PR #40714 完整报告

vllm-project/vllm

[CPU] Create Proper Numa topology for s390x

合并时间：2026-07-13 12:58

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40714

执行摘要

- 一句话：为 s390x 构建层级 NUMA 拓扑以优化 TP 性能
- 推荐动作：建议阅读此 PR 以了解如何为异构 CPU 架构适配 NUMA 感知的核绑定。特别值得关注的点：cpu_resource_utils.py 中通过 numa_node 字段复用现有分组机制的设计权衡，以及 ompmultiprocessing.py 中根据 CPU 列表直接派生拓扑域以绕开物理内存节点的思路。虽然缺少自动化测试，但核心逻辑以注释和 warning 记录了假设，整体改动清晰。

功能与动机

s390x 的核层次结构为 drawer > book > socket > core，之前仅考虑 socket 层级，导致 TP 场景下跨 book 绑定核心，性能受损。本 PR 创建正确的 NUMA 拓扑以修复此问题。

实现拆解

1. 拓扑信息获取改造（vllm/utils/cpu_resource_utils.py）：在 _get_cpu_list 中，当检测到平台为 s390x 时，使用 lscpu -J -e=CPU,CORE,NODE,SOCKET,BOOK 获取扩展字段。解析后统计 book 和 socket 数量，优先选择 book（若多于一个 distinct 值）作为分组键，否则尝试 socket；若两者都无区分则回退通用路径。选定的分组键值被写入 LogicalCPUInfo.numa_node 字段，后续逻辑透明使用。
2. 内存节点初始化适配（vllm/v1/worker/cpu_worker.py）：CPUWorker.__init__ 中，由于 numa_node 在 s390x 上可能是合成的 book ID（不映射物理内存节点），新增检查：若 cpu_core.numa_node 在 allowed_memory_nodes 中则直接使用，否则发出警告并回退到 allowed_memory_nodes[0]。这避免了 C 侧内存初始化使用不存在的节点号。
3. OpenMP 自动绑核适配（vllm/utils/ompmultiprocessing.py）：_get_autobind_cpu_ids 中，当 CPU 架构为 s390x 时，从逻辑 CPU 列表的 numa_node 值直接推导允许的拓扑域（allowed_numa_nodes），替代原先通过环境变量 CPU_VISIBLE_MEMORY_NODES 获取物理内存节点的路径。这使得绑核逻辑以拓扑分组（book/socket）而非物理 NUMA 节点进行，与 cpu_resource_utils 的映射保持一致。
4. 平台感知扩展（vllm/platforms/cpu.py、vllm/distributed/device_communicators/cpu_communicator.py）：在需要 LD_PRELOAD libgomp 的架构列表中加入 CpuArchEnum.S390X，并在支持 SHM 通信器的架构列表中加入 S390X，确保进程间通信和 OpenMP 绑定在 s390x 上正确工作。

5. Docker 构建简化（docker/Dockerfile.s390x）：移除了 Apache Arrow 构建阶段，升级 numba 到 0.65.0 并直接依赖 llvmlite 0.47.0，精简了构建流程。

关键文件：

- vllm/utils/cpu_resource_utils.py（模块 资源工具；类别 source；类型 core-logic；符号 _get_cpu_list, LogicalCPUInfo, LogicalCPUInfo._int, LogicalCPUInfo.json_decoder）：核心改动：扩展 _get_cpu_list 以在 s390x 上解析 book/socket 拓扑并映射到 numa_node，实现分层 CPU 分组。
- vllm/v1/worker/cpu_worker.py（模块 CPU 工作器；类别 source；类型 core-logic；符号 CPUWorker.init）：使用者适配：处理 s390x 上 numa_node 可能为合成 book ID 的情况，回滚到真实内存节点。
- vllm/utils/ompmultiprocessing.py（模块 OMP 绑定；类别 source；类型 core-logic；符号 OMPMultiprocessing._get_autobind_cpu_ids, CpuArchEnum）：适配自动绑核：s390x 上从 CPU 列表的 numa_node 推导拓扑域，代替内存节点环境变量。
- vllm/platforms/cpu.py（模块 平台层；类别 source；类型 core-logic；符号 CpuPlatform.check_and_update_config）：将 S390X 加入需要 LD_PRELOAD libgomp 的架构列表，修正线程绑定。
- vllm/distributed/device_communicators/cpu_communicator.py（模块 通信器；类别 source；类型 core-logic；符号 CpuCommunicator.init）：将 S390X 加入支持 SHM 的架构列表，确保进程间通信正常。
- docker/Dockerfile.s390x（模块 构建脚本；类别 infra；类型 infrastructure）：大幅精简 Docker 镜像构建：移除了 Apache Arrow 构建阶段、升级 numba 和 llvmlite。
- requirements/common.txt（模块 依赖文件；类别 docs；类型 documentation）：可能更新了依赖版本（仅 1 行变化），对整体影响较小。

关键符号：_get_cpu_list, CPUWorker.init, OMPMultiprocessing._get_autobind_cpu_ids

关键源码片段

vllm/utils/cpu_resource_utils.py

核心改动：扩展 _get_cpu_list 以在 s390x 上解析 book/socket 拓扑并映射到 numa_node，实现分层 CPU 分组。

```
# vllm/utils/cpu_resource_utils.py
import platform
from dataclasses import dataclass
from typing import Any

@dataclass
class LogicalCPUInfo:
    id: int = -1
    physical_core: int = -1
    numa_node: int = -1 # 在 s390x 上可能映射为 book/socket ID

    @staticmethod
```

```
    def json_decoder(obj_dict: dict) -> dict:
        # 从 lscpu JSON 行提取字段
        return {
            "id": int(obj_dict.get("cpu", -1)),
            "physical_core": int(obj_dict.get("core", -1)),
            "numa_node": int(obj_dict.get("node", -1)),
        }

def _get_cpu_list() -> list[LogicalCPUInfo]:
    # ... 前置判断（macOS 等）...
    if platform.machine() == "s390x":
        # s390x 上使用扩展列 SOCKET, BOOK
        lscpu_output = subprocess.check_output(
            "lscpu -J -e=CPU,CORE,NODE,SOCKET,BOOK", shell=True, text=True
        )
    else:
        lscpu_output = subprocess.check_output(
            "lscpu --json --extended=CPU,CORE,NODE --online", shell=True, text=True
        )

    # 处理 NUMA undefined（映射为 0）
    lscpu_output = re.sub(r'"node":\s*-\s*(,|\n|\})', r'"node": 0\1', lscpu_output)

    if platform.machine() == "s390x":
        # 处理 book/socket 中可能的 "-"
        lscpu_output = re.sub(r'"book":\s*-\s*(,|\n|\})', r'"book": 0\1', lscpu_output)
        lscpu_output = re.sub(r'"socket":\s*-\s*(,|\n|\})', r'"socket": 0\1', lscpu_output)

        raw_cpus = json.loads(lscpu_output)["cpus"]
        # 收集 distinct values
        distinct_books = {LogicalCPUInfo._int(str(e.get("book", "-1"))) for e in raw_cpus} - {-1}
        distinct_sockets = {LogicalCPUInfo._int(str(e.get("socket", "-1"))) for e in raw_cpus} - {-1}

        # 选择最佳分组：优先 book（若多于 1 个），否则 socket
        if len(distinct_books) > 1:
            group_key = "book"
        elif len(distinct_sockets) > 1:
            group_key = "socket"
        else:
            group_key = None

        if group_key is not None:
            result = []
            for entry in raw_cpus:
                cpu_id = LogicalCPUInfo._int(str(entry.get("cpu", "-1")))
                core = LogicalCPUInfo._int(str(entry.get("core", "-1")))
                group = LogicalCPUInfo._int(str(entry.get(group_key, "-1")))
                if -1 not in (cpu_id, core, group):
                    # 将分组键值写入 numa_node，复用后续所有 NUMA 亲和逻辑
```

```
                result.append(LogicalCPUInfo(id=cpu_id, physical_core=core, numa_node=group))
            if result:
                return result
        # 若没有有效分组，回退通用路径
    # 通用路径（非 s390x 或 s390x 无分组）
    # ...
```

vllm/v1/worker/cpu_worker.py

使用者适配：处理 s390x 上 numa_node 可能为合成 book ID 的情况，回滚到真实内存节点。

```
# vllm/v1/worker/cpu_worker.py
class CPUWorker(Worker):
    def __init__(self, vllm_config, local_rank, rank, distributed_init_method, is_driver_worker=
    False):
        allowed_memory_nodes = get_visible_memory_node()
        allowed_cpu_list = get_allowed_cpu_list()
        cpu_core = allowed_cpu_list[0]

        # s390x 上 numa_node 可能是 book ID，不一定是真实内存节点
        if cpu_core.numa_node in allowed_memory_nodes:
            memory_node = cpu_core.numa_node
        else:
            # 回退到第一个可用的内存节点，并记录警告
            logger.warning(
                "CPU group key %s is not a valid memory node. "
                "Falling back to memory node %s.",
                cpu_core.numa_node,
                allowed_memory_nodes[0],
            )
            memory_node = allowed_memory_nodes[0]

        torch.ops._C.init_cpu_memory_env([memory_node])
        # 后续内存查询也使用回退后的 memory_node
        memory_status = get_memory_node_info(memory_node)
        # ...
```

vllm/utils/ompmultiprocessing.py

适配自动绑核：s390x 上从 CPU 列表的 numa_node 推导拓扑域，代替内存节点环境变量。

```
# vllm/utils/ompmultiprocessing.py
class OMPMultiprocessing:
    def _get_autobind_cpu_ids(self, cpu_selector):
        allowed_numa_nodes = cr_utils.get_visible_memory_node()
        logical_cpu_list = cr_utils.get_allowed_cpu_list()
        local_world_size = self.local_world_size

        # s390x 上 numa_node 已被重映射为拓扑分组键（book/socket）
        # 不再对应物理内存节点，改为从 CPU 列表派生域
        cpu_arch = current_platform.get_cpu_architecture()
```

```
if cpu_arch == CpuArchEnum.S390X:
    # 使用所有出现在 CPU 列表中的不同 numa_node 值作为拓扑域
    allowed_numa_nodes = sorted(set(cpu.numa_node for cpu in logical_cpu_list))

assert len(allowed_numa_nodes) >= local_world_size or self.simulate_multi_node, (
    f"Not enough allowed NUMA nodes to bind threads of "
    f"{local_world_size} local CPUWorkers. "
    f"Allowed NUMA nodes are {allowed_numa_nodes}."
)
# 后续按 allowed_numa_nodes 分配每个 rank 的 CPU 列表
# ...
```

评论区精华

BigPYJ1151 指出通用代码路径改动过多，建议通过复用 LogicalCPUInfo.numa_node 作为抽象分组键来减少侵入（如将 book/socket 映射到 numa_node 字段），作者采纳并简化了实现。另外 BigPYJ1151 质疑是否所有层级都必要，作者回应去掉 drawer 但仍保留 book，因为实测对 TP 性能有益。BigPYJ1151 要求对内存节点 fallback 添加日志，作者添加了 logger.warning。

- 通用代码路径改动过多 (design): 作者简化实现，将 book/socket 映射到 numa_node，利用现有分组模式。
- 支持层级深度 (design): 作者去掉 drawer，只保留 book 和 socket，因为 book 对 TP 有益且实测性能好。
- Synthetic numa_node 回退警告 (testing): 作者在 cpu_worker.py 中添加了 logger.warning。

风险与影响

- 风险：主要风险在通用代码路径的修改（_get_cpu_list）对非 s390x 平台的影响，但改动封闭在 platform.machine() == 's390x' 分支内，回归风险较低。s390x 上若 book ID 与物理内存节点不一致，回退逻辑可能掩盖配置错误，导致内存初始化使用错误的节点（虽然健康检查会捕获）。另外缺少自动化测试覆盖，只能依赖人工镜像测试。Dockerfile 的大幅删减可能破坏已有镜像的 Arrow 依赖，但清理动机明确。
- 影响：对用户：s390x 上 TP 和 DP 场景的推理性能显著改善（具体数值未提供，但 PR 作者展示绑核日志显示核分配正确）。对系统：改进了 CPU 资源管理工具的可扩展性，未来其他架构可参考加入自定义拓扑分组。对团队：维护者 BigPYJ1151 深度参与了设计和代码简化，减少了后续维护负担。
- 风险标记：缺少测试覆盖，平台特定路径回归风险低，Docker 依赖变更可能回滚

关联脉络

- 暂无明显关联 PR