PR #32588 完整报告

sgl-project/sglang

feat(grpc): add generation request semantics

合并时间：2026-08-05 09:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32588

执行摘要

- 一句话：gRPC 增加生成控制，并重构 n>1 并行采样 RID 生命周期
- 推荐动作：值得精读两个层面：一是 request_utils.rs 中 proto → GenerateReqInput 的映射与校验（特别是 choices 转义、unset/false 区分、legacy 兼容），这是 gRPC 桥接层值得复用的范式；二是 merrymercy 的 post-merge review 与 #34160 回滚，作为 " 过度工程化 " 的反面教材——面对边缘场景应先评估调度器已有能力（如 startswith RID 匹配）。不建议在新代码中沿用本 PR 的 lifecycle 机制。

功能与动机

PR body 明确指出：原生 gRPC generation 目前无法表达确定性随机种子、请求优先级、推理控制、思考预算以及 SGLang 支持的全套 guided-decoding 约束；同时并行采样会把调用方可见的 request ID 展开成占位 ID，导致 n > 1 时父请求取消、清理和已完成 RID 复用不可靠。因此需要把 typed-generation 草案 #32121 中与请求语义和生命周期相关的水平切片先落地，为 Dynamo sidecar 等 gRPC 调用方提供与 HTTP 对齐的控制能力。

实现拆解

实现按以下步骤拆解：

1. proto 契约扩展（proto/sglang/runtime/v1/sglang.proto） - SamplingParams 新增 seed = 16 与 guided_decoding = 17（oneof：json_schema / regex / ebnf / choice / structural_tag），原 json_schema = 14、regex = 15 标记为 deprecated 兼容字段。 - TextGenerateRequest 与 GenerateRequest 均新增 priority、require_reasoning、max_thinking_tokens 请求级控制字段（review 中 alexnails 要求从 SamplingParams 移到请求层）。 - 新增 ChoiceConstraint 消息承载 choices 列表。
2. Rust 侧映射与校验（rust/sglang-grpc/src/utils/request_utils.rs、src/server.rs） - sampling_params_to_map 改为返回 Result：guided_decoding 与 legacy json_schema/regex 互斥报错；空约束拒绝；choices 通过 regex_escape_literal 转义后拼成 (?:a|b|c) 正则。 - 新增 insert_generation_controls，仅在字段为 Some 时写入 priority / require_reasoning / max_thinking_tokens，避免破坏 unset 与显式 false 的区分。 - server.rs 将映射错误经 map_err(Status::invalid_argument) 转为 gRPC INVALID_ARGUMENT。

3. Python 请求结构改造（python/sglang/srt/managers/io_struct.py） - GenerateReqInput 新增 max_thinking_tokens 字段，__getitem__ 传递 require_reasoning / max_thinking_tokens / priority。 - _normalize_rid 改为每个原始 batch 项一个逻辑 RID，不再按 parallel_sample_num 展开；regenerate_rid(prefix) 支持生成带父逻辑 RID 前缀的子 RID，__getitem__ 用 i % batch_size 取逻辑 RID。

4TokenizerManage生命周期机制（python/sglang/srt/managers/tokenizer_manager.py） - 新增 RequestAbortedError(status_code=499)，ReqState 增加 abort_requested、lifecycle_id、dispatched 字段。 - 引入 logical_rid_to_child_rids 与 child_rid_to_logical_rid 两张映射表，配合 _init_child_req_state、_register_child_rid、_remove_req_state 管理父子关系。 - _init_req_state 先做全量 duplicate 预检再插入，避免批次失败留下半截状态；_discard_pending_req_states 事务化清理，仅对已 dispatched 的 RID 向调度器发 abort。 - abort_request 对逻辑 RID 做 abort 扇出到所有活跃子 RID；_stream_batch_responses / _collect_batch_responses 在某个 choice 失败时取消并关闭兄弟生成器任务。 - generate_request 在多个 await 点插入 _raise_if_logical_request_aborted / _raise_if_logical_rid_aborted，异常路径改为捕获 BaseException 并携带 lifecycle 信息清理。 - max_thinking_tokens 在未开启 --enable-strict-thinking 时直接报错，并在 _create_tokenized_object 中写入 custom_params["thinking_budget"]。

5. gRPC bridge 与配套（python/sglang/srt/entrypoints/grpc_bridge.py、http_server.py） - _run_generate 流式模式按 completed choice 集合判断整批是否结束（先完成的一个 choice 不再是 batch 终止信号），非流式模式逐个 choice 发送，finally 中统一 gen.aclose()。 - http_server 的错误响应改用 getattr(e, "status_code", 400)，使 RequestAbortedError 可以以 499 返回。

6. 测试配套 - 新增 test/registered/unit/entrypoints/test_grpc_bridge.py，覆盖非流式全部 choice 返回、流式首个 choice 完成不终止 batch。 - 扩展 test_tokenizer_manager_rid_cleanup.py（重新启用此前被 skip 的文件）：批次 duplicate 预检不插半截状态、并行清理 abort 子节点并允许父 RID 复用、stale cleanup 不误删复用 RID、兄弟任务取消关闭。 - test_io_struct.py 新增并行采样保留单逻辑 RID 与 regenerate_rid(prefix) 前缀测试。 - Rust 侧新增 generation controls 保留、choices 转义、非法组合拒绝等单测。

关键文件：

- python/sglang/srt/managers/tokenizer_manager.py（模块 请求管理；类别 source；类型 core-logic；符号 RequestAbortedError, _collect_batch_responses, _stream_batch_responses, _logical_rids）：核心请求路径：引入逻辑 / 子 RID 两张映射表、abort 扇出、事务化清理与多次 abort gate，是本次改动中风险最高、争议最大的文件。
- rust/sglang-grpc/src/utils/request_utils.rs（模块 gRPC 映射；类别 source；类型 core-logic；符号 regex_escape_literal, sampling_params_to_map, insert_generation_controls, generate_dicts_preserve_optional_generation_controls）：gRPC 请求语义的落地点：guided decoding 契约映射、choices 转义正则、generation controls 注入与非法组合拒绝。

- test/registered/unit/managers/test_tokenizer_manager_rid_cleanup.py（模块 生命周期测试；类别 test；类型 test-coverage；符号 _make_req_state, test_batch_duplicate_preflight_does_not_insert_partial_state, test_discard_single, test_discard_single_aborts_scheduler_before_cleanup）：重新启用被 skip 的测试文件并承载生命周期回归覆盖：并行 abort 扇出、stale cleanup、RID 复用、事务化清理。
- python/sglang/srt/managers/io_struct.py（模块 请求结构；类别 source；类型 core-logic；符号 regenerate_rid, new_rid, _normalize_rid）：RID 语义核心变更：逻辑 RID 与子 RID 分离，_normalize_rid 与 regenerate_rid 的行为直接影响并行采样。
- test/registered/unit/entrypoints/test_grpc_bridge.py（模块 桥接测试；类别 test；类型 test-coverage；符号 _ChunkStatus, _RecordingCallback, init, call）：新增测试文件，覆盖 gRPC bridge 在并行采样下流式 / 非流式 response 的完成语义。
- python/sglang/srt/entrypoints/grpc_bridge.py（模块 桥接层；类别 source；类型 core-logic）：gRPC 响应语义收口：流式按 choice 集合判定整体完成，非流式逐个 choice 发送并统一关闭生成器。
- test/registered/unit/managers/test_io_struct.py（模块 结构测试；类别 test；类型 test-coverage；符号 test_parallel_sampling_keeps_one_logical_rid_per_prompt, test_regenerate_rid_with_parent_prefix）：为逻辑 RID 保留与子 RID 前缀生成补充单元测试。
- python/sglang/srt/entrypoints/http_server.py（模块 服务入口；类别 source；类型 core-logic）：HTTP 错误响应透传 status_code，使 RequestAbortedError 以 499 返回。
- rust/sglang-grpc/src/server.rs（模块 gRPC 服务；类别 source；类型 core-logic）：把请求构建函数的 Result 错误映射为 gRPC INVALID_ARGUMENT。
- proto/sglang/runtime/v1/sglang.proto（模块 接口定义；类别 other；类型 core-logic）：gRPC 契约定义：新增 generation controls 与 guided decoding oneof，旧字段标记 deprecated。

关键符号：sampling_params_to_map, regex_escape_literal, insert_generation_controls, _normalize_rid, regenerate_rid, abort_request, _init_req_state, _init_child_req_state, _register_child_rid, _remove_req_state, _raise_if_logical_rid_aborted, _discard_pending_req_states, _stream_batch_responses, _collect_batch_responses, _run_generate

关键源码片段

python/sglang/srt/managers/io_struct.py

RID 语义核心变更：逻辑 RID 与子 RID 分离，_normalize_rid 与 regenerate_rid 的行为直接影响并行采样。

```
def regenerate_rid(self, prefix: Optional[str] = None):
    """生成新的请求 ID（子 RID），可选携带父逻辑 RID 前缀。"""

    def new_rid() -> str:
        # 子 RID 形如 {logical_rid}_{uuid}，便于审计时通过 startswith 反查父请求；
        # 无前缀时退化为纯 uuid
```

```
            suffix = uuid.uuid4().hex
            return f"{prefix}_{suffix}" if prefix is not None else suffix

        if isinstance(self.rid, list):
            self.rid = [new_rid() for _ in range(len(self.rid))]
        else:
            self.rid = new_rid()
        return self.rid


    def _normalize_rid(self):
        """每个原始 batch 项只保留一个逻辑 RID，并行采样子 RID 由 TokenizerManager 内部生成。"""
        if self.rid is None:
            # 只按 batch_size 生成，不再按 parallel_sample_num 展开后的数量生成
            self.rid = [uuid.uuid4().hex for _ in range(self.batch_size)]
        elif isinstance(self.rid, str):
            # 单请求保持原 RID；多 batch 用 {rid}_{i} 前缀展开
            if self.batch_size == 1:
                self.rid = [self.rid]
            else:
                self.rid = [f"{self.rid}_{i}" for i in range(self.batch_size)]
        elif isinstance(self.rid, list):
            # 用户显式指定的 RID 数量必须等于 batch_size，而非并行展开后的数量
            if len(self.rid) != self.batch_size:
                raise ValueError(
                    "The specified rids length mismatch with the batch_size for batch processing."
                )
        else:
            raise ValueError("The rid should be a string or a list of strings.")
```

python/sglang/srt/entrypoints/grpc_bridge.py

gRPC 响应语义收口：流式按 choice 集合判定整体完成，非流式逐个 choice 发送并统一关闭生成器。

```
    async def _run_generate(self, obj, chunk_callback, stream: bool, request):
        ready_event = None
        gen = None
        try:
            ready_event = self._install_on_ready(chunk_callback)
            gen = self.tokenizer_manager.generate_request(obj, request=request)
            if stream:
                completed_choices = set()
                expected_choices = obj.batch_size * obj.parallel_sample_num
                async for chunk in gen:
                    choice_finished = (
                        chunk.get("meta_info", {}).get("finish_reason") is not None
                    )
                    if choice_finished:
                        # 用 index 或 choice id 去重；先完成的一个 choice 不是整批结束
```

```
                    choice_id = chunk.get(
                        "index", chunk.get("meta_info", {}).get("id")
                    )
                    completed_choices.add(choice_id)
                finished = len(completed_choices) >= expected_choices
                keep_going = await self._send_with_backpressure(
                    chunk_callback,
                    ready_event,
                    chunk,
                    finished=finished,
                    timeout_abort_rid=obj.rid,
                )
                if finished or not keep_going:
                    return
            # 防御：生成器未带 finish_reason 就退出时补一个完成帧
            self._safe_callback(chunk_callback, {}, finished=True)
        else:
            result = await gen.__anext__()
            # 非流式可能一次返回多个 choice，逐个发并只在最后一个标记 finished
            chunks = result if isinstance(result, list) else [result]
            for index, chunk in enumerate(chunks):
                keep_going = await self._send_with_backpressure(
                    chunk_callback,
                    ready_event,
                    chunk,
                    finished=index == len(chunks) - 1,
                    timeout_abort_rid=obj.rid,
                )
                if not keep_going:
                    return
    except StopAsyncIteration:
        self._safe_callback(chunk_callback, {}, finished=True)
    except Exception as e:
        logger.error("gRPC generate error for rid=%s: %s", obj.rid, e)
        self._send_native_error(chunk_callback, str(e))
    finally:
        # 无论成功失败都关闭生成器，确保兄弟任务被取消清理
        if gen is not None:
            await gen.aclose()
        self._uninstall_on_ready(chunk_callback)
```

评论区精华

核心讨论如下：

> proto 字段归属（alexnails）：require_reasoning、max_thinking_tokens 不应放在 SamplingParams，应移到请求层；priority 不能只加在 tokenized GenerateRequest，TextGenerateRequest 也要有。——作者已分别修复，并将旧字段保留为兼容字段。

unset 与显式 false 的区分（Claude review 转述）：unwrap_or(false) 会在每次请求里无条件插入 require_reasoning，破坏未设置与显式 false 的语义。——作者改为仅在 Some 时插入。

整体设计质疑（merrymercy）："overall i feel this pr adds too many code for an edge case. do you have a simpler method?"，并提出基于随机内部 group prefix + 单次 AbortReq(group_prefix)（调度器已支持 startswith 匹配）的替代方案，可去掉两张映射表、lifecycle_id、per-child dispatched 与多处 abort gate。作者未在合并前回应，merrymercy 在合入后给出 post-merge review，并开出 #34160 做部分回滚。

AbortReq 批量语义（alexnails）："let AbortReq carry a list of rids rather than N round trips?"——作者认为 AbortReq 是单向 IPC、改动范围大，建议作为 follow-up。

跳过测试重新启用（alexnails）：对重新启用 test_tokenizer_manager_rid_cleanup.py 有疑虑——作者说明原跳过原因已随 ServerArgs 迁移完成消失，且该文件承载了本 PR 的新增生命周期回归覆盖。

生成器关闭顺序（alexnails）：要求确认 cancel 与 generator close 的时序——作者确认先 cancel 并 await 所有任务，再统一 aclose()，并配有流式 / 非流式两个回归测试。

- proto 字段归属：require_reasoning/max_thinking_tokens 应放在请求层而非 SamplingParams (design): 作者将三个字段统一移到 TextGenerateRequest 与 GenerateRequest 请求层，SamplingParams 只保留 seed 与 guided_decoding。
- guided decoding 互斥校验与 legacy 兼容 (correctness): 校验已落地，非法组合返回 INVALID_ARGUMENT。
- 生命周期机制过度设计与更简单方案 (design): #34160 已回滚逻辑 / 子 RID 生命周期机械，保留 gRPC 请求控制与 sibling-task cleanup。
- 多次调用 _raise_if_logical_request_aborted 的必要性 (design): 该问题随 #34160 回滚大部分生命周期调用点而缓解，但请求控制部分仍保留关键检查。
- AbortReq 是否应携带 rid 列表 (performance): 保持现状，作为后续优化方向。
- 生成器取消与关闭顺序 (correctness): 作者确认先 cancel 并 await 全部任务，再统一 aclose，并新增流式 / 非流式回归测试。
- 重新启用被 skip 的 rid cleanup 测试文件 (testing): 测试文件重新启用并通过。
- unwrap_or(false) 破坏 unset 与显式 false 区分 (correctness): 改为仅当 Some 时插入，保留 GenerateReqInput 默认值。

风险与影响

- 风险：主要风险集中在生命周期改造与契约变更：

1. TokenizerManager 生命周期复杂度风险（高）：logical_rid_to_child_rids / child_rid_to_logical_rid / lifecycle_id / dispatched 等多重状态叠加在核心请求路径上，任何遗漏都会造成状态泄漏、重复 abort 或误删复用 RID。merrymercy 明确认为这是过度设计，且最终 #34160 已回滚大部分机械（保留请求控制与 sibling 清理），说明该设计在团队内部未获认可，存在维护负担。

2. RID 语义行为变更：_normalize_rid 从 " 按并行展开数量生成 RID" 改为 " 按原始 batch 项生成逻辑 RID"，__getitem__ 引入 i % batch_size。作者称内部调用方不受影响，但任何依赖旧展开 RID 行为的代码（例如直接构造 GenerateReqInput 并读取 rid 列表的调用方）都可能受到破坏。
3. proto 兼容性：json_schema = 14、regex = 15 保留但标记 deprecated；若新客户端同时设置 legacy 与新 guided_decoding 会被 INVALID_ARGUMENT 拒绝，空字符串 legacy 字段也会从静默接受变为报错，属于收紧行为。
4. 流式完成判定变化：grpc_bridge 现在以 completed choice 集合判定 batch 完成，依赖 index / meta_info.id 标识；若上游 response 缺少稳定 choice 标识，可能导致完成判断错误或提前终止。
5. HTTP 错误码：错误响应开始透传 e.status_code（如 499），可能改变 API 层错误契约，需要确认下游对 499 的兼容。 - 影响：影响面：
   - 原生 gRPC 用户：获得确定性 seed、优先级、推理控制、思考预算与完整 guided decoding 能力，与 HTTP 入口对齐；n > 1 并行采样的取消与 RID 复用行为变得可靠。
   - SGLang 内核：TokenizerManager 是请求必经路径，本次改动把并行采样生命周期从隐式展开改为显式父子管理，属于核心请求路径变更，影响所有使用 parallel_sample_num 的请求（HTTP 与 gRPC 共用）。
   - 性能：AIPerf A/B 锁定的 H100 NVL 测试中输出吞吐仅下降 0.72%（3,243.74 → 3,220.44 tok/s），高于 95% 接受线；inter-chunk p99 +1.84%，无重启。
   - 团队与后续演进：该 PR 是 typed-generation 草案 #32121 的水平切片，最终被 #34160 部分回滚，留下 " 先做复杂生命周期、再简化为 group prefix" 的架构教训，影响后续请求生命周期相关设计。
   - 风险标记：核心路径变更，生命周期逻辑被部分回滚，proto 契约变更，n>1 RID 语义变化，错误码语义变化

关联脉络

- PR #32121 typed-generation draft (umbrella): 本 PR 是 PR body 所述从其抽取的第一个水平切片，覆盖 generation request semantics 与生命周期，response typing 与 bridge metadata 留在 umbrella PR 中。
- PR #34160 Partial revert of #32588 (simplify n>1 RID lifecycle): merrymercy post-merge review 后开出的小型回滚：移除 GenerateReqInput/TokenizerManager 中的逻辑 / 子 RID 生命周期机械，保留独立的 gRPC 请求控制与 sibling-task cleanup。