PR #22658 完整报告

sgl-project/sglang

PD streaming: batch notify + SSE fast path

合并时间：2026-04-22 17:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22658

执行摘要

- 一句话：批处理通知 +SSE 快速路径，PD streaming 吞吐 +20%
- 推荐动作：强烈推荐合并。该 PR 提供了明确的性能收益和可衡量的基准结果。值得精读的部分包括：如何用 msgspec 替换 Pydantic 进行 SSE 序列化，以及批量通知模式的设计权衡。后续可关注 IPC 序列化迁移（PR #21643）以进一步释放性能。

功能与动机

在高并发 PD streaming（如 2048 并发）下，*decode* 侧 tokenizer_manager 成为 CPU 瓶颈：*asyncio wakeup storm* 导致频繁上下文切换，每个 token 的 SSE chunk 构建 3 个 Pydantic 对象并调用 model_dump_json() 产生大量开销。

实现拆解

1. 批量通知机制(tokenizer_manager.py): 将 _handle_batch_output 改为 async，引入 pending_notify 字典，每积累 batch_notify_size（默认 16）个通知统一 *event.set()* 并 *await asyncio.sleep(0)* 让出事件循环，循环结束后 flush 剩余。
2. SSE 快速路径(serving_chat.py): 新增 msgspec.Struct 子类 (*_StreamDelta*, *_StreamChoice*, *_StreamChunk*) 和 *_fast_sse_content* 辅助函数，用预创建的 msgspec.json.Encoder 编码替代 Pydantic model_dump_json()，仅在 4 个热路径 yield 点使用。
3. 配置参数化(server_args.py): 添加 --batch-notify-sizeCLI 参数，默认 16，允许根据负载调整。

关键文件：

- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 API 入口；类别 source；类型 dependency-wiring；符号 *_StreamDelta*, *_StreamChoice*, *_StreamChunk*, *_fast_sse_content*）：替换 Pydantic 序列化为 msgspec，新增 *_fast_sse_content* 热路径，是性能优化的核心
- python/sglang/srt/managers/tokenizer_manager.py（模块 分词管理器；类别 source；类型 core-logic；符号 _handle_batch_output）：将 _handle_batch_output 改为异步，引入批量通知机制，减少 asyncio 唤醒风暴
- python/sglang/srt/server_args.py（模块 配置；类别 source；类型 core-logic）：新增 batch_notify_size 参数，使批量通知大小可配置

关键符号：_handle_batch_output, _fast_sse_content, _StreamDelta, _StreamChoice, _StreamChunk

关键源码片段

python/sglang/srt/entrypoints/openai/serving_chat.py

替换 Pydantic 序列化为 msgspec，新增 _fast_sse_content 热路径，是性能优化的核心

```
# 使用 msgspec.Struct 定义 SSE chunk 结构，omit_defaults=True 使输出更紧凑
class _StreamDelta(msgspec.Struct, omit_defaults=True):
    # reasoning_content 必须为必填（无默认值），因为 OpenAI SDK 的 ChoiceDelta 通过 extra
    暴露该字段
    # omit_defaults=True 下若默认 None 会丢失 key，导致客户端 AttributeError
    reasoning_content: Optional[str]
    role: Optional[str] = None
    content: Optional[str] = None

class _StreamChoice(msgspec.Struct):
    index: int
    delta: _StreamDelta
    logprobs: Optional[dict] = None
    finish_reason: Optional[str] = None
    matched_stop: Union[None, int, str] = None

class _StreamChunk(msgspec.Struct, omit_defaults=True):
    id: str
    object: str
    created: int
    model: str
    choices: List[_StreamChoice]
    usage: Optional[dict] = None

# 复用 msgspec.json.Encoder 实例避免重复创建
_stream_encoder = msgspec.json.Encoder()

# 热路径 SSE 构造：直接构建 struct，用 msgspec 编码，替代 Pydantic model_dump_json
# 参数覆盖所有 hot-path 字段：role, content, reasoning_content, finish_reason 等
# 非热路径（hidden_states, tool_calls 等）仍走原 Pydantic 路径
_SSE_DATA_B = b"data: "
_SSE_NL_B = b"

"

def _fast_sse_content(
    chunk_id: str, created: int, model: str, index: int,
    role: Optional[str] = None, content: Optional[str] = None,
    reasoning_content: Optional[str] = None,
    finish_reason: Optional[str] = None,
    logprobs: Optional[dict] = None,
```

```
    matched_stop: Union[None, int, str] = None,
    usage: Optional[dict] = None,
) -> str:
    delta = _StreamDelta(role=role, content=content, reasoning_content=reasoning_content)
    choice = _StreamChoice(index=index, delta=delta, logprobs=logprobs,
                           finish_reason=finish_reason, matched_stop=matched_stop)
    chunk = _StreamChunk(id=chunk_id, object="chat.completion.chunk",
                         created=created, model=model, choices=[choice], usage=usage)
    # 直接拼接 bytes 再 decode，避免中间字符串
    return (_SSE_DATA_B + _stream_encoder.encode(chunk) + _SSE_NL_B).decode()
```

python/sglang/srt/managers/tokenizer_manager.py

将 _handle_batch_output 改为异步，引入批量通知机制，减少 asyncio 唤醒风暴

```
# 在 handle_loop 中，对 batch 输出类型提前过滤，直接 await 异步处理
async def handle_loop(self):
    while True:
        with self.soft_watchdog.disable():
            recv_obj = await self.recv_from_detokenizer.recv_pyobj()
        if isinstance(recv_obj, (BatchStrOutput, BatchEmbeddingOutput, BatchTokenIDOutput)):
            await self._handle_batch_output(recv_obj)
        else:
            self._result_dispatcher(recv_obj)
        self.last_receive_tstamp = real_time()
        self.soft_watchdog.feed()

async def _handle_batch_output(self, recv_obj):
    pending_notify: dict[str, ReqState] = {}
    batch_notify_size = self.server_args.batch_notify_size # 缓存配置，避免循环内属性链查找
    for i, rid in enumerate(recv_obj.rids):
        state = self.rid_to_state.get(rid, None)
        if state is None:
            logger.error(...)
            continue
        # ... 构建 meta_info 等逻辑不变 ...
        # 不立即 event.set()，而是暂存
        pending_notify[rid] = state
        # 每积累 batch_notify_size 个通知，统一唤醒并 yield
        if len(pending_notify) >= batch_notify_size:
            for s in pending_notify.values():
                s.event.set()
            pending_notify = {}
            await asyncio.sleep(0) # 让出事件循环，允许其他协程处理已唤醒的流
        # ... 后续日志 /metrics 不变 ...
    # 循环结束，flush 剩余通知（handle_loop 会立即 await recv，无需额外 sleep）
    for s in pending_notify.values():
        s.event.set()
    # handle_loop awaits next recv immediately
```

评论区精华

> alexnails：建议使用 msgspec struct 并添加 omit_defaults=True 减少 35% payload，移除 tool_calls 字段，类型化 usage 字段。作者已采纳。 alexnails：硬编码 16 应改为可配置参数。作者添加 --batch-notify-size。 alexnails：缓存 batch_notify_size 避免循环内属性链查找。作者已修改。 alexnails：统一 SSE chunk 构造路径。作者已将首 chunk 也通过 _fast_sse_content 生成。

- 使用 msgspec struct 替代 Pydantic 序列化 (design): 采用 msgspec.Struct 并添加 _fast_sse_content 辅助函数。
- batch_notify_size 硬编码为 16，建议可配置 (design): 添加 --batch-notify-size CLI 参数，默认 16。
- 移除 _StreamDelta 中无用的 tool_calls 字段 (performance): 作者同意并移除该字段。
- 使用 omit_defaults=True 减小 SSE payload (performance): 添加 omit_defaults=True，同时保留首个 chunk 的 content="" 以匹配 OpenAI 预期。
- 类型化 usage 字段避免 Any 和编码回退 (performance): 采用 Option1，调用侧先 .model_dump()，usage 类型改为 Optional[dict]。
- 缓存 batch_notify_size 避免循环内属性链查找 (performance): 在循环前缓存 batch_notify_size = self.server_args.batch_notify_size。
- 最终 flush 后是否需要 asyncio.sleep(0) (correctness): 添加注释 '# handle_loop awaits next recv immediately'。
- 统一 SSE chunk 构造路径，将首 chunk 也通过 _fast_sse_content 生成 (design): 三个路径已合并使用 _fast_sse_content。

风险与影响

- 风险： 1) 核心路径变更：_handle_batch_output 改为 async，但通过 asyncio.sleep(0) 保证公平调度，已通过 CI。 2) 缺少测试覆盖：无新增单元测试，依赖现有 CI。 3) 输出格式兼容性：omit_defaults=True 可能改变 wire format，但首个 chunk 的 content="" 已保留以匹配 OpenAI SDK 预期。 4) IPC 序列化（pickle）仍为瓶颈，但 PR #21643 已在迁移中。
  - 影响：直接影响所有使用 PD disaggregation streaming 的部署，特别是高并发场景（如 2048 并发）。用户将体验到更低的 TPOT 和更高的吞吐量。对非 PD streaming 场景无影响。团队需注意新参数 batch-notify-size 的调优。
  - 风险标记：核心路径变更，缺少测试覆盖，输出格式兼容性

关联脉络

- PR #21643 IPC 序列化迁移（pickle→msgpack/msgspec）：PR body 提及该 PR 正在进行 IPC 序列化迁移，与本 PR 的 SSE 序列化优化形成互补，共同解决高并发 PD streaming 的 CPU 瓶颈。