

PR #48866 完整报告

vllm-project/vllm

[Metrics] Consolidate Prometheus histogram bucket defaults into a single module

合并时间: 2026-09-11 22:12

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48866>

执行摘要

- 一句话: Prometheus 直方图桶默认值收敛到单一模块, 零行为变更
- 推荐动作: 值得精读, 但目标读者是 metrics/observability 维护者和准备做桶配置化的开发者, 普通模型开发者可以只看结论。建议重点看三处设计决策: 一是用 Literal + get_args 派生 BUCKET_FAMILY_KEYS, 让「家族集合」在类型层和运行时各有一份但同源, 便于测试做集合等值断言; 二是快照测试的双层策略 (访问器级 + 通过 PrometheusStatLogger 与 Prometheus 注册表的端到端回读), 其中第二层捕捉到的正是「main 合进来一个新指标」这类跨分支漂移, 非常值得在其他「单一数据源」重构中复用; 三是 histogram_buckets 明确返回新副本的契约, 为将来「默认值 + 用户覆盖」的合并逻辑留出了安全边界。审阅时建议继续追问两点: 五个指标共享同一 list 实例是否应改为各自取副本, 以及未知家族键是否应统一抛 ValueError——这两点都会在 #48867 落地时被放大。

功能与动机

issue #34370 的核心诉求是: 直方图桶目前静态写死在源码里, 「在高吞吐、低延迟的生产环境中, 运维方需要更细的亚 300 ms 粒度、随负载画像自定义的桶分布、与既有 SLO 对齐, 以及在多个服务间保持一致的桶定义», 而改桶只能打补丁并重新编译 vLLM。markmc 在该 issue 的评论中给出了分步落地方案, 其中「PR 1 — Consolidate Bucket Types」正是本 PR 要实现的步骤。改造前的直接痛点是: vllm/v1/metrics/loggers.py 里 request_latency 的 21 个边界在 e2e、queue、inference、prefill、decode 五个指标上逐字重复, inter_token_latency 列表也在两个指标上重复; 想做配置化就必须先有单一数据源, 否则配置项无处挂载。作者也刻意划清了与既有 PR 的边界: #35952 提出同一合并方案但自 5 月起停摆且缺少测试计划; #48175 改动 request_latency 默认值而本 PR 不改任何值; #34564/#34720 走的是「每个指标一个 CLI flag」的路线, 已被 changes requested 否决。

实现拆解

1. 新增单一数据源模块 `vllm/v1/metrics/buckets.py` (+228)

- 用 BucketFamilyKey = Literal[...] 声明八个家族键, 再用 BUCKET_FAMILY_KEYS: frozenset[str] = frozenset(get_args(BucketFamilyKey)) 派生出可做成员检查的集合, 使「有哪些家族」只有一处定义, 测试可直接对其做等值断言。
- 七个静态家族以模块级 tuple[float, ...] 常量承载 (REQUEST_LATENCY_BUCKETS、TIME_TO_FIRST_TOKEN_BUCKETS、INTER_TOKEN_LATENCY_BUCKETS、

ITERATION_TOKENS_BUCKETS、REQUEST_PARAMS_N_BUCKETS、REQUEST_NUM_PREEMPTIONS_BUCKETS、KV_CACHE_RESIDENCY_BUCKETS)，每个常量配一段 docstring 说明数值范围与用途，这是 issue 计划里明确要求的「per-familyrationale」。这些常量不进入 _STATIC_FAMILY_DEFAULTS 字典的那个家族是 request_tokens，因为它的边界随 max_model_len 动态生成。

- build_buckets() 与 build_1_2_5_buckets() 从 loggers.py 迁移进来，并在迁移时把返回类型从 list[int] 修正为 list[float]——原先 [1, 2, 5] 的整型推导与 10**exponent 产生的浮点值混用，类型标注是不准确的。
- histogram_buckets(family, max_model_len=None) 是唯一访问器：request_tokens 走 build_1_2_5_buckets(max_model_len) 且缺参时抛 ValueError，其余家族返回 list(_STATIC_FAMILY_DEFAULTS[family]) 的新副本，保证调用方可以自由修改返回值而不污染模块级常量。

2. 改造 vllm/v1/metrics/loggers.py (+23/-147)

- 导入从 build_1_2_5_buckets 改为 histogram_buckets，并在 PrometheusStatLogger.__init__ 中、创建任何直方图之前集中解析七个局部列表变量，随后各指标直接引用变量，取代原先散落的内联字面量与重复的 build_1_2_5_buckets(max_model_len) 调用。
- request_tokens_buckets 一次解析后被五个 token 计数类指标共享（prompt tokens、generation tokens、max_num_generation_tokens、params_max_tokens、prefill_kv_computed tokens），这正是原先重复最严重的部分。
- kv_cache_residency_buckets 仍留在 if self.kv_cache_metrics_enabled: 分支内按需取，避免为未启用的指标做无谓计算。
- 文件末尾重复定义的 build_buckets / build_1_2_5_buckets 被删除，loggers.py 的职责收敛为「注册并更新指标」，不再持有边界知识。

指标名与家族键的映射关系（由测试中的 METRIC_FAMILIES 固化）：

家族键	覆盖的 vllm:* 指标
request_latency	e2e / queue / inference / prefill / decode 五个请求阶段耗时
time_to_first_token	time_to_first_token_seconds
inter_token_latency	inter_token_latency_seconds、request_time_per_output_token_seconds
iteration_tokens	iteration_tokens_total
request_params_n	request_params_n
request_num_preemptions	request_num_preemptions
request_tokens	prompt / generation / max_num_generation / params_max_tokens / prefill_kv_computed tokens

家族键	覆盖的 <code>vllm:*</code> 指标
<code>kv_cache_residency</code>	<code>kv_block_lifetime</code> / <code>idle_before_evict</code> / <code>reuse_gap</code>

3. 新增双层快照测试 `tests/v1/metrics/test_histogram_buckets.py` (+273)

- 第一层是访问器级：`test_default_bucket_snapshots` 参数化遍历 `DEFAULT_BUCKET_SNAPSHOTS`，`test_request_tokens_bucket_snapshot` 校验 `max_model_len=32768` 下的 1-2-5 序列，`test_request_tokens_requires_max_model_len` 校验缺参报错，`test_buckets_return_fresh_copies` 校验返回值是可变的新副本。
- 第二层是端到端：`test_prometheus_logger_default_buckets` 用 `distilbert/distilgpt2` (`max_model_len=1024`) 构造 `VllmConfig + ObservabilityConfig(kv_cache_metrics=True)`，实例化 `PrometheusStatLogger` 后通过 `prometheus_client.REGISTRY.collect()` 按 le 标签回读每个 `vllm:*` 直方图的有限桶边界，与 `METRIC_FAMILIES` 逐项比对；`finally` 中调用 `unregister_vllm_metrics()` 清理进程级全局注册表。
- 期望值刻意写成硬编码字面量而不是 `import` 常量，文件 `docstring` 说明「任何默认桶边界的改动都必须表现为这里的一个失败测试」，这是刻意的变更探测器 (`change detector`) 而不是普通的等价性测试。

4. 演进中的两次修正（无测试 / 配置 / 部署之外的配套改动）

- 合并 `main` 后，`vllm:request_num_preemptions` 这个新指标带着内联边界落地，快照测试立刻以「注册集合与家族映射不一致」失败；commit `6f14d1f` 因此把 `REQUEST_NUM_PREEMPTIONS_BUCKETS` 作为第八个家族补进来，边界保持不变。
- commit `4638a10` 收紧文档：把 `single-source-of-truth` 的说法限定到核心引擎直方图，说明配置了 `KV connector` 时 `KVConnectorProm` 会注册自己的直方图并保有自有边界；同时改写 `PREEMPTION` 家族的 `docstring`，如实描述第一个桶同时容纳未被抢占和被抢占一次的请求。
- 本 PR 不含任何配置 `schema`、CLI flag 或部署变更——这正是它被定义为「纯重构」的前提，配置化留给 #48867。

关键文件：

- `vllm/v1/metrics/buckets.py`（模块 指标桶定义；类别 `source`；类型 `dependency-wiring`；符号 `BucketFamilyKey`, `BUCKET_FAMILY_KEYS`, `REQUEST_LATENCY_BUCKETS`, `TIME_TO_FIRST_TOKEN_BUCKETS`）：新增的单一数据源模块，定义八个桶家族、家族键类型与唯一访问器 `histogram_buckets()`，是整个改动与后续可配置化 PR 的支点。
- `vllm/v1/metrics/loggers.py`（模块 指标导出器；类别 `source`；类型 `core-logic`；符号 `PrometheusStatLogger.init`, `kv_cache_residency_buckets`, `request_tokens_buckets`, `build_buckets`）：移除 147 行内联桶定义与重复的 `build_buckets/build_1_2_5_buckets`，改为一处解析、多处复用，是本次重构的落地面。
- `tests/v1/metrics/test_histogram_buckets.py`（模块 指标测试；类别 `test`；类型 `test-coverage`；符号 `DEFAULT_BUCKET_SNAPSHOTS`, `METRIC_FAMILIES`，

REQUEST_TOKENS_SNAPSHOT_FOR_TEST_MODEL, test_default_bucket_snapshots) :
新增的双层快照测试，把默认边界与指标到家族的映射同时锁死，是防止后续无意改动桶定义的主要护栏。

关键符号：histogram_buckets, build_buckets, build_1_2_5_buckets,
PrometheusStatLogger.init, collect_histogram_buckets,
test_prometheus_logger_default_buckets, test_bucket_family_keys,
test_buckets_return_fresh_copies

关键源码片段

vllm/v1/metrics/buckets.py

新增的单一数据源模块，定义八个桶家族、家族键类型与唯一访问器 histogram_buckets(),
是整个改动与后续可配置化 PR 的支点。

vllm/v1/metrics/buckets.py: 家族声明、静态默认值与唯一访问器

```
# vllm/v1/metrics/buckets.py (新增, 228 行)
from typing import Literal, get_args

# 家族键先用 Literal 声明，再用 get_args() 派生为可做成员检查的 frozenset:
# 「有哪些家族」因此只有一处定义，测试可以直接对 BUCKET_FAMILY_KEYS 做集合等值断言，
# 后续可配置化也能复用它校验用户输入。
BucketFamilyKey = Literal[
    "request_latency",
    "time_to_first_token",
    "inter_token_latency",
    "iteration_tokens",
    "request_params_n",
    "request_num_preemptions",
    "request_tokens",
    "kv_cache_residency",
]
BUCKET_FAMILY_KEYS: frozenset[str] = frozenset(get_args(BucketFamilyKey))

# 静态家族用模块级 tuple 常量承载默认边界，并配 docstring 说明覆盖范围；
# 下面只完整展示 request_latency，其余六个静态家族结构一致、仅数值不同。
# 注意 request_tokens 不在这里：它的边界随 max_model_len 动态生成。
REQUEST_LATENCY_BUCKETS: tuple[float, ...] = (
    0.3, 0.5, 0.8, 1.0, 1.5, 2.0, 2.5,
    5.0, 10.0, 15.0, 20.0, 30.0, 40.0, 50.0,
    60.0, 120.0, 240.0, 480.0, 960.0, 1920.0, 7680.0,
)

"""亚秒级调度延迟到多小时批量请求；被 e2e、queue、inference、prefill、decode
五个请求阶段耗时直方图共享。"""

_STATIC_FAMILY_DEFAULTS: dict[str, tuple[float, ...]] = {
    "request_latency": REQUEST_LATENCY_BUCKETS,
    "time_to_first_token": TIME_TO_FIRST_TOKEN_BUCKETS,
```

```

"inter_token_latency": INTER_TOKEN_LATENCY_BUCKETS,
"iteration_tokens": ITERATION_TOKENS_BUCKETS,
"request_params_n": REQUEST_PARAMS_N_BUCKETS,
"request_num_preemptions": REQUEST_NUM_PREEMPTIONS_BUCKETS,
"kv_cache_residency": KV_CACHE_RESIDENCY_BUCKETS,
}

```

```

def build_buckets(mantissa_lst: list[int], max_value: int) -> list[float]:
    """按 10 的递增幂缩放尾数，直到超出 max_value 为止。"""
    exponent = 0
    buckets: list[float] = []
    while True:
        for m in mantissa_lst:
            value = m * 10**exponent
            if value <= max_value:
                buckets.append(value)
            else:
                # 任一尾数越界即整体返回，保证结果严格递增且不超过上限
                return buckets
        exponent += 1

```

```

def build_1_2_5_buckets(max_value: int) -> list[float]:
    """1-2-5 序列：build_1_2_5_buckets(100) 得到 [1, 2, 5, 10, 20, 50, 100]。"""
    return build_buckets([1, 2, 5], max_value)

```

```

def histogram_buckets(
    family: BucketFamilyKey,
    max_model_len: int | None = None,
) -> list[float]:
    """唯一访问器：返回可被调用方自由修改的桶上界新列表。"""
    if family == "request_tokens":
        if max_model_len is None:
            # request_tokens 是唯一的动态家族，缺参属于调用方错误，显式报错而非静默降级
            raise ValueError(
                "max_model_len is required for the 'request_tokens' bucket family"
            )
        return build_1_2_5_buckets(max_model_len)
    # list(...) 做浅拷贝，避免调用方原地修改污染模块级共享常量
    return list(_STATIC_FAMILY_DEFAULTS[family])

```

[vllm/v1/metrics/loggers.py](#)

移除 147 行内联桶定义与重复的 build_buckets/build_1_2_5_buckets，改为一处解析、多处复用，是本次重构的落地面。

[vllm/v1/metrics/loggers.py](#)：一次解析、多处复用

```

# vllm/v1/metrics/loggers.py (PrometheusStatLogger.__init__ 内)
from vllm.v1.metrics.buckets import histogram_buckets

# 在创建任何直方图之前集中解析边界，取代原先散落在各处的内联列表：
# request_tokens 需要 max_model_len，其余家族忽略该参数。
request_tokens_buckets = histogram_buckets(
    "request_tokens", max_model_len=max_model_len
)
iteration_tokens_buckets = histogram_buckets("iteration_tokens")
request_params_n_buckets = histogram_buckets("request_params_n")
time_to_first_token_buckets = histogram_buckets("time_to_first_token")
inter_token_latency_buckets = histogram_buckets("inter_token_latency")
request_latency_buckets = histogram_buckets("request_latency")
request_num_preemptions_buckets = histogram_buckets("request_num_preemptions")

# 原先这里是重复五次的内联 build_1_2_5_buckets(max_model_len) 调用，
# 现在五个 token 计数指标共享同一个列表对象，这是本次重构最需要留意的形态变化。
histogram_num_prompt_tokens_request = self._histogram_cls(
    name="vllm:request_prompt_tokens",
    documentation="Number of prefill tokens processed.",
    buckets=request_tokens_buckets,
    labelnames=labelnames,
)
self.histogram_num_prompt_tokens_request = create_metric_per_engine(
    histogram_num_prompt_tokens_request, per_engine_labelvalues
)

# 其余指标沿用同一模式：
# vllm:iteration_tokens_total -> iteration_tokens_buckets
# vllm:request_params_n -> request_params_n_buckets
# vllm:time_to_first_token_seconds -> time_to_first_token_buckets
# vllm:inter_token_latency_seconds -> inter_token_latency_buckets
# vllm:request_time_per_output_token_seconds -> inter_token_latency_buckets
# vllm:request_num_preemptions -> request_num_preemptions_buckets
# vllm:e2e_request_latency_seconds 等 5 个 -> request_latency_buckets

# KV 缓存驻留相关指标仍只在开启开关时取桶，避免无谓计算：
if self.kv_cache_metrics_enabled:
    kv_cache_residency_buckets = histogram_buckets("kv_cache_residency")

# 文件末尾原先重复定义的 build_buckets / build_1_2_5_buckets 已删除，
# 迁入 buckets.py 时顺带把返回类型从 list[int] 修正为 list[float]。

```

tests/v1/metrics/test_histogram_buckets.py

新增的双层快照测试，把默认边界与指标到家族的映射同时锁死，是防止后续无意改动桶定义的主要护栏。

tests/v1/metrics/test_histogram_buckets.py: 从 Prometheus 注册表回读的端到端快照

```
# tests/v1/metrics/test_histogram_buckets.py (新增, 273 行)
# 期望值刻意写成硬编码字面量而不是 import 常量:
# 任何默认桶边界的改动都必须表现为这里的一个失败测试。
DEFAULT_BUCKET_SNAPSHOTS: dict[str, list[float]] = {
    "request_latency": [0.3, 0.5, 0.8, ..., 7680.0],
    "time_to_first_token": [0.001, 0.005, ..., 2560.0],
    "inter_token_latency": [0.01, 0.025, ..., 80.0],
    "iteration_tokens": [1, 8, 16, ..., 16384],
    "request_params_n": [1, 2, 5, 10, 20],
    "request_num_preemptions": [1, 2, 3, 4, 5, 10, 20],
    "kv_cache_residency": [0.001, 0.002, ..., 1800],
}

# 指标名 -> 家族键, 是「每个直方图都必须登记家族」的强制清单
METRIC_FAMILIES: dict[str, str] = {
    "vllm:e2e_request_latency_seconds": "request_latency",
    # ... 另外 4 个 request_latency 指标、2 个 inter_token_latency 指标 ...
    "vllm:request_prefill_kv_computed_tokens": "request_tokens",
    "vllm:kv_block_reuse_gap_seconds": "kv_cache_residency",
}

def collect_histogram_buckets() -> dict[str, list[float]]:
    """按指标名回读每个 vllm:* 直方图的有限桶上界。"""
    found: dict[str, list[float]] = {}
    for metric in prometheus_client.REGISTRY.collect():
        if metric.name.startswith("vllm:") and metric.type == "histogram":
            found[metric.name] = sorted(
                {
                    float(sample.labels["le"])
                    for sample in metric.samples
                    # 排除 +Inf, 只保留显式配置的上界
                    if sample.name.endswith("_bucket")
                    and sample.labels["le"] != "+Inf"
                }
            )
    return found

def test_prometheus_logger_default_buckets():
    """端到端: 每个核心引擎直方图都必须落在自己家族的默认边界上。

    配置中刻意不设置 kv_transfer_config: KV connector 会注册自己的直方图,
    它们保有自有边界, 不受 METRIC_FAMILIES 覆盖。
    """
    config = build_logger_config(ObservabilityConfig(kv_cache_metrics=True))
```

```

assert config.model_config.max_model_len == TEST_MODEL_MAX_LEN # 1024
try:
    PrometheusStatLogger(config)
    found = collect_histogram_buckets()
    # 注册集合必须与家族映射精确相等：新增指标却忘记登记就会失败
    assert set(found) == set(METRIC_FAMILIES)
    for metric_name, family in METRIC_FAMILIES.items():
        expected = (
            REQUEST_TOKENS_SNAPSHOT_FOR_TEST_MODEL
            if family == "request_tokens"
            else DEFAULT_BUCKET_SNAPSHOTS[family]
        )
        assert found[metric_name] == [float(b) for b in expected], metric_name
finally:
    # Prometheus 注册表是进程级全局状态，必须清理以免污染其他测试
    unregister_vllm_metrics()

```

评论区精华

本 PR 的评审痕迹非常薄：`review_comments_count` 为 0，唯一的实质性人类动作是维护者 robertgshaw2-redhat 的 APPROVED（无正文），claude[bot] 只留下一句「该 PR 来自 fork——自动 review 被禁用」，其余 7 条评论全是 `/ci run` 与 Buildkite 触发回执。因此真正的技术讨论集中在 PR body 与 commit message 里：

PR body 划界：「零行为变更：默认桶边界与当前 `main` 完全一致。issue 计划中唯一会改变行为的一项（把 `request_prefill_time` 重新分桶到 `prefill` 尺度的家族）被刻意推迟，以保持本 PR 是纯重构。」——这说明作者把「可评审性」放在「一次做完」之上。

关于重复实现：「#35952 提出同一合并方案，但自 5 月起停摆且测试计划未填；本 PR 的差异在于按家族给出理由文档（这正是计划里明确要求的）。如果 @npache 更倾向，我很乐意把两者收敛。」

commit `4638a10` 关于文档准确性：「该指标对每个结束的请求都会记录，因此第一个桶同时包含从未被抢占和只被抢占一次的请求；此前的措辞声称了边界并不提供的单位分辨率。」

commit `6f14d1f` 关于测试护栏：「快照测试断言注册的直方图集合与家族映射精确一致，所以 `main` 一合并进来它就捕获了这个新指标。」——这是本 PR 少见的「测试先于代码发现问题」的实证。

未解决的疑虑：没有人类 reviewer 对「快照测试硬编码是否值得维护成本」给出明确表态，也没有人评估过把同一个 bucket 列表对象共享给五个 Histogram 的隐式契约。

- 与既有合并 PR #35952 及改值 PR #48175 的边界划分 (design): 维护者接受本 PR 的定位（纯重构 + 按家族文档），以 APPROVED 结束；收敛 #35952 的可能性被搁置。
- 是否在本 PR 中一并重新分桶 `request_prefill_time` (design): 行为变更被显式延后，`request_prefill_time` 暂时与 `e2e/queue/inference/decode` 共用同一套桶；这一决策被测试中的 METRIC_FAMILIES 映射如实固化。

- 快照测试为何使用硬编码字面量而非 import 常量 (testing): 采用访问器级 + 注册表级双层快照; 后者在合并 main 后立刻捕获了新增的 `vllm:request_num_preemptions` 指标, 实证了该设计的价值。
- `histogram_buckets` 返回新副本的契约 (correctness): 契约被测试锁定; 但「多个 Histogram 共享同一返回列表」这一新形态未被测试覆盖。
- single-source-of-truth 的适用范围与 KV connector 自有边界 (documentation): 文档与测试的范围声明对齐, 避免了「全部指标边界都在此集中」的过度承诺。
- 合并 main 后暴露的新指标未纳入家族 (testing): 新增家族后测试恢复通过; 该事件被写进 commit message, 作为「注册集合精确匹配」断言有效性的证据。
- 自动评审不可用, 缺少独立技术 review (other): 以维护者批准合并收尾; 技术层面的独立复核缺位。

风险与影响

- 风险:
 1. 共享列表实例的隐式契约 (低概率、影响面大): `request_tokens_buckets` 是同一个 list 对象, 被 `vllm:request_prompt_tokens`、`request_generation_tokens`、`request_max_num_generation_tokens`、`request_params_max_tokens`、`request_prefill_kv_computed_tokens` 五个指标同时引用。这依赖 `prometheus_client` 在 Histogram 构造时把传入的 `buckets` 复制成新的浮点列表; 一旦该行为变化或后续有人在 `__init__` 之后原地 `append/sort` 这个列表, 五个指标的桶边界会一起被改写。`test_buckets_return_fresh_copies` 只锁定了访问器返回新副本, 没有覆盖「多个 Histogram 共享一个实例」这个新引入的形态。
 2. 错误类型不一致: `histogram_buckets` 对 `request_tokens` 缺参抛 `ValueError`, 但传入未知家族键时会走 `_STATIC_FAMILY_DEFAULTS[family]` 抛 `KeyError`。当前只有内部调用点, 风险很低; 但 #48867 引入用户可配置的家族名时, 必须显式用 `BUCKET_FAMILY_KEYS` 做校验并统一成 `ValueError`, 否则用户配置错误会以 `KeyError` 的形式冒出。
 3. 快照测试的维护与冲突成本: 273 行测试里大部分是硬编码数值, 任何有意的桶边界调整都必须同步改测试。例如 #48175 想改 `request_latency` 默认值时会直接红在这份快照上——这是设计意图, 但会让 metrics 相关的多 PR 并行变得更容易冲突; 本 PR 自己就因为 main 上新增的 `request_num_preemptions` 反复 merge 并返工。
 4. 测试对全局状态与具体模型的依赖: 端到端用例依赖 `prometheus_client` 的进程级 `REGISTRY` 和 `distilbert/distilgpt2` 的默认 `max_model_len == 1024`。注册表若被其他测试污染、或模型注册表默认值变化, 会以难以归因的方式失败; finally: `unregister_vllm_metrics()` 只降低了污染别人的风险, 没有隔离被别人污染的路径。
 5. 回归风险本身很低: 所有边界逐值与 main 一致 (PR body 明确声明, 并被快照测试固化), 对 Prometheus 抓取端、`histogram_quantile` 查询和既有 SLO 面板是零影响; 不改配置 schema、不改 CLI、不改序列化格式。
- 影响:
 - 对运维方 / 监控侧: 零影响。发布后所有 `vllm:*` 直方图的 bucket 边界与 main 完全相同, 已配置的 Grafana 面板、告警规则和 SLO 查询不需要任何调整。

- 对 vLLM 开发者: `vllm/v1/metrics/loggers.py` 减少 147 行内联数据, 新增指标时只需在 `buckets.py` 选择一个家族键, 边界知识集中在一处; 代价是任何默认边界的调整都必须同步更新 `tests/v1/metrics/test_histogram_buckets.py` 的硬编码快照, 属于「摩擦换安全」的显式取舍。
- 对后续功能: 本 PR 是 #48867 「桶可配置化」的直接前置。BucketFamilyKey / BUCKET_FAMILY_KEYS / `histogram_buckets()` 三层接口已经把「家族集合」与「默认值解析」解耦, 配置化只需替换 `histogram_buckets` 的数据来源, 调用点无需再改。
- 对团队协作: 新增的 METRIC_FAMILIES 映射 + 「注册集合必须精确等于映射集合」的断言, 会让任何新增 `vllm:*` 直方图却忘记登记家族的 PR 立刻失败——这是跨 PR 的强制登记机制, 正负面作用并存。
- 风险标记: 共享桶列表实例的隐式契约, 快照测试硬编码带来维护与冲突成本, 未知家族键抛 `KeyError` 而非 `ValueError`, 测试依赖进程级 Prometheus 注册表, 仅有维护者批准, 缺少独立技术 review

关联脉络

- PR #48867 Make Prometheus histogram buckets configurable (本 PR body 中引用的 follow-up 链接): PR body 明确把本 PR 定义为其 groundwork; `buckets.py` 中的 BucketFamilyKey / BUCKET_FAMILY_KEYS / `histogram_buckets()` 就是为了让配置化只替换数据来源而不动调用点。
- PR #35952 Consolidate Prometheus histogram bucket defaults (标题据 PR body 转述): 由 @npache 提出的同一合并方案, 自 5 月起停摆且测试计划未填; 本 PR 通过补充按家族理由文档与快照测试与其形成差异, 并在 body 中表示愿意收敛。
- PR #48175 调整 `request_latency` 默认桶值 (标题据 PR body 转述): 改动 `request_latency` 的默认值, 与本 PR 的零值变更定位正交; 两者都触及同一份边界清单, 很可能在本 PR 新增的快照测试上产生冲突。
- PR #34564 以每指标 CLI flag 暴露直方图桶 (标题据 PR body 转述, 已关闭): 与 #34720 一起代表被否决的「每指标一个 CLI flag」方案, 正是这些 changes requested 促成了「先合并再配置化」的路线, 本 PR 是该路线的第一步。
- PR #34720 Prometheus 直方图桶配置化的另一版尝试 (标题据 PR body 转述, 已关闭): 同上, 与本 PR 属于同一需求的不同实现路径, 用于解释本 PR 为何选择集中式家族模型而非分散 flag。