

PR #51570 完整报告

vllm-project/vllm

[Rust][Benchmark] Load HF datasets from parquet shards via hf-hub, fixing truncated-cache sampling

合并时间: 2026-08-21 13:07

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51570>

执行摘要

本 PR 修复了 Rust 侧 `vllm-bench` 的 HF 数据集加载器一个隐蔽且危险的问题: 数据集被缓存为按 `dataset-config-split` 键控的 JSON 快照, 行数被第一次运行的请求量截断, 后续请求更多提示词时静默复用截断文件 (GSM8k 请求 1319 条实际只跑 100 条)。改造后, 数据改为通过 hf-hub 从 dataset-viewer 的 parquet 分片下载到标准 HF hub 缓存, 缓存单位是不可变 shard, 物理截断会因缺 parquet footer 而显式报错; 同时为私有数据集新增原生 parquet/JSON/JSONL 文件发现回退, 并把采样语义与 Python `vllm bench serve` 对齐。合并前两条 review 意见 (brotli codec、"default" config 歧义) 均在后续提交中修复, BugenZhao 批准合入。

功能与动机

PR body 明确描述了这个静默少采样 bug 的危害:

HF datasets were cached as JSON snapshots truncated to the row count of whatever run happened first... requesting the full 1319-prompt GSM8k test split yielded `samples=100 requested=1319 + skipping dataset oversampling`, so the benchmark ran on 100 prompts while claiming to be a GSM8k run.

这会让 benchmark 结果失真且难以察觉。此外 `/rows` 分页接口按官方文档只对拥有 parquet 导出的数据集有效, 删掉该回退不会损失能力。重构的核心原则是: 缓存内容绝不能依赖前一次运行的请求大小——把缓存单位从 "请求相关的快照" 变为 "不可变 shard", 从根上消除缓存污染。

实现拆解

- 缓存管线重构 (`hf_dataset.rs`): 删除自定义 `cache_dir()`、`sanitize_name()` 与 JSON 快照缓存, 改为经 hf-hub 下载到标准 HF hub 缓存 (`$HF_HOME/hub`), 与 hf download、Python `huggingface_hub` 双向共享。
- 主路径: parquet 分片读取 (`hf_dataset.rs`): `convert_parquet_shards` 列出 `refs/convert/parquet` revision 的分片, 经 `SerializedFileReader` 读取, `Row::to_json_value()` 喂给既有 `ColumnFormat serde_json` 列检测管线; `partial-{split}` (>5GB 数据集) 分片被正确包含。
- 回退路径: 私有数据集原生文件发现 (`hf_dataset.rs + hub.rs`): `/info` 失败时, `native_data_files` 与 `select_native_files` 借助新增的 `HubRepo::list_files()` 枚举 main 分

支文件，按 top_level_dirs config 目录与 path_matches_split split 命名约定匹配 *.parquet/*.json/*.jsonl; dataset_with_revision 支持指定 revision。

4. 采样语义对齐 Python (hf_dataset.rs + benchmark.rs) : download_hf_dataset 新签名接收 seed/disable_shuffle, 分片按 seed 打乱 (对应 datasets 库 streaming shuffle) , 读取最多 2 倍 num_rows_needed 行供有效性过滤回填; benchmark.rs 中 load_hf_dataset 输入从路径改为 entries 向量。
5. 认证链路统一 (hf_dataset.rs) : token 解析改为 HF_TOKEN 环境变量优先、hf_hub::Cache::from_env().token() 兜底, discovery 与下载共用。
6. 配置与文档配套: workspace 与 bench crate 引入 parquet (default-features = false, 无 arrow 依赖, review 后补齐 brotli 等 codec feature) ; Cargo.lock +133 行传递依赖 ; README 修正 --random-range-ratio 过时语义, AGENTS.md 更新模块描述。

测试配套: 未新增独立测试文件, 靠 cargo test -p vllm-bench (148 passed) 与 #[ignored] 网络测试 test_download_gsm8k、test_native_parquet 守护下载链路。

rust/src/bench/src/datasets/hf_dataset.rs

PR 核心文件: 重写下载管线, 删除 JSON 快照缓存, 新增 parquet 分片读取、原生文件发现与采样对齐逻辑。

```
// rust/src/bench/src/datasets/hf_dataset.rs —— HF 数据集下载主入口 (改动核心)
// SPDX-License-Identifier: Apache-2.0
// SPDX-FileCopyrightText: Copyright contributors to the vLLM project
```

```
use std::path::Path;
use std::sync::Arc;
```

```
use parquet::file::reader::{FileReader, SerializedFileReader};
use rand::rngs::StdRng;
use rand::seq::SliceRandom;
use rand::{Rng, SeedableRng};
```

```
use super::SampleRequest;
use crate::error::{BenchError, Result};
use crate::tokenizer::TokenizerKind;
```

```
/// 下载并解析 HF 数据集, 返回行数据而非缓存文件路径。
```

```
///
```

```
/// 行数据来自经 hf-hub 下载到标准 HF hub 缓存 (`$HF_HOME/hub`) 的数据文件:
```

```
/// 优先用 dataset-viewer 的 `refs/convert/parquet` 导出, 否则退回 main 分支上
```

```
/// 的原生 parquet/JSON/JSONL 文件 (dataset viewer 不处理的私有数据集走此路径)。
```

```
/// 分片顺序按 `seed` 打乱 (除非 `disable_shuffle`), 并读取最多 2 倍
```

```
/// `num_rows_needed` 的行, 使下游有效性过滤能回填真实行而不是复制样本。
```

```
pub async fn download_hf_dataset(
    dataset: &str,
    subset: Option<&str>,
    split: Option<&str>,
    num_rows_needed: usize,
```

```

seed: u64,
disable_shuffle: bool,
) -> Result<(Vec<serde_json::Value>, String, String)> {
    let encoded_dataset: String =
        url::form_urlencoded::byte_serialize(dataset.as_bytes()).collect();

    let mut client_builder =
        request::Client::builder().timeout(std::time::Duration::from_secs(120));

    // 与 hf-hub 下载一致的认证链路: HF_TOKEN 环境变量优先, 其次读
    // `hf auth login` 写入的 token 文件。此前 discovery 只认环境变量,
    // 导致已登录但未设环境变量的用户访问 gated 数据集时在发现阶段失败。
    let token = std::env::var("HF_TOKEN")
        .ok()
        .filter(!t.is_empty())
        .or_else(|| hf_hub::Cache::from_env().token());
    if let Some(token) = token {
        let mut headers = request::header::HeaderMap::new();
        let header_value = request::header::HeaderValue::from_str(&format!("Bearer {token}"))
            .map_err(|e| BenchError::Config(format!("Invalid HF token: {e}")))?;
        headers.insert(request::header::AUTHORIZATION, header_value);
        client_builder = client_builder.default_headers(headers);
    }

    let client = client_builder
        .build()
        .map_err(|e| BenchError::Config(format!("Failed to build HTTP client: {e}")))?;

    // 解析 config 与 split: 两者都显式给出时跳过 /info 发现调用;
    // 否则调 /info 发现可用 configs 与 splits, 私有数据集未被 viewer
    // 处理时该调用失败, 会走原生文件发现的回退路径。
    let (resolved_config, resolved_split) = if let (Some(cfg), Some(spl)) = (subset, split) {
        (cfg.to_string(), spl.to_string())
    } else if let Some(info) = {
        // ... (/info 请求与返回值解析, 略)
        None
    } {
        {}
        // ... (按 /info 结果解析 resolved_config / resolved_split, 略)
        unreachable!()
    } else {
        // ... (缺省 config/split 的解析, 含 review 中 "default" 歧义的处理, 略)
        return Err(BenchError::Config("cannot resolve config/split".into()));
    };

    // 后续主流程 (本片段之外) :
    // 1. convert_parquet_shards: 列出 refs/convert/parquet revision 的分片
    // (含 >5GB 数据集的 partial-{split} 分片), 逐个经 SerializedFileReader
    // 读取, 按 seed 打乱后取最多 2 * num_rows_needed 行;
    // 2. 无 parquet 导出时走 native_data_files + select_native_files 的

```

```

// 原生文件发现回退;
// 3. Row::to_json_value() 喂给既有 serde_json 列检测管线 (ColumnFormat) 。
let _ = (client, resolved_config, resolved_split);
unimplemented!("完整实现见 convert_parquet_shards / native_data_files 流程")
}

```

rust/src/bench/src/hub.rs

为分片下载与原生文件发现新增 dataset_with_revision 与 list_files 能力，是私有数据集回退路径的基础。

```

// rust/src/bench/src/hub.rs —— hf-hub 仓库句柄（新增能力）
// SPDX-License-Identifier: Apache-2.0
// SPDX-FileCopyrightText: Copyright contributors to the vLLM project

use std::path::PathBuf;

use hf_hub::api::tokio::{ApiBuilder, ApiRepo};
use hf_hub::{Repo, RepoType};

/// 对 HuggingFace Hub 仓库的句柄，下载统一走 hf-hub 的磁盘缓存
/// (`$HF_HOME/hub`)，与 `hf download` 和 Python `huggingface_hub` 互通。
pub struct HubRepo {
    repo: ApiRepo,
}

impl HubRepo {
    pub fn model(model_id: String) -> Result<Self, String> {
        Self::new(Repo::model(model_id))
    }

    pub fn dataset(repo_id: String) -> Result<Self, String> {
        Self::new(Repo::dataset(repo_id))
    }

    /// 以指定 revision 打开数据集仓库，用于下载 dataset-viewer 的
    /// `refs/convert/parquet` 自动转换分片。
    pub fn dataset_with_revision(repo_id: String, revision: String) -> Result<Self, String> {
        Self::new(Repo::with_revision(repo_id, RepoType::Dataset, revision))
    }

    fn new(repo: Repo) -> Result<Self, String> {
        let mut builder = ApiBuilder::from_env();
        if let Ok(token) = std::env::var("HF_TOKEN") {
            builder = builder.with_token(Some(token));
        }
        let api = builder.build().map_err(|e| format!("Failed to init HF API: {e}"))?;
        Ok(Self { repo: api.repo(repo) })
    }
}

```

```

/// 下载（或从缓存取回）仓库中的单个文件，认证由 hf-hub 依 HF_TOKEN
/// 或登录 token 文件处理。
pub async fn get(&self, filename: &str) -> Result<PathBuf, String> {
    self.repo.get(filename).await.map_err(|e| format!("{e}"))
}

/// 列出仓库在该 revision 下的全部文件路径，供私有数据集的原生
/// parquet/JSON/JSONL 文件发现使用。
pub async fn list_files(&self) -> Result<Vec<String>, String> {
    let info = self.repo.info().await.map_err(|e| format!("{e}"))?;
    Ok(info.siblings.into_iter().map(|s| s.rfilename).collect())
}
}

```

评论区精华

"Enable Brotli before accepting arbitrary Parquet shards... *SerializedFileReader* will reject it because *default-features* are disabled and the *brotli* codec feature is omitted here. This regresses those datasets from the previous */rows* path..." (chatgpt-codex-connector, P2)

"A safer fallback would be to error out (or warn explicitly) when multi-config layouts are detected... the opposite of the PR's stated goal of failing loudly instead of silently returning wrong data." (claude[bot] 关于 "default" config 回退)

"It correctly includes those shards rather than skipping them, so the split's row count isn't silently truncated there." (claude[bot] 关于 partial-{split} 分片，确认无误)

两条实质问题（brotli codec、config 歧义）均由提交 a58b63a 一次性修复，BugenZhao 最终给出 LGTM!。

风险与影响

- 回归风险：read_parquet_rows 对整行物化的内存占用高于原分页路径；select_native_files 的布局启发式对非常规仓库可能误选文件。
- 兼容性风险：num_prompts < split size 时的采样行集合变化，与历史 Rust bench 结果不可直接对比；全量 split 不受影响。
- 依赖成本：parquet 及其传递依赖（thrift、snap、lz4_flex、brotli）增加编译时长与二进制体积。
- 测试覆盖：无新增独立测试文件，关键路径只靠 #[ignored] 网络测试，常规 CI 检测弱。
- 正面影响：benchmark 数据可信度显著提升，冷缓存加载从约 14 次分页调用降到单分片下载（GSM8k 冷加载 1.2s），与 Python 实现逐 token 对齐（77527 = 77527）。

关联脉络

PR body 主动排查了同区域的其他 PR：

- #50277 (draft)：使 datasets-server endpoint 可配置以支持镜像路由，与本 PR 在 hf_dataset.rs 有文本冲突；本 PR 的 shard 下载（流量大头）已遵循 HF_ENDPOINT，方

向互补。

- #49641: 修复 Python loader 省略 `--hf-split` 的问题, 属同一功能线 (HF 数据集加载) 的 Python 侧对应。
- #51104: Rust 多轮 ShareGPT, PR body 确认与 `hf_dataset.rs` 无重叠。

结合近期同仓库历史 (如 #53054 Rust 前端 HY3 parser、#52892 用 `protox` 替换 `protoc`) , 可以看到 Rust 前端在工具链自洽、依赖精简与行为对齐 Python 实现上的持续投入; 本 PR 是其中 "benchmark 数据可信度 " 这一环的关键修复。