PR #14194 完整报告

sgl-project/sglang

[feature] implement dcp for deepseek_v2

合并时间：2026-06-26 06:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/14194

执行摘要

- 一句话：为 DeepSeek-v2 实现 Decode Context Parallelism
- 推荐动作：该 PR 设计稳健、讨论充分，值得深入阅读。特别是 DCP 的虚拟 KV 容量策略和 Triton 在线 softmax 校正内核值得关注。建议在后续 PR 中补充 MHA 模型 DCP 的测试和 FlashMLA FP8 支持。对于需要超长上下文的 DeepSeek 用户，建议升级到包含此 PR 的版本。

功能与动机

来自 Issue #12196，目标是在 8×H20 环境下以 TP8 支持远超单 GPU 内存的上下文长度。DCP 通过让不同 rank 只存储部分 KV cache 来成倍扩展有效 KV cache 容量，同时保持计算精度。

实现拆解

1. 分布式初始化：在 server_args.py 中新增 --dcp-size 参数；在 parallel_state.py 的 initialize_model_parallel 中创建 _DCP group，并暴露 get_dcp_group、get_dcp_world_size 等 API。同时增加 reduce_scatter_along_dim 方法，支持沿任意维度的 reduce-scatter，用于 DCP 最终输出凝聚。
2. KV Cache 虚拟容量扩展：在 KV cache 分配层将可分配 token 数放大 dcp_world_size 倍，使得调度器认为可用 KV 槽位成倍增加。实际存储时每个 DCP rank 只保存 1/dcp_world_size 的 KV page，通过 out_cache_loc % dcp_world_size == rank 过滤。
3. 注意力计算重构（MLA 解码）：在 forward_mla.py 的 forward_absorb_prepare 中，对 decode batch 执行 all_gather_q_for_mla_decode，将各 TP rank 的局部 q 头收集为完整 q 头。在 forward_absorb_core 中，使用带 LSE 返回的 attn_mqa_for_dcp_decode（额外构造的 RadixAttention，其 num_heads = local_heads * dcp_world_size）计算局部注意力输出和 LSE，随后调用 cp_lse_ag_out_rs 进行所有 DCP rank 的 LSE all-gather、在线 softmax 校正，并最终 reduce-scatter 回每个 TP rank 的局部输出。
4. Prefill 扩展时 KV 收集：对于 extend 阶段，通过 all_gather_kv_cache_for_mla_extend 将各 DCP rank 的 prefix KV 张量收集到临时缓冲，使注意力计算看到完整的 prefix。
5. 测试与验证：新增端到端测试 test_dsv31_dcp8_gsm8k.py，在 8×GPU 上启动 DCP8+TP8 服务，验证 GSM8K 准确率 >=0.90 以及基本解码正确性。新增单元测试 test_reduce_scatter_along_dim.py，验证 reduce_scatter_along_dim 在不同维度、形状

和数据类型的正确性。

关键文件：

- python/sglang/srt/layers/utils/dcp_utils.py（模块 DCP 工具层；类别 source；类型 core-logic；符号 dcp_enabled, get_attention_dcp_group, get_attention_dcp_world_size, get_attention_dcp_rank）：核心 DCP 实现，包含 Triton JIT 内核的在线 softmax 校正函数及通信辅助函数。
- python/sglang/srt/distributed/parallel_state.py（模块 分布式层；类别 source；类型 core-logic；符号 reduce_scatter_along_dim, get_dcp_group_no_assert, get_dcp_world_size, get_dcp_rank）：新增 DCP 分布式组和 reduce_scatter_along_dim 方法，支撑 DCP 通信。
- test/registered/dcp/test_dsv31_dcp8_gsm8k.py（模块 集成测试；类别 test；类型 test-coverage；符号 _get_max_total_num_tokens, TestDSV31DCP8TP8GSM8K, setUpClass, tearDownClass）：端到端 DCP 正确性验证，包括 GSM8K 准确率和基本解码检查。

关键符号：GroupCoordinator.reduce_scatter_along_dim, get_dcp_group_no_assert, get_dcp_world_size, get_dcp_rank, dcp_enabled, get_attention_dcp_group, get_attention_dcp_world_size, get_attention_dcp_rank, _correct_attn_cp_out_kernel, correct_attn_out, CPTritonContext.call_kernel, all_gather_q_for_mla_decode, all_gather_kv_cache_for_mla_extend, cp_lse_ag_out_rs, prepare_decode_context_parallel_metadata, prepare_context_parallel_metadata_for_dcp, TestDSV31DCP8TP8GSM8K.test_dcp_activation_check, TestDSV31DCP8GSM8K.test_gsm8k_accuracy

关键源码片段

python/sglang/srt/distributed/parallel_state.py

新增 DCP 分布式组和 reduce_scatter_along_dim 方法，支撑 DCP 通信。

```
# python/sglang/srt/distributed/parallel_state.py (modified)
# 在 GroupCoordinator 中新增 reduce_scatter_along_dim

def reduce_scatter_along_dim(
    self, input_: torch.Tensor, dim: int = -1
) -> torch.Tensor:
    """沿指定 dim 进行 reduce-scatter，支持任意维度。

    先将目标维度移到 dim 0，make contiguous，再进行标准的
    reduce_scatter_tensor，最后将结果移回原维度位置。
    """
    world_size = self.world_size
    if world_size == 1:
        return input_
    assert (
        -input_.dim() <= dim < input_.dim()
```

```
    ), f"Invalid dim ({dim}) for shape {input_.size()}"
    if dim < 0:
        dim += input_.dim()
    with self.use_symmetric_memory(self):
        input_tensor = input_.movedim(dim, 0).contiguous()
    assert input_tensor.shape[0] % world_size == 0
    chunk_size = input_tensor.shape[0] // world_size
    output_shape = (chunk_size,) + input_tensor.shape[1:]
    with self.use_symmetric_memory(self):
        output_tensor = torch.empty(
            output_shape, dtype=input_tensor.dtype, device=input_tensor.device
        )
    self.reduce_scatter_tensor(output_tensor, input_tensor)
    return output_tensor.movedim(0, dim)

# DCP 组的初始化（在 initialize_model_parallel 中）
if decode_context_parallel_size > 1:
    # 创建 DCP 子组，划分方式与 TP 配合
    # ... 创建 _DCP group ...

# 全局访问函数
def get_dcp_group_no_assert() -> Optional[GroupCoordinator]:
    return _DCP

def get_dcp_group() -> GroupCoordinator:
    assert _DCP is not None, "DCP group not initialized"
    return _DCP

def get_dcp_world_size() -> int:
    if _DCP is None:
        return 1
    return _DCP.world_size

def get_dcp_rank() -> int:
    if _DCP is None:
        return 0
    return _DCP.rank
```

test/registered/dcp/test_dsv31_dcp8_gsm8k.py

端到端 DCP 正确性验证，包括 GSM8K 准确率和基本解码检查。

```
# test/registered/dcp/test_dsv31_dcp8_gsm8k.py (extract)
class TestDSV31DCP8TP8GSM8K(GSM8KMixin, BasicDecodeCorrectnessMixin, CustomTestCase)
:
    """DCP=8 + TP=8 的 CI 准确性门控测试。"""

    @classmethod
    def setUpClass(cls):
        # 启动 DCP8+TP8 服务
```

```
        cls.server_args = [
            "--tp-size", "8",
            "--dcp-size", "8",
            "--attention-backend", "flashinfer",
            "--enable-cache-report",
            "--chunked-prefill-size", "16384",
            "--cuda-graph-max-bs", "256",
            # ...
        ]
        cls.process = popen_launch_server(
            cls.model_path,
            cls.server_args,
            return_output=True,
        )

    def test_dcp_activation_check(self):
        """验证 DCP 激活：max_total_num_tokens 应随 dcp_world_size 缩放。"""
        info = requests.get(f"{base_url}/server_info").json()
        self.assertGreater(
            info["max_total_num_tokens"],
            _EXPECTED_BASELINE, # 比非 DCP 时大
            "DCP not active: max_total_num_tokens not scaled",
        )

    def test_gsm8k_accuracy(self):
        """GSM8K 200 题 5-shot 准确率 >= 0.90。"""
        accuracy = self.eval_gsm8k(num_questions=200, num_shots=5)
        self.assertGreaterEqual(accuracy, 0.90,
            f"GSM8K accuracy {accuracy} below threshold")
```

评论区精华

- 对称内存与 DCP 耦合：kpham-sgl 质疑对称内存与 DCP 的紧密耦合，staugust 解释对称内存可减少通信但因 NCCL 地址问题暂时只能对 DCP 启用；thanhhao98 补充对称内存非强约束，但能提升性能。
- 代码组织：kpham-sgl 要求将 DCP 辅助函数从 utils.py 移到新的 dcp_utils.py，并采用类似 cp_utils.py 的元数据类模式，staugust 遵从。
- 调度器改动争议：kpham-sgl 强调尽量减少对调度器的修改，staugust 解释 add_one_req_ignore_eos 中的改动是必要的，最后通过移除调度器修改并调整 prefill chunk 对齐解决。
- FlashMLA FP8 限制：自动代码审查指出 FlashMLA 不支持 DCP 与 FP8 组合，应添加文档说明。
- 中文注释：多位审查者要求将中文 TODO 注释翻译为英文，staugust 最终清理。
  - 对称内存与 DCP 耦合 (design): 保留环境变量 SGLANG_DCP_SYMM_ONLY 让对称内存仅对 DCP 启用，并计划在 torch 升级后移除。

- 代码组织：DCP 辅助函数移至 dcp_utils.py (style): staugust 同意并迁移，同时新建 attn_dcp_metadata 数据类。
- 调度器最小修改原则 (design): 放弃对 schedule_policy 的修改，改为在 prefill 阶段对齐 chunk 大小。
- FlashMLA FP8 限制 (performance): 添加明确断言和注释，提醒用户。
- 中文 TODO 注释清理 (style): staugust 最终清理并替换为英文注释。

风险与影响

- 风险：
  - GPU 内存虚拟化风险：KV 池容量被虚拟放大，若其他部分（如顶层 GPU 内存上下文）未相应调整，可能触发 OOM 或分配失败。
  - Attention 后端依赖：当前仅 flashinfer 后端完整支持 DCP，flashmla 后端存在 FP8 限制，若用户切换后端可能静默失败。
  - CUDA Graph 兼容性：虽然支持 CUDA graph，但 replay 时需重新捕获 DCP 相关张量，若设备预留内存不足可能导致捕获失败。
  - MHA extend 路径覆盖不全：DeepSeek-v2 使用 MLA，测试未覆盖 MHA extend 下的 DCP 路径，一旦未来模型使用 MHA，需补充支持。
  - 性能回退：DCP 引入额外 all-gather/reduce-scatter 通信，在小 batch 或短上下文场景可能不如纯 TP。
- 影响：
  - 用户影响：启用 --dcp-size N 后，用户可以处理更长的上下文或更大 batch，但需要多 GPU 并开启 flashinfer 后端。不启用时无行为变化。
  - 系统影响：引入新的分布式组 DCP，增加通信依赖。对称内存可用于优化通信。
  - 团队影响：该 PR 为后续 Helix Parallelism、TP+DP+DCP 等高级并行方案奠定了基础。团队需熟悉 DCP 新架构。
  - 风险标记：GPU 内存虚拟化风险，仅 flashinfer 后端，CUDA Graph 兼容性，MHA 路径未测试，通信开销增加

关联脉络

- PR #12196 Support for processing extremely long context: 父 Issue，本 PR 是其第一步实现。
- PR #21788 Helix Parallelism (TBD): 被维护者提及依赖于本 PR，表明 DCP 是更大并行方案的基础。