PR #26378 完整报告

sgl-project/sglang

bench_serving: add Zipfian shared-prefix sampling to generated-shared-prefix

合并时间：2026-05-29 05:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26378

执行摘要

- 一句话：为 GSP 数据集增加 Zipf 前缀分布采样
- 推荐动作：该 PR 设计干净，值得阅读：1）_zipf_group_probs 数学实现简洁；2）CLI 校验前移，减少用户等待；3）RNG 隔离保证可复现性；4）缓存键细分，避免不同分布互相污染；5）测试覆盖全面，包括数学验证和子进程 CLI 测试。

功能与动机

真实缓存导向的负载测试通常需要热头 / 长尾前缀分布（hot-head / long-tail prefix distribution），而现有的 generated-shared-prefix 均匀分配每个组相同数量的请求，无法模拟这种偏斜。PR body 明确提到：'real cache-oriented serving studies often need a hot-head / long-tail prefix distribution'。

实现拆解

1. 新增 Zipf 概率计算函数_zipf_group_probs（generated_shared_prefix.py），计算 rank-based 概率向量。
2. 扩展 Dataset 类：GeneratedSharedPrefixDataset 新增 group_distribution 和 zipf_alpha 字段，from_args 增加防御性校验，load 透传新参数。
3. 统一采样循环：修改 sample_generated_shared_prefix_requests，通过预计算的 assignment 数组统一 uniform 和 zipf 路径，Zipf 分支使用隔离的 numpy.random.default_rng(seed) 保证可复现性。
4. CLI 参数与校验：在 bench_serving.py 的 --gsp-* 参数组中注册 --gsp-group-distribution 和 --gsp-zipf-alpha，提供 _finite_positive_float 类型验证器，新增 _validate_parsed_gsp_args 在 parse 阶段提前拒绝非法组合（如 zipf 缺少 alpha 或 uniform 携带 alpha）。
5. 缓存键分离：get_gen_prefix_cache_path 的缓存键加入 group_distribution 和 zipf_alpha，确保不同分布及不同 alpha 使用独立缓存文件，uniform 模式文件名不变向后兼容。
6. 文档更新：docs/developer_guide/bench_serving.md 和 docs_new/docs/developer_guide/bench_serving.mdx 增加新参数说明和 Zipf 示例命令。
7. 测试覆盖：新增多组确定性单元测试，包括数学正确性（最大偏差 8.8e-4）、可复现性、不同 seed 结果不同、缓存隔离、CLI 错误子进程验证等。

关键文件：

- python/sglang/benchmark/datasets/generated_shared_prefix.py（模块 共享前缀；类别 source；类型 core-logic；符号 _zipf_group_probs, GeneratedSharedPrefixDataset, get_gen_prefix_cache_path, sample_generated_shared_prefix_requests）：核心实现，新增 Zipf 概率计算、扩展数据集类、修改采样函数、更新缓存键
- python/sglang/bench_serving.py（模块 Bench CLI；类别 source；类型 dependency-wiring；符号 _finite_positive_float, _validate_parsed_gsp_args）：CLI 入口，注册新参数、添加类型验证器和交叉参数校验函数
- test/registered/bench_fn/test_benchmark_datasets_api.py（模块 测试用例；类别 test；类型 test-coverage；符号 _run_gsp, test_zipf_group_probs_helper, test_zipf_reproducible_with_seed, test_zipf_different_seeds_differ）：大量新增测试，覆盖 Zipf 数学正确性、可复现性、缓存隔离、CLI 错误等
- docs_new/docs/developer_guide/bench_serving.mdx（模块 开发者文档；类别 other；类型 documentation）：文档更新，包含新参数说明和 Zipf 用法示例

关键符号：_zipf_group_probs, _finite_positive_float, _validate_parsed_gsp_args, GeneratedSharedPrefixDataset.from_args, GeneratedSharedPrefixDataset.load, sample_generated_shared_prefix_requests, get_gen_prefix_cache_path

关键源码片段

python/sglang/benchmark/datasets/generated_shared_prefix.py

核心实现，新增 Zipf 概率计算、扩展数据集类、修改采样函数、更新缓存键

```
# python/sglang/benchmark/datasets/generated_shared_prefix.py

def _zipf_group_probs(num_groups: int, alpha: float) -> np.ndarray:
    """Rank-based Zipf 概率向量，rank 从 1 开始。

    权重(rank)      = 1 / rank ** alpha       (rank = 1..num_groups)
    概率(rank) = 权重(rank) / 所有权重之和

    返回数组长度为 num_groups，索引 i 对应 rank i+1，即 group 0 最热。
    """
    if num_groups <= 0:
        raise ValueError(f"num_groups must be > 0, got {num_groups}")
    ranks = np.arange(1, num_groups + 1, dtype=np.float64)
    weights = 1.0 / (ranks ** alpha)
    return weights / weights.sum()


@dataclass
class GeneratedSharedPrefixDataset(BaseDataset):
    # ... 其他字段 ...
    group_distribution: str = "uniform"
    zipf_alpha: Optional[float] = None
```

```
    @classmethod
    def from_args(cls, args: Namespace) -> "GeneratedSharedPrefixDataset":
        assert not getattr(args, "tokenize_prompt", False)
        group_distribution = args.gsp_group_distribution
        zipf_alpha = args.gsp_zipf_alpha

        # 防御性校验：防止跳过 argparse 的手动调用
        if group_distribution not in ("uniform", "zipf"):
            raise ValueError(
                f"--gsp-group-distribution must be 'uniform' or 'zipf', "
                f"got {group_distribution!r}"
            )
        if group_distribution == "zipf":
            if zipf_alpha is None:
                raise ValueError(
                    "--gsp-group-distribution=zipf requires --gsp-zipf-alpha "
                    "(a finite float > 0)"
                )
            if not math.isfinite(zipf_alpha) or zipf_alpha <= 0:
                raise ValueError(
                    f"--gsp-zipf-alpha must be a finite float > 0, got {zipf_alpha!r}"
                )
        elif zipf_alpha is not None:
            raise ValueError(
                "--gsp-zipf-alpha is only meaningful with "
                "--gsp-group-distribution=zipf; remove --gsp-zipf-alpha "
                "or set --gsp-group-distribution=zipf"
            )

        return cls(
            num_groups=args.gsp_num_groups,
            prompts_per_group=args.gsp_prompts_per_group,
            # ... 其他字段 ...
            group_distribution=group_distribution,
            zipf_alpha=zipf_alpha,
        )
```

python/sglang/bench_serving.py

CLI 入口，注册新参数、添加类型验证器和交叉参数校验函数

```
# python/sglang/bench_serving.py

def _finite_positive_float(value) -> float:
    """argparse 类型：有限且严格的正浮点数。"""
    try:
        parsed = float(value)
    except (TypeError, ValueError) as exc:
        raise argparse.ArgumentTypeError(
            f"expected a finite float > 0, got {value!r}"
```

```
        ) from exc
    if not math.isfinite(parsed) or parsed <= 0:
        raise argparse.ArgumentTypeError(f"expected a finite float > 0, got {value!r}")
    return parsed


def _validate_parsed_gsp_args(
    parser: argparse.ArgumentParser, args: argparse.Namespace
) -> None:
    """在 parse 阶段拒绝非法的 GSP 分布/alpha 组合。

    在 parser.parse_args() 之后立即调用，使用户在 server/model/tokenizer
    设置之前就看到清晰的 argparse 错误。
    """
    distribution = getattr(args, "gsp_group_distribution", None)
    alpha = getattr(args, "gsp_zipf_alpha", None)
    if distribution == "zipf" and alpha is None:
        parser.error(
            "--gsp-group-distribution=zipf requires --gsp-zipf-alpha "
            "(a finite float > 0)"
        )
    if distribution == "uniform" and alpha is not None:
        parser.error(
            "--gsp-zipf-alpha is only meaningful with "
            "--gsp-group-distribution=zipf; remove --gsp-zipf-alpha "
            "or set --gsp-group-distribution=zipf"
        )


if __name__ == "__main__":
    parser = ArgumentParser(...)
    # ... 其他参数 ...
    group = parser.add_argument_group("Generated Shared Prefix flags")
    group.add_argument(
        "--gsp-group-distribution",
        type=str,
        choices=["uniform", "zipf"],
        default="uniform",
        help=(
            "Prefix-group sampling distribution for generated-shared-prefix. "
            "'uniform' (default) assigns each group an equal number of requests. "
            "'zipf' samples each request's group by rank with "
            "p(rank) = (1/rank**alpha) / sum_k(1/k**alpha); rank starts at 1 "
            "and group index 0 is the hottest. Requires --gsp-zipf-alpha "
            "(a finite float > 0) when set to 'zipf'. Total request count is "
            "still num_groups * prompts_per_group, identical to uniform mode; "
            "only the per-request group assignment changes. The on-disk "
            "dataset cache uses a distinct key per (group_distribution, "
            "zipf_alpha), so uniform-mode caches are never mixed with "
```

```
            "zipf-mode caches and zipf runs with different alpha use "
            "separate files."
        ),
    )
    group.add_argument(
        "--gsp-zipf-alpha",
        type=_finite_positive_float,
        default=None,
        help=(
            "Zipf exponent alpha for --gsp-group-distribution=zipf, with "
            "p(rank) = (1/rank**alpha) / sum_k(1/k**alpha) and rank starting "
            "at 1. Must be a finite float strictly greater than 0; larger "
            "values concentrate requests on lower-ranked (hotter) groups. "
            "Required when distribution is 'zipf'; must be omitted otherwise."
        ),
    )
```

test/registered/bench_fn/test_benchmark_datasets_api.py

大量新增测试，覆盖 Zipf 数学正确性、可复现性、缓存隔离、CLI 错误等

```
# test/registered/bench_fn/test_benchmark_datasets_api.py ( 部分 )

    def _run_gsp(self, *, mode="uniform", alpha=None, seed=42, num_groups=4,
                 prompts_per_group=5, **kwargs):
        """辅助方法：封装 GSP 数据集生成，统一测试入口。"""
        # 注意：GSP 的 seed 仅影响缓存文件名；compute_random_lens 的
        # 可复现性需要先 seed 全局 random 和 numpy。
        random.seed(seed)
        np.random.seed(seed)

        ds = GeneratedSharedPrefixDataset(
            num_groups=num_groups,
            prompts_per_group=prompts_per_group,
            # ... 其他参数 ...
            group_distribution=mode,
            zipf_alpha=alpha,
        )
        rows = ds.load(self.tokenizer)
        # 验证总行数
        self.assertEqual(len(rows), num_groups * prompts_per_group)
        return rows

    def test_zipf_group_probs_helper(self):
        """验证 _zipf_group_probs 的数学正确性。"""
        probs = _zipf_group_probs(5, 1.0)
        # 理论概率：sum_{k=1}^5 1/k = 2.28333...
        # rank 1: 1/1 / sum = 0.4379...
        self.assertAlmostEqual(probs[0], 1.0 / 2.28333, places=4)
        self.assertAlmostEqual(probs.sum(), 1.0)
```

```
        # 单调递减
        for i in range(len(probs) - 1):
            self.assertGreater(probs[i], probs[i + 1])

    def test_zipf_reproducible_with_seed(self):
        """相同 seed 保证可复现。"""
        rows1 = self._run_gsp(mode="zipf", alpha=1.0, seed=42)
        rows2 = self._run_gsp(mode="zipf", alpha=1.0, seed=42)
        for r1, r2 in zip(rows1, rows2):
            self.assertEqual(r1, r2)
```

评论区精华

Review 过程无实质争论，两位 reviewer（alexnails 和 zijiexia）均直接 approve。唯一的评论来自 gemini-code-assist[bot] 的自动 Code Review 总结，未引发任何修改。

- 暂无高价值评论线程

风险与影响

- 风险：影响范围限于基准测试模块，不涉及模型推理核心路径。主要风险：1）缓存键变更可能导致旧缓存被错误复用？但 uniform 模式缓存路径未变，新格式向后兼容；2）Zipf RNG 隔离确保不干扰现有 random/numpy 全局状态；3）CLI 参数早期校验避免因错误参数触发不必要的 server 初始化；4）测试 CI 时间从 6s 增加到 40s，但仅在 base-a-test-cpu suite 中，影响可控。
- 影响：仅影响使用 --dataset-name generated-shared-prefix 的用户，新增两个可选参数，默认行为完全一致。现有脚本和缓存文件无需修改。测试部分 CI 运行时间从 6s 增加到 40s（因大量新测试），但仅在 base-a-test-cpu suite 中。
- 风险标记：非核心路径，缓存键兼容性需验证，CI 时间增加

关联脉络

- 暂无明显关联 PR