open-WebUI 管理员界面中，设置 OpenAI API 外部连接（或接入 Ollama a API）。



6) 创建聊天窗口，选择 QwQ-32B 模型。



4. 适用场景建议

- 1) **Ollama**：个人开发者快速验证模型效果、低配置硬件（如仅有 16GB 内存的笔记本电脑）；需要快速交互式对话或原型开发。
- 2) 选择 **vLLM**：企业级 API 服务、高并发批量推理（如智能客服、文档处理）；需要高精度模型输出或定制化参数调整。

对比维度	Ollama	vLLM
核心定位	轻量级本地化工具，适合个人开发者和小规模实验	生产级推理框架，专注高并发、低延迟的企业级场景
硬件要求	支持 CPU 和 GPU，低显存占用 (默认使用量化模型)	必须依赖 Nvidia GPU，显存占用高
模型支持	内置预训练模型库（支持 1700+ 模型），自动下载量化版本（int4 为主）	需手动下载原始模型文件（如 Hugging Face 格式），支持更广泛模型
部署难度	一键安装，开箱即用，无需编程基础	需配置 Python 环境、CUDA 驱动，依赖技术经验
性能特征	单次推理速度快，但并发处理能力弱	高吞吐量，支持动态批处理和千级并发请求
资源管理	灵活调整资源占用，空闲时自动释放显存	显存占用固定，需预留资源应对峰值负载