LCOV - code coverage report

Current view: top level - src/common/transformations/src/transformations/common_optimizations - convert_pagedattn_inputs.cpp (source / fu
Test: full_covergae.info
Test Date: 2025-04-10 19:26:12

```
      Line data    Source code
    1             : // Copyright (C) 2018-2025 Intel Corporation
    2             : // SPDX-License-Identifier: Apache-2.0
    3             : //
    4             :
    5             : #include "transformations/common_optimizations/convert_pagedattn_inputs.hpp"
    6             :
    7             : #include <cstdint>
    8             : #include <memory>
    9             : #include <transformations/utils/gen_pattern.hpp>
   10             :
   11             : #include "itt.hpp"
   12             : #include "openvino/core/rt_info.hpp"
   13             : #include "openvino/op/add.hpp"
   14             : #include "openvino/op/constant.hpp"
   15             : #include "openvino/op/paged_attention.hpp"
   16             : #include "openvino/util/log.hpp"
   17             : #include "transformations/utils/utils.hpp"
   18             : using namespace ov::gen_pattern;
   19             :
   20           1 : ov::pass::ConvertPagedAttnInputs::ConvertPagedAttnInputs(const KVCacheConfig& config, UpdateShapeFunc func)
   21           1 :     : m_config(config),
   22           1 :       m_update_shape_func(std::move(func)) {
   23           1 :     MATCHER_SCOPE(ConvertPagedAttnInputs);
   24             :
   25           1 :     auto Q = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   26           1 :     auto K = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   27           1 :     auto V = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   28           1 :     auto key_cache_0 = makePattern<ov::op::v0::Parameter>({});
   29           1 :     auto value_cache_0 = makePattern<ov::op::v0::Parameter>({});
   30           1 :     auto past_lens = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   31           1 :     auto subsequence_begins = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   32           1 :     auto block_indices = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   33           1 :     auto block_indices_begins = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   34           1 :     auto scale = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   35           1 :     auto sliding_window = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   36           1 :     auto alibi_slopes = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   37           1 :     auto max_context_len = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   38           1 :     auto rotated_block_indices = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   39           1 :     auto rotation_deltas = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   40           1 :     auto rotation_trig_lut = ov::pass::pattern::any_input(ov::pass::pattern::has_static_rank());
   41             :
   42          13 :     auto pa_1 = makePattern<op::PagedAttentionExtension>({Q,
   43             :                                                           K,
   44             :                                                           V,
   45             :                                                           key_cache_0,
   46             :                                                           value_cache_0,
   47             :                                                           past_lens,
   48             :                                                           subsequence_begins,
   49             :                                                           block_indices,
   50             :                                                           block_indices_begins,
   51             :                                                           scale,
   52             :                                                           sliding_window,
   53             :                                                           alibi_slopes,
   54          14 :                                                           max_context_len});
   55             :
   56          16 :     auto pa_2 = makePattern<op::PagedAttentionExtension>({Q,
   57             :                                                           K,
   58             :                                                           V,
   59             :                                                           key_cache_0,
   60             :                                                           value_cache_0,
   61             :                                                           past_lens,
   62             :                                                           subsequence_begins,
   63             :                                                           block_indices,
   64             :                                                           block_indices_begins,
   65             :                                                           scale,
   66             :                                                           sliding_window,
   67             :                                                           alibi_slopes,
   68             :                                                           max_context_len,
   69             :                                                           rotated_block_indices,
   70             :                                                           rotation_deltas,
   71          17 :                                                           rotation_trig_lut});
   72           1 :     auto result = pa_1 | pa_2;
   73          34 :     ov::matcher_pass_callback callback = [=](ov::pass::pattern::Matcher& m) {
   74          32 :         const auto pa_op = m.get_match_root();
   75          32 :         auto key_cache = ov::as_type_ptr<ov::op::v0::Parameter>(pa_op->get_input_node_shared_ptr(3));
   76          32 :         auto value_cache = ov::as_type_ptr<ov::op::v0::Parameter>(pa_op->get_input_node_shared_ptr(4));
   77             : #if defined(OPENVINO_ARCH_ARM64)
   78          64 :         auto format_cache_precision = [](ov::element::Type cache_precision, ov::element::Type infer_precision) {
   79          64 :             return ov::element::u8;
   80             :         };
   81             : #else
   82             :         auto format_cache_precision = [](ov::element::Type cache_precision, ov::element::Type infer_precision) {
   83             :             return cache_precision == ov::element::f16 && infer_precision == ov::element::bf16 ? infer_precision
   84             :                                                                                               : cache_precision;
   85             :         };
   86             : #endif
   87           0 :         auto init_cache_shape = [&](const size_t head_nums,
   88             :                                     const size_t head_size,
   89             :                                     const size_t block_size,
   90             :                                     const ov::element::Type precision,
   91             :                                     const size_t group_size,
   92             :                                     const bool bychannel,
   93             :                                     const std::vector<size_t>& orders) {
```

```
94          0 :             ov::Dimension::value_type _block_size = block_size;
95          0 :             ov::Dimension::value_type _head_nums = head_nums;
96          0 :             ov::Dimension::value_type _head_size = head_size;
97          0 :             ov::Dimension::value_type _group_size = group_size;
98          0 :             _group_size = _group_size ? _group_size : _head_size;
99          0 :             if (!bychannel) {
100         0 :                 if (_head_size % _group_size != 0) {
101         0 :                     OPENVINO_THROW("cache head_size ", head_size, "cannot be divided by group_size ", group_size);
102           :                 }
103           :             }
104         0 :             size_t group_num = _head_size / _group_size;
105         0 :             m_update_shape_func(precision, bychannel, group_num, _head_size, _block_size);
106           :
107         0 :             auto block_shape = ov::PartialShape::dynamic(4);
108         0 :             block_shape[orders[0]] = -1;
109         0 :             block_shape[orders[1]] = _head_nums;
110         0 :             block_shape[orders[2]] = _block_size;
111         0 :             block_shape[orders[3]] = _head_size;
112           :
113         0 :             return block_shape;
114         0 :         };
115        32 :         auto key_cache_precision = format_cache_precision(m_config.keyCachePrecision, m_config.inferencePrecision);
116        32 :         auto value_cache_precision = format_cache_precision(m_config.valueCachePrecision, m_config.inferencePrecision);
117        32 :         key_cache->set_element_type(key_cache_precision);
118        32 :         value_cache->set_element_type(value_cache_precision);
119        64 :         if (!pa_op->get_rt_info().count("num_k_heads") || !pa_op->get_rt_info().count("k_head_size") ||
120        64 :             !pa_op->get_rt_info().count("num_v_heads") || !pa_op->get_rt_info().count("num_v_heads")) {
121           :             OPENVINO_DEBUG("PagedAttn ",
122           :                            pa_op->get_friendly_name(),
123           :                            " doesn't have rtinfo for num_k_heads/k_head_size/num_v_heads/num_v_heads");
124        32 :             return false;
125           :         }
126         0 :         const auto key_cache_shape = init_cache_shape(pa_op->get_rt_info()["num_k_heads"].as<size_t>(),
127         0 :                                                       pa_op->get_rt_info()["k_head_size"].as<size_t>(),
128           :                                                       m_config.keyCacheBlockSize,
129           :                                                       key_cache_precision,
130           :                                                       m_config.keyCacheGroupSize,
131           :                                                       m_config.keyCacheQuantBychannel,
132         0 :                                                       m_config.keyCacheDimOrder);
133         0 :         const auto value_cache_shape = init_cache_shape(pa_op->get_rt_info()["num_v_heads"].as<size_t>(),
134         0 :                                                         pa_op->get_rt_info()["v_head_size"].as<size_t>(),
135           :                                                         m_config.valueCacheBlockSize,
136           :                                                         value_cache_precision,
137           :                                                         m_config.valueCacheGroupSize,
138           :                                                         m_config.valueCacheQuantBychannel,
139         0 :                                                         m_config.valueCacheDimOrder);
140           :
141         0 :         key_cache->set_partial_shape(key_cache_shape);
142         0 :         value_cache->set_partial_shape(value_cache_shape);
143           :
144         0 :         key_cache->validate_and_infer_types();
145         0 :         value_cache->validate_and_infer_types();
146         0 :         return true;
147        33 :     };
148           :
149         1 :     auto m = std::make_shared<ov::pass::pattern::Matcher>(result, matcher_name);
150         1 :     this->register_matcher(m, callback);
151         3 : }
152           :
153         0 : void ov::pass::ConvertPagedAttnInputs::setKVCacheConfig(const KVCacheConfig& config) {
154         0 :     m_config = config;
155         0 : }
156           :
157         0 : const ov::pass::ConvertPagedAttnInputs::KVCacheConfig& ov::pass::ConvertPagedAttnInputs::getKVCacheConfig() const {
158         0 :     return m_config;
159           : }
```