



 Markshilong committed on Jun 11

 Commits on Jun 13, 2023


update

 Markshilong committed on Jun 13

 Commits on Jun 14, 2023


update

Shilong Lei authored and Shilong Lei committed on Jun 14


 Commits on Jun 19, 2023

update

 Markshilong committed on Jun 19


 Commits on Jul 4, 2023


update

 Markshilong committed on Jul 4

 Commits on Jul 7, 2023


update

 Markshilong committed on Jul 7


 Commits on Jul 18, 2023


update

 Markshilong committed last month

 Commits on Jul 25, 2023

updateupdate

 Markshilong committed 3 weeks ago

 Commits on Aug 4, 2023

update

 Markshilong committed last week

 Showing 12 changed files with 191 additions and 47 deletions.


Split

Unified

 7 MANIFEST.in 

Load diff




This file was deleted.

2   csrc/aio/py_lib/deepspeed_py_aio_handle.cpp 

```

172 172     }
173 173
174 174     int deepspeed_aio_handle_t::wait()
175 175     - {
175 175     + {
176 176         assert(_num_pending_ops > 0);
177 177         auto num_completed_ops = 0;
178 178

```



6   deepspeed/runtime/engine.py 

```

306 306         model_parameters = list(model_parameters)
307 307
308 308         if has_optimizer:
309 309     +         print("!!!!!! line 309")
309 310             self._configure_optimizer(optimizer, model_parameters)
310 311             self._configure_lr_scheduler(lr_scheduler)
311 312             self._report_progress(0)
312 313         elif self.zero_optimization():
313 314     +         print("!!!!!! line 314")
313 315             # no optim selected but zero is enabled
314 316             self.optimizer = self._configure_zero_optimizer(optimizer=None)
315 317         elif self.bfloat16_enabled():
1379 1381         return optimizer
1380 1382
1381 1383     def _configure_zero_optimizer(self, optimizer):
1382 1384     +         # lslmark: _configure_zero_optimizer
1382 1385         zero_stage = self.zero_optimization_stage()
1383 1386         mics_shard_size = self.mics_shard_size()
1384 1387
1412 1415         if overlap_comm:
1413 1416             logger.warning("Pipeline parallelism does not support overlapped
communication, will be disabled.")
1414 1417             overlap_comm = False
1415 1418     +         # lslmark: DeepSpeedZeroOptimizer
1415 1419             optimizer = DeepSpeedZeroOptimizer(
1416 1420                 optimizer,
1417 1421                 self.param_names,
1444 1448             elif zero_stage == ZeroStageEnum.weights:
1445 1449                 assert not self.has_moe_layers, "MoE not supported with Stage 3"
1446 1450                 if isinstance(optimizer, DummyOptim):
1447 1451     +                 print("!!!!!!!!!!!!Getting into - if isinstance(optimizer, DummyOptim): line 1449
of engine.py")
1447 1452                 log_dist("Creating ZeRO Offload", ranks=[0])
1448 1453                 optimizer = DeepSpeedZeROOffload(self.module,
1449 1454                     timers=timers,
1457 1462                     offload_param_config=self.zero_offload_param(),
1458 1463                     mpu=self.mpu)
1459 1464             else:

```

	1465	+	<code>print("!!!!!!Getting into - DummyOptim else: line 1463 of engine.py")</code>
1460	1466		<code>log_dist(</code>
1461	1467		<code> f'Creating fp16 ZeRO stage {zero_stage} optimizer,'</code>
1462	1468		<code> f' MiCS is enabled {mics_shard_size>0},'</code>

27  deepspeed/runtime/pipe/engine.py 

```

4      4      # DeepSpeed Team
5      5
6      6      from types import MethodType
7      7      -
8      8      + from deepspeed.utils.nvtx import my_nvtx_wrapper, my_nvtx_wrapper
9      9      + from torch.cuda import nvtx
10     10     import torch
11     11     from deepspeed import comm as dist
54     55     def __init__(self, has_bool_tensors=False, *super_args, **super_kwargs):
55     56     super().__init__(*super_args, **super_kwargs)
56     57     assert isinstance(self.module, PipelineModule), "model must base PipelineModule"
57     57     -
58     58     - assert self.zero_optimization_stage() < 2, "ZeRO-2 and ZeRO-3 are incompatible with
59     59     + # my comment:
60     60     + print("!!!! DISABLE: ZeRO-2 and ZeRO-3 are incompatible with pipeline parallelism")
61     61     + # assert self.zero_optimization_stage() < 2, "ZeRO-2 and ZeRO-3 are incompatible with
62     62     pipeline parallelism"
222    224     # We schedule the all-reduces, so disable it in super().backward()
223    225     self.enable_backward_allreduce = False
224    226     pipe_data_loader = RepeatingLoader(pipe_data_loader)
225    227     self.set_data_loader(pipe_data_loader)
226    228
227    229     + @my_nvtx_wrapper
228    230     def _exec_reduce_tied_grads(self):
229    231     +
230    232     # We need to run this first to write to self.averaged_gradients;
231    233     # since this class turns `enable_backward_allreduce` off,
232    234     # `self.overlapping_partition_gradients_reduce_epilogue()` defined in the
233    235     DeepSpeedEngine
239    243     grad = weight._hp_grad if self.bfloat16_enabled() else weight.grad
240    244     dist.all_reduce(grad, group=group)
241    245
242    246     + @my_nvtx_wrapper
243    247     def _exec_reduce_grads(self):
244    248     self._force_grad_boundary = True
245    249     if self.pipeline_enable_backward_allreduce:
333    338     sched = schedule.TrainSchedule(micro_batches=self.micro_batches,
334    339                                     stages=self.num_stages,
335    340                                     stage_id=self.stage_id)
341    341     + # nvtx.range_push("Execute_schedule")
336    342     self._exec_schedule(sched)
343    343     + # nvtx.range_pop()
337    344     self.agg_train_loss = self._aggregate_total_loss()
338    345
339    346     self.timers('train_batch').stop()
340    347     # TODO: should return precisely what loss returned and allow others to be queried?
341    348     return self.agg_train_loss
342    349
343    350
344    351
345    352
346    353
347    354
348    355
349    356
350    357
351    358
352    359

```

```

370 + @my_nvtx_wrapper
363 371 def eval_batch(self, data_iter, return_logits=False, compute_loss=True,
      reduce_output='avg'):
364 372     """Evaluate the pipeline on a batch of data from ``data_iter``. The
365 373     engine will evaluate ``self.train_batch_size()`` total samples
597 605
598 606     return batch
599 607
608 + @my_nvtx_wrapper
600 609 def _exec_forward_pass(self, buffer_id):
601 610     self.tput_timer.start()
602 611     self.mem_status('BEFORE FWD', reset_max=True)
677 686     for idx, l in enumerate(self.loss):
678 687         self.total_loss[idx] += l.detach()
679 688
689 +
690 + @my_nvtx_wrapper
680 691 def _exec_backward_pass(self, buffer_id):
681 692     assert self.optimizer is not None, "must provide optimizer during " \
682 693         "init in order to use backward"
751 762
752 763     self.mem_status('AFTER BWD')
753 764
765 + @my_nvtx_wrapper
754 766 def _exec_load_micro_batch(self, buffer_id):
755 767     if self.wall_clock_breakdown():
756 768         self.timers('batch_input').start()
910 922     else:
911 923         raise NotImplementedError(f'Could not receive type {type(recv_type)}')
912 924
925 + @my_nvtx_wrapper
913 926 def _exec_send_activations(self, buffer_id):
914 927     if self.wall_clock_breakdown():
915 928         self.timers('pipe_send_output').start()
946 959     if self.wall_clock_breakdown():
947 960         self.timers('pipe_send_output').stop()
948 961
962 + @my_nvtx_wrapper
949 963 def _exec_send_grads(self, buffer_id):
950 964     if self.wall_clock_breakdown():
951 965         self.timers('pipe_send_grad').start()
1002 1016     if self.wall_clock_breakdown():
1003 1017         self.timers('pipe_send_grad').stop()
1004 1018
1019 + @my_nvtx_wrapper
1005 1020 def _exec_rcv_activations(self, buffer_id):
1006 1021     if self.wall_clock_breakdown():
1007 1022         self.timers('pipe_rcv_input').start()
1045 1060     if self.wall_clock_breakdown():
1046 1061         self.timers('pipe_rcv_input').stop()
1047 1062
1063 + @my_nvtx_wrapper
1048 1064 def _exec_rcv_grads(self, buffer_id):
1049 1065     if self.wall_clock_breakdown():
1050 1066         self.timers('pipe_rcv_grad').start()
1102 1118     if self.wall_clock_breakdown():
1103 1119         self.timers('pipe_rcv_grad').stop()
1104 1120
1121 + @my_nvtx_wrapper

```

```

1105 1122     def _exec_optimizer_step(self, lr_kwargs=None):
1106 1123         if self.wall_clock_breakdown():
1107 1124             self.timers('step_microstep').start()
1289 1306         schedule.SendGrad: _exec_send_grads,
1290 1307         schedule.RecvGrad: _exec_recv_grads,
1291 1308     }
1292 1309 -
1292 1309 + @my_nvtx_wrapper
1293 1310     def _exec_schedule(self, pipe_schedule):
1294 1311         # Reserve and reset buffers.
1295 1312         self._reserve_pipe_buffers(pipe_schedule.num_pipe_buffers())
1304 1321
1305 1322         # Equivalent to: self._exec_forward_pass(buffer_id=0)
1306 1323         self._exec_instr = MethodType(self._INSTRUCTION_MAP[type(cmd)], self)
1306 1324 +         # nvtx.range_push(f'{self._exec_instr.__name__}')
1307 1325         self._exec_instr(**cmd.kwargs)
1307 1326 +         # nvtx.range_pop()

```

2 deepspeed/runtime/pipe/p2p.py

```

61 61         else:
62 62
63 63         if can_send_recv():
64 64 -         return dist.send(tensor, dest_rank)
64 64 +         return dist.send(tensor, dest_rank) # torch.distributed.send(tensor=tensor, dst=dst,
        group=group, tag=tag)
65 65         else:
66 66             group = _get_send_recv_group(src_stage, dest_stage)
67 67             src_rank = _grid.stage_to_global(stage_id=src_stage)

```

43 deepspeed/runtime/swap_tensor/partitioned_param_swapper.py

```

35 35
36 36     class AsyncPartitionedParameterSwapper(object):
37 37
38 38 -     def __init__(self, ds_config, model_dtype):
38 38 +     def __init__(self, ds_config, model_dtype, my_version):
39 39 +
40 40 +         self.my_version = my_version
39 41
40 42         aio_op = AsyncIOBuilder().load(verbose=False)
41 43         self.aio_handle = aio_op.aio_handle
74 76
75 77         self.invalid_buffer = torch.tensor(1).half()
76 78
79 79 +         if self.my_version:
80 80 +             self.finished_flag = False
81 81 +         else:
82 82 +             self.finished_flag = True
83 83 +
77 84         if dist.get_rank() == 0:
78 85             exclude_list = ['aio_read_handle', 'aio_write_handle', 'buffers']
79 86             print_object(obj=self, name='AsyncPartitionedParameterSwapper',
        exclude_list=exclude_list)
86 93         torch_dtype_string = str(self.dtype).split(".")[1]
87 94         self.swap_folder = os.path.join(self.swap_config.nvme_path, 'zero_stage_3',
        f'{torch_dtype_string}params',
88 95         f'rank{dist.get_rank()}')

```

```

89      -      shutil.rmtree(self.swap_folder, ignore_errors=True)
90      -      os.makedirs(self.swap_folder, exist_ok=True)
91
92      96      +
93      97      +      if (self.my_version == False):
94      98      +          shutil.rmtree(self.swap_folder, ignore_errors=True)
95      99      +          os.makedirs(self.swap_folder, exist_ok=True)
96
97      100
98      101      self.swap_element_size = torch.tensor([], dtype=self.dtype).element_size()
99      102
100     103
101     104
102     105      print_rank_0(f"param {param.ds_id} is assigned swap in buffer id {buffer_id} ")
103     106
104     107      self.param_id_to_buffer_id[param_id] = buffer_id
105     108
106     109      aligned_swap_numel = self._io_aligned_numel(self.param_id_to_numel[param_id])
107     110
108     111      -      swap_buffer = self.buffers.narrow(0, int(buffer_id *
109     112      self.aligned_elements_per_buffer), aligned_swap_numel)
110     113
111     114      190      +      swap_buffer = self.buffers.narrow(0, int(buffer_id *
112     115      self.aligned_elements_per_buffer), aligned_swap_numel) # torch.Tensor.narrow(dim, start, length)
113     116      → Tensor
114
115     117
116     118      self.param_id_to_swap_buffer[param_id] = swap_buffer
117     119
118     120      compute_buffer = swap_buffer.narrow(0, 0, self.param_id_to_numel[param_id])
119     121
120     122      def synchronize_writes(self):
121     123
122     124          if self.pending_writes == 0:
123     125
124     126              return
125     127
126     128      194      -      assert self.pending_writes == self.aio_write_handle.wait()
127     129
128     130          203      +      # original
129     131          204      +      # assert self.pending_writes == self.aio_write_handle.wait()
130     132          205      +
131     133          206      +      # my start
132     134          207      +      if (self.my_version):
133     135          208      +          if self.finished_flag == True:
134     136          209      +              assert self.pending_writes == self.aio_write_handle.wait()
135     137          210      +          else:
136     138          211      +              assert self.pending_writes == self.aio_write_handle.wait()
137     139          212      +          # my end
138     140          213      +
139     141
140     142      self.pending_writes = 0
141     143
142     144      self.remove_partition_and_release_buffers(self.swap_out_params)
143     145
144     146      self.swap_out_params = []
145     147
146     148      201     220      if self.pending_reads == 0:
147     149      202     221          return
148     150
149     151
150     152      204      -      assert self.pending_reads == self.aio_read_handle.wait()
151     153
152     154          223      +      assert self.pending_reads == self.aio_read_handle.wait() #SEARCH: int
153     155          deepspeed_aio_handle_t::wait()
154     156
155     157
156     158      self.pending_reads = 0
157     159
158     160
159     161
160     162
161     163      swap_out_params = self._get_swap_buffers(params)
162     164
163     165      self._track_numel(params)
164     166
165     167
166     168
167     169      252      -      swap_out_tensors(self.aio_write_handle, swap_out_params, swap_out_paths)
168     170
169     171          271      +      # my start
170     172          272      +      if (self.my_version):
171     173          273      +          swap_out_tensors(self.aio_write_handle, swap_out_params, swap_out_paths,
172     174          self.finished_flag)
173     175          274      +          else:
174     176          275      +              swap_out_tensors(self.aio_write_handle, swap_out_params, swap_out_paths, True)
175     177          276      +          # my end
176     178          277      +

```

```

278 +         # original
279 +         # swap_out_tensors(self.aio_write_handle, swap_out_params, swap_out_paths, True)
253 280
254 281         self.pending_writes += len(swap_out_params)
255 282         self.swap_out_params += params
307 334         self._update_inflight_swap_in(params, swap_in_buffers, inflight_numel)
308 335
309 336         if not async_op:
310 337 -             self.synchronize_reads()
310 337 +             self.synchronize_reads() # lsl: After this, all self.pending_reads == 0, copy from
buffer to param.ds_tensor.data
311 338
312 339         # Enables swapping into buffer that is out the control of swapper. This is always
synchronous
313 340         def swap_into_buffer(self, param, dest_buffer):

```

6 deepspeed/runtime/swap_tensor/utils.py

```

21 21         assert (swap_handle.async_pread(buffer, path) == 0)
22 22
23 23
24 24 - def swap_out_tensors(swap_handle, tensor_buffers, swap_paths):
24 24 + def swap_out_tensors(swap_handle, tensor_buffers, swap_paths, finished_flag=True):
25 25         for buffer, path in zip(tensor_buffers, swap_paths):
26 26 -             assert (swap_handle.async_pwrite(buffer, path) == 0)
26 26 +             # my next line is mine
27 27 +             if finished_flag == True:
28 28 +                 assert (swap_handle.async_pwrite(buffer, path) == 0)
27 29
28 30
29 31         def print_object(obj, name, exclude_list=[]):

```

8 deepspeed/runtime/zero/parameter_offload.py

```

246 246         self.forward_hooks = []
247 247         self.backward_hooks = []
248 248
249 249 - self.setup_zero_stage3_hooks()
249 249 + # self.setup_zero_stage3_hooks()
250 250         print_rank_0(
251 251             f'Created module hooks: forward = {len(self.forward_hooks)}, backward =
{len(self.backward_hooks)}',
252 252             force=False)
280 280
281 281         def _convert_to_zero_parameters(self, ds_config, module, mpu):
282 282             non_zero_params = [p for p in module.parameters() if not is_zero_param(p)]
283 283 -             if non_zero_params:
283 283 +             if non_zero_params: # if non_zero_params is not empty
284 284                 zero_params = [p for p in module.parameters() if is_zero_param(p)]
285 285 -             if zero_params:
285 285 +             if zero_params: # if zero_params is not empty, then we need to convert to zero
parameters
286 286                 zero_params[0].convert_to_zero_parameters(param_list=non_zero_params)
287 287 -             else:
287 287 +             else: # Here, first time - zero_params is empty
288 288                 group = None
289 289                 if mpu:
290 290                     group = mpu.get_data_parallel_group()

```


55 deepspeed/runtime/zero/partition_parameters.py

```

40 40
41 41     class NoGatherHandle:
42 42
43 43 +     @instrument_w_nvtx
44 44     def __init__(self, param: Parameter) -> None:
45 45         if param.ds_status != ZeroParamStatus.INFLIGHT:
46 46             raise RuntimeError(f"expected param {param.ds_summary()} to be available")
47 47 -
48 48 +
49 48         param.data = param.ds_tensor.data.to(device=get_accelerator().current_device_name(),
50 49             non_blocking=True).view(param.ds_shape)
51 50
52 51         self.__param = param
53 52
54 53 +     @instrument_w_nvtx
55 54     def wait(self) -> None:
56 55         get_accelerator().current_stream().synchronize()
57 56         self.__param.ds_status = ZeroParamStatus.AVAILABLE
58 57
59 58     class InsertPostInitMethodToModuleSubClasses(object):
60 59
61 60     def __init__(self, enabled=True, mem_efficient_linear=True, ds_config=None, dtype=None):
62 61
63 62 +         import sys
64 63 +         sys.path.append('/home/mark/Research/a_MoE_experiments/my_debug_utils')
65 64 +         from my_debug_utils import my_skip_1_enabled
66 65 +         self.my_version = my_skip_1_enabled # True to skip stage 1 (write to NVMe .swp)
67 66 +         self.my_print = True
68 67 +
69 68
70 69         self.mem_efficient_linear = mem_efficient_linear
71 70         self.enabled = enabled
72 71         self._set_dtype(ds_config, dtype)
73 72         self.wrapped_cls = set()
74 73
75 74     def __enter__(self):
76 75 +         if(self.my_print):print("!!!!!!! executing Init.fatherClass - __enter__()!!!!!!")
77 76         if not self.enabled:
78 77             return
79 78
80 79
81 80         zero_init_context.append(self)
82 81
83 82     def __exit__(self, exc_type, exc_value, traceback):
84 83 +         if(self.my_print):print("!!!!!!! executing Init.fatherClass - __exit__()!!!!!!")
85 84         if not self.enabled:
86 85             return
87 86
88 87         zero_init_context.pop()
89 88         if self.nest_level == 0:
90 89             if dist.get_rank() == 0:
91 90 +                 if(self.my_version):self.set_finished_flag_True()
92 91                 logger.info("finished initializing model with %.2fB parameters", param_count /
93 92                     1e9)
94 93 -
95 94         # Now that we cleaned up the metaclass injection, raise the exception.
96 95         if exc_type is not None:
97 96             return False
98 97
99 98         # To be implemented by inheriting classes
100 99     def _post_init_method(self, module):

```

```

462 472         pass
473 +
474 +     # To be implemented by inheriting classes
475 +     def set_finished_flag_True(self):
476 +         pass
463 477
464 478     def _set_dtype(self, ds_config, dtype):
465 479         if ds_config is not None and dtype is None:
585 599
586 600         self.complete = True
587 601
588 -
602 + @instrument_w_nvtx
589 603     def _no_gather_coalesced(params: Iterable[Parameter]) -> AllGatherCoalescedHandle:
590 604         for param in params:
591 605             if param.ds_status != ZeroParamStatus.NOT_AVAILABLE:
717 731
718 732                 model = deepspeed.zero.Init(module=model)
719 733             """"
734 +         # SEARCH: zero.Init()
720 735         if config is not None:
721 736             config_dict_or_path = config
722 737             logger.warning(
726 741         if _ds_config is not None:
727 742             mem_efficient_linear = _ds_config.zero_config.memory_efficient_linear
728 743         super().__init__(enabled=enabled, mem_efficient_linear=mem_efficient_linear,
ds_config=_ds_config, dtype=dtype)
744 +         if (self.my_print): print("!!!!!!! executing deepspeed.zero.Init __init__()!!!!!!")
745 +
729 746         if not dist.is_initialized():
730 747             init_distributed()
731 748         assert dist.is_initialized(), "Parameters cannot be scattered without initializing
deepspeed.comm"
759 776
760 777         # Enable fp16 param swapping to NVMe
761 778         if self.remote_device == OffloadDeviceEnum.nvme:
762 -         self.param_swapper = AsyncPartitionedParameterSwapper(_ds_config, self.dtype)
779 +         self.param_swapper = AsyncPartitionedParameterSwapper(_ds_config, self.dtype,
self.my_version)
763 780         else:
764 781             self.param_swapper = None
765 782
772 789         if not self.use_all_gather_into_tensor:
773 790             logger.info(f"all_gather_into_tensor API is not available in torch
{torch.__version__}")
774 791
792 +     def set_finished_flag_True(self):
793 +         if (self.my_version and self.param_swapper is not None):
794 +             self.param_swapper.finished_flag = True
795 +             if (self.my_print): print("!!!!!!! flag has been set to True!!!!!!")
796 +
775 797     def _update_persist_config(self, ds_config):
776 798         Init.apply_param_persistence = True
777 799         Init.param_persistence_threshold = ds_config.zero_config.param_persistence_threshold
805 827         see_memory_usage(f"Before converting and partitioning parmas in
{module.__class__.__name__}", force=False)
806 828
807 829         global param_count
830 +

```

```

808 831         for name, param in module.named_parameters(recurse=False):
809 832             param_count += param.numel()
810 833             if not is_zero_param(param):
811 834                 self._convert_to_deepspeed_param(param)
812 835                 print_rank_0(
813 836                     f"Partitioning param {debug_param2name_id_shape(param)} module=
{debug_module2name(module)}")

814 837
815 -             if get_accelerator().on_accelerator(param):
816 -                 dist.broadcast(param, 0, self.get_dp_process_group())
817 -             else:
818 -                 if dist.get_rank() == 0:
819 -                     logger.warn(f"param `{name}` in {module.__class__.__name__} "
820 -                               f"not on GPU so was not broadcasted from rank 0")
821 -
822 838 +             # if get_accelerator().on_accelerator(param):
823 839 +             #     # lslmark: for Pipeline, temporarily comment
824 840 +             #     dist.broadcast(param, 0, self.get_dp_process_group())
825 841 +             # else:
826 842 +             #     if dist.get_rank() == 0:
827 843 +             #         logger.warn(f"param `{name}` in {module.__class__.__name__} "
828 844 +             #                       f"not on GPU so was not broadcasted from rank 0")
829 845 +
830 846 +             param.partition()
831 847 +             see_memory_usage(
832 848 +                 f"Param count {param_count}. After converting and partitioning paramas in
{module.__class__.__name__}",
1035 1059         def _partition_numel(self, param):
1036 1060             return param.ds_tensor.ds_numel
1037 1061
1038 1062 + @instrument_w_nvtx
1039 1063         def _ensure_availability_of_partitioned_params(self, params):
1040 1064             swap_in_list = []
1041 1065             swap_in_flight = []
1042 1066
1043 1067             assert param.ds_tensor.final_location == OffloadDeviceEnum.nvme and
param.ds_status == ZeroParamStatus.NOT_AVAILABLE
1044 1068             swap_in_flight.append(param)
1045 1069
1046 1070             if len(swap_in_list) > 0:
1047 1071 -                 swap_in_list[0].nvme_swapper.swap_in(swap_in_list, async_op=False)
1048 1072 +                 swap_in_list[0].nvme_swapper.swap_in(swap_in_list, async_op=False) # SEARCH: def
1049 1073 + swap_in(self, params, async_op=True, swap_in_buffers=None):
1050 1074 +             elif len(swap_in_flight) > 0:
1051 1075 -                 swap_in_flight[0].nvme_swapper.synchronize_reads()
1052 1076 +                 swap_in_flight[0].nvme_swapper.synchronize_reads() # SEARCH: def
1053 1077 + synchronize_reads(self):
1054 1078
1055 1079         @instrument_w_nvtx
1056 1080         def _all_gather(self, param_list, async_op=False, hierarchy=None):
1057 1081             # print(f"Releasing {param.data.numel()}")
1058 1082             if param.ds_tensor is not None and not has Been updated:
1059 1083
1060 1084 +
1061 1085             #param.data = param.ds_tensor.data
1062 1086
1063 1087             see_memory_usage(f'Before partitioning param {param.ds_id} {param.shape}',
force=False)
1064 1088             start = partition_size * self.get_partition_rank()
1065 1089             end = start + partition_size
1066 1090

```

```

1163 | -         one_dim_param = param.contiguous().view(-1)
1189 | +         one_dim_param = param.contiguous().view(-1) # create a new tensor with the same
      | data but with a contiguous memory layout
1164 | 1190
1165 |         if start < param.ds_numel and end <= param.ds_numel:
1166 |             src_tensor = one_dim_param.narrow(0, start, partition_size)
1658 | 1684         handles = [dist.broadcast(p, self.src_rank, group=p.ds_process_group, async_op=True) for
      | p in self.params]
1659 |         for h in handles:
1660 |             h.wait()
      | +
1661 | 1688         self.params[0].partition(param_list=self.params, has_been_updated=True)

```

70 deepspeed/runtime/zero/partitioned_param_coordinator.py

```

15 | 15     from deepspeed.runtime.swap_tensor.partitioned_param_swapper import PartitionedParamStatus
16 | 16     from deepspeed.utils.debug import debug_module2name_id, debug_param2name_id
17 | 17     from deepspeed.accelerator import get_accelerator
18 | 18
19 | 18 + # my
20 | 19 + # from deepspeed.utils.debug import countt, module_index, my_print_params_info,
      | my_saveload_module_individually
21 | 20 + import sys
22 | 21 + sys.path.append('/shared_ssd_storage/shilonglei/00C/a_MoE_experiments')
23 | 22 + from my_debug_utils import countt, module_index, my_print_params_info,
      | my_saveload_module_individually, forward_prehook_time_output
24 | 23
25 | 24 def debug_rank0(message: str) -> None:
26 | 25     if dist.get_rank() == 0:
27 | 26
28 | 27
29 | 28
30 | 29
31 | 30
32 | 31
33 | 32
34 | 33
35 | 34
36 | 35
37 | 36
38 | 37
39 | 38
40 | 39
41 | 40
42 | 41
43 | 42
44 | 43
45 | 44
46 | 45
47 | 46
48 | 47
49 | 48
50 | 49
51 | 50
52 | 51
53 | 52
54 | 53
55 | 54
56 | 55
57 | 56
58 | 57
59 | 58
60 | 59
61 | 60
62 | 61
63 | 62
64 | 63
65 | 64
66 | 65
67 | 66
68 | 67
69 | 68
70 | 69
71 | 70
72 | 71
73 | 72
74 | 73
75 | 74
76 | 75
77 | 76
78 | 77
79 | 78
80 | 79
81 | 80
82 | 81
83 | 82
84 | 83
85 | 84
86 | 85
87 | 86
88 | 87
89 | 88
90 | 89
91 | 90
92 | 91
93 | 92
94 | 93
95 | 94
96 | 95
97 | 96
98 | 97
99 | 98
100 | 99
101 | 100
102 | 101
103 | 102
104 | 103
105 | 104
106 | 105
107 | 106
108 | 107
109 | 108
110 | 109
111 | 110
112 | 111
113 | 112
114 | 113
115 | 114
116 | 115
117 | 116
118 | 117
119 | 118
120 | 119
121 | 120
122 | 121
123 | 122
124 | 123
125 | 124
126 | 125
127 | 126
128 | 127
129 | 128
130 | 129
131 | 130
132 | 131
133 | 132
134 | 133
135 | 134
136 | 135
137 | 136
138 | 137
139 | 138
140 | 139
141 | 140
142 | 141
143 | 142
144 | 143
145 | 144
146 | 145
147 | 146
148 | 147
149 | 148
150 | 149
151 | 150
152 | 151
153 | 152
154 | 153
155 | 154
156 | 155
157 | 156
158 | 157
159 | 158
160 | 159
161 | 160
162 | 161
163 | 162
164 | 163
165 | 164
166 | 165
167 | 166
168 | 167
169 | 168
170 | 169
171 | 170
172 | 171
173 | 172
174 | 173
175 | 174
176 | 175
177 | 176
178 | 177
179 | 178
180 | 179
181 | 180
182 | 181
183 | 182
184 | 183
185 | 184
186 | 185
187 | 186
188 | 187
189 | 188
190 | 189
191 | 190
192 | 191
193 | 192
194 | 193
195 | 194
196 | 195
197 | 196
198 | 197
199 | 198
200 | 199
201 | 200
202 | 201
203 | 202
204 | 203
205 | 204
206 | 205
207 | 206
208 | 207
209 | 208
210 | 209
211 | 210
212 | 211
213 | 212
214 | 213
215 | 214
216 | 215
217 | 216
218 | 217
219 | 218
220 | 219
221 | 220
222 | 221
223 | 222
224 | 223
225 | 224
226 | 225
227 | 226
228 | 227
229 | 228
230 | 229
231 | 230
232 | 231
233 | 232
234 | 233
235 | 234
236 | 235
237 | 236
238 | 237
239 | 238
240 | 239
241 | 240
242 | 241
243 | 242
244 | 243
245 | 244
246 | 245
247 | 246
248 | 247
249 | 248
250 | 249
251 | 250
252 | 251
253 | 252
254 | 253
255 | 254
256 | 255
257 | 256
258 | 257
259 | 258
260 | 259
261 | 260
262 | 261
263 | 262
264 | 263
265 | 264
266 | 265
267 | 266
268 | 267
269 | 268
270 | 269
271 | 270
272 | 271
273 | 272
274 | 273
275 | 274
276 | 275
277 | 276
278 | 277
279 | 278
280 | 279
281 | 280
282 | 281
283 | 282
284 | 283
285 | 284
286 | 285
287 | 286
288 | 287
289 | 288
290 | 289
291 | 290
292 | 291
293 | 292
294 | 293
295 | 294
296 | 295
297 | 296
298 | 297
299 | 298
300 | 299
301 | 300
302 | 301
303 | 302
304 | 303
305 | 304
306 | 305
307 | 306
308 | 307
309 | 308
310 | 309
311 | 310
312 | 311
313 | 312
314 | 313
315 | 314
316 | 315
317 | 316
318 | 317
319 | 318
320 | 319
321 | 320
322 | 321
323 | 322
324 | 323
325 | 324
326 | 325
327 | 326
328 | 327
329 | 328
330 | 329
331 | 330
332 | 331
333 | 332
334 | 333
335 | 334
336 | 335
337 | 336
338 | 337
339 | 338
340 | 339
341 | 340
342 | 341
343 | 342
344 | 343
345 | 344
346 | 345
347 | 346
348 | 347
349 | 348
350 | 349
351 | 350
352 | 351
353 | 352
354 | 353
355 | 354
356 | 355
357 | 356
358 | 357
359 | 358
360 | 359
361 | 360
362 | 361
363 | 362
364 | 363
365 | 364
366 | 365
367 | 366
368 | 367
369 | 368
370 | 369
371 | 370
372 | 371
373 | 372
374 | 373
375 | 374
376 | 375
377 | 376
378 | 377
379 | 378
380 | 379
381 | 380
382 | 381
383 | 382
384 | 383
385 | 384
386 | 385
387 | 386
388 | 387
389 | 388
390 | 389
391 | 390
392 | 391
393 | 392
394 | 393
395 | 394
396 | 395
397 | 396
398 | 397
399 | 398
400 | 399
401 | 400
402 | 401
403 | 402
404 | 403
405 | 404
406 | 405
407 | 406
408 | 407
409 | 408
410 | 409
411 | 410
412 | 411
413 | 412
414 | 413
415 | 414
416 | 415
417 | 416
418 | 417
419 | 418
420 | 419
421 | 420
422 | 421
423 | 422
424 | 423
425 | 424
426 | 425
427 | 426
428 | 427
429 | 428
430 | 429
431 | 430
432 | 431
433 | 432
434 | 433
435 | 434
436 | 435
437 | 436
438 | 437
439 | 438
440 | 439
441 | 440
442 | 441
443 | 442
444 | 443
445 | 444
446 | 445
447 | 446
448 | 447
449 | 448
450 | 449
451 | 450
452 | 451
453 | 452
454 | 453
455 | 454
456 | 455
457 | 456
458 | 457
459 | 458
460 | 459
461 | 460
462 | 461
463 | 462
464 | 463
465 | 464
466 | 465
467 | 466
468 | 467
469 | 468
470 | 469
471 | 470
472 | 471
473 | 472
474 | 473
475 | 474
476 | 475
477 | 476
478 | 477
479 | 478
480 | 479
481 | 480
482 | 481
483 | 482
484 | 483
485 | 484
486 | 485
487 | 486
488 | 487
489 | 488
490 | 489
491 | 490
492 | 491
493 | 492
494 | 493
495 | 494
496 | 495
497 | 496
498 | 497
499 | 498
500 | 499
501 | 500
502 | 501
503 | 502
504 | 503
505 | 504
506 | 505
507 | 506
508 | 507
509 | 508
510 | 509
511 | 510
512 | 511
513 | 512
514 | 513
515 | 514
516 | 515
517 | 516
518 | 517
519 | 518
520 | 519
521 | 520
522 | 521
523 | 522
524 | 523
525 | 524
526 | 525
527 | 526
528 | 527
529 | 528
530 | 529
531 | 530
532 | 531
533 | 532
534 | 533
535 | 534
536 | 535
537 | 536
538 | 537
539 | 538
540 | 539
541 | 540
542 | 541
543 | 542
544 | 543
545 | 544
546 | 545
547 | 546
548 | 547
549 | 548
550 | 549
551 | 550
552 | 551
553 | 552
554 | 553
555 | 554
556 | 555
557 | 556
558 | 557
559 | 558
560 | 559
561 | 560
562 | 561
563 | 562
564 | 563
565 | 564
566 | 565
567 | 566
568 | 567
569 | 568
570 | 569
571 | 570
572 | 571
573 | 572
574 | 573
575 | 574
576 | 575
577 | 576
578 | 577
579 | 578
580 | 579
581 | 580
582 | 581
583 | 582
584 | 583
585 | 584
586 | 585
587 | 586
588 | 587
589 | 588
590 | 589
591 | 590
592 | 591
593 | 592
594 | 593
595 | 594
596 | 595
597 | 596
598 | 597
599 | 598
600 | 599
601 | 600
602 | 601
603 | 602
604 | 603
605 | 604
606 | 605
607 | 606
608 | 607
609 | 608
610 | 609
611 | 610
612 | 611
613 | 612
614 | 613
615 | 614
616 | 615
617 | 616
618 | 617
619 | 618
620 | 619
621 | 620
622 | 621
623 | 622
624 | 623
625 | 624
626 | 625
627 | 626
628 | 627
629 | 628
630 | 629
631 | 630
632 | 631
633 | 632
634 | 633
635 | 634
636 | 635
637 | 636
638 | 637
639 | 638
640 | 639
641 | 640
642 | 641
643 | 642
644 | 643
645 | 644
646 | 645
647 | 646
648 | 647
649 | 648
650 | 649
651 | 650
652 | 651
653 | 652
654 | 653
655 | 654
656 | 655
657 | 656
658 | 657
659 | 658
660 | 659
661 | 660
662 | 661
663 | 662
664 | 663
665 | 664
666 | 665
667 | 666
668 | 667
669 | 668
670 | 669
671 | 670
672 | 671
673 | 672
674 | 673
675 | 674
676 | 675
677 | 676
678 | 677
679 | 678
680 | 679
681 | 680
682 | 681
683 | 682
684 | 683
685 | 684
686 | 685
687 | 686
688 | 687
689 | 688
690 | 689
691 | 690
692 | 691
693 | 692
694 | 693
695 | 694
696 | 695
697 | 696
698 | 697
699 | 698
700 | 699
701 | 700
702 | 701
703 | 702
704 | 703
705 | 704
706 | 705
707 | 706
708 | 707
709 | 708
710 | 709
711 | 710
712 | 711
713 | 712
714 | 713
715 | 714
716 | 715
717 | 716
718 | 717
719 | 718
720 | 719
721 | 720
722 | 721
723 | 722
724 | 723
725 | 724
726 | 725
727 | 726
728 | 727
729 | 728
730 | 729
731 | 730
732 | 731
733 | 732
734 | 733
735 | 734
736 | 735
737 | 736
738 | 737
739 | 738
740 | 739
741 | 740
742 | 741
743 | 742
744 | 743
745 | 744
746 | 745
747 | 746
748 | 747
749 | 748
750 | 749
751 | 750
752 | 751
753 | 752
754 | 753
755 | 754
756 | 755
757 | 756
758 | 757
759 | 758
760 | 759
761 | 760
762 | 761
763 | 762
764 | 763
765 | 764
766 | 765
767 | 766
768 | 767
769 | 768
770 | 769
771 | 770
772 | 771
773 | 772
774 | 773
775 | 774
776 | 775
777 | 776
778 | 777
779 | 778
780 | 779
781 | 780
782 | 781
783 | 782
784 | 783
785 | 784
786 | 785
787 | 786
788 | 787
789 | 788
790 | 789
791 | 790
792 | 791
793 | 792
794 | 793
795 | 794
796 | 795
797 | 796
798 | 797
799 | 798
800 | 799
801 | 800
802 | 801
803 | 802
804 | 803
805 | 804
806 | 805
807 | 806
808 | 807
809 | 808
810 | 809
811 | 810
812 | 811
813 | 812
814 | 813
815 | 814
816 | 815
817 | 816
818 | 817
819 | 818
820 | 819
821 | 820
822 | 821
823 | 822
824 | 823
825 | 824
826 | 825
827 | 826
828 | 827
829 | 828
830 | 829
831 | 830
832 | 831
833 | 832
834 | 833
835 | 834
836 | 835
837 | 836
838 | 837
839 | 838
840 | 839
841 | 840
842 | 841
843 | 842
844 | 843
845 | 844
846 | 845
847 | 846
848 | 847
849 | 848
850 | 849
851 | 850
852 | 851
853 | 852
854 | 853
855 | 854
856 | 855
857 | 856
858 | 857
859 | 858
860 | 859
861 | 860
862 | 861
863 | 862
864 | 863
865 | 864
866 | 865
867 | 866
868 | 867
869 | 868
870 | 869
871 | 870
872 | 871
873 | 872
874 | 873
875 | 874
876 | 875
877 | 876
878 | 877
879 | 878
880 | 879
881 | 880
882 | 881
883 | 882
884 | 883
885 | 884
886 | 885
887 | 886
888 | 887
889 | 888
890 | 889
891 | 890
892 | 891
893 | 892
894 | 893
895 | 894
896 | 895
897 | 896
898 | 897
899 | 898
900 | 899
901 | 900
902 | 901
903 | 902
904 | 903
905 | 904
906 | 905
907 | 906
908 | 907
909 | 908
910 | 909
911 | 910
912 | 911
913 | 912
914 | 913
915 | 914
916 | 915
917 | 916
918 | 917
919 | 918
920 | 919
921 | 920
922 | 921
923 | 922
924 | 923
925 | 924
926 | 925
927 | 926
928 | 927
929 | 928
930 | 929
931 | 930
932 | 931
933 | 932
934 | 933
935 | 934
936 | 935
937 | 936
938 | 937
939 | 938
940 | 939
941 | 940
942 | 941
943 | 942
944 | 943
945 | 944
946 | 945
947 | 946
948 | 947
949 | 948
950 | 949
951 | 950
952 | 951
953 | 952
954 | 953
955 | 954
956 | 955
957 | 956
958 | 957
959 | 958
960 | 959
961 | 960
962 | 961
963 | 962
964 | 963
965 | 964
966 | 965
967 | 966
968 | 967
969 | 968
970 | 969
971 | 970
972 | 971
973 | 972
974 | 973
975 | 974
976 | 975
977 | 976
978 | 977
979 | 978
980 | 979
981 | 980
982 | 981
983 | 982
984 | 983
985 | 984
986 | 985
987 | 986
988 | 987
989 | 988
990 | 989
991 | 990
992 | 991
993 | 992
994 | 993
995 | 994
996 | 995
997 | 996
998 | 997
999 | 998
1000 | 999
1001 | 1000
1002 | 1001
1003 | 1002
1004 | 1003
1005 | 1004
1006 | 1005
1007 | 1006
1008 | 1007
1009 | 1008
1010 | 1009
1011 | 1010
1012 | 1011
1013 | 1012
1014 | 1013
1015 | 1014
1016 | 1015
1017 | 1016
1018 | 1017
1019 | 1018
1020 | 1019
1021 | 1020
1022 | 1021
1023 | 1022
1024 | 1023
1025 | 1024
1026 | 1025
1027 | 1026
1028 | 1027
1029 | 1028
1030 | 1029
1031 | 1030
1032 | 1031
1033 | 1032
1034 | 1033
1035 | 1034
1036 | 1035
1037 | 1036
1038 | 1037
1039 | 1038
1040 | 1039
1041 | 1040
1042 | 1041
1043 | 1042
1044 | 1043
1045 | 1044
1046 | 1045
1047 | 1046
1048 | 1047
1049 | 1048
1050 | 1049
1051 | 1050
1052 | 1051
1053 | 1052
1054 | 1053
1055 | 1054
1056 | 1055
1057 | 1056
1058 | 1057
1059 | 1058
1060 | 1059
1061 | 1060
1062 | 1061
1063 | 1062
1064 | 1063
1065 | 1064
1066 | 1065
1067 | 1066
1068 | 1067
1069 | 1068
1070 | 1069
1071 | 1070
1072 | 1071
1073 | 1072
1074 | 1073
1075 | 1074
1076 | 1075
1077 | 1076
1078 | 1077
1079 | 1078
1080 | 1079
1081 | 1080
1082 | 1081
1083 | 1082
1084 | 1083
1085 | 1084
1086 | 1085
1087 | 1086
1088 | 1087
1089 | 1088
1090 | 1089
1091 | 1090
1092 | 1091
1093 | 1092
1094 | 1093
1095 | 1094
1096 | 1095
1097 | 1096
1098 | 1097
1099 | 1098
1100 | 1099
1101 | 1100
1102 | 1101
1103 | 1102
1104 | 1103
1105 | 1104
1106 | 1105
1107 | 1106
1108 | 1107
1109 | 1108
1110 | 1109
1111 | 1110
1112 | 1111
1113 | 1112
1114 | 1113
1115 | 1114
1116 | 1115
1117 | 1116
1118 | 1117
1119 | 1118
1120 | 1119
1121 | 1120
1122 | 1121
1123 | 1122
1124 | 1123
1125 | 1124
1126 | 1125
1127 | 1126
1128 | 1127
1129 | 1128
1130 | 1129
1131 | 1130
1132 | 1131
1133 | 1132
1134 | 1133
1135 | 1134
1136 | 1135
1137 | 1136
1138 | 1137
1139 | 1138
1140 | 1139
1141 | 1140
1142 | 1141
1143 | 1142
1144 | 1143
1145 | 1144
1146 | 1145
1147 | 1146
1148 | 1147
1149 | 1148
1150 | 1149
1151 | 1150
1152 | 1151
1153 | 1152
1154 | 1153
1155 | 1154
1156 | 1155
1157 | 1156
1158 | 1157
1159 | 1158
1160 | 1159
1161 | 1160
1162 | 1161
1163 | 1162
1164 | 1163
1165 | 1164
1166 | 1165
1167 | 1166
1168 | 1167
1169 | 1168
1170 | 1169
1171 | 1170
1172 | 1171
1173 | 1172
1174 | 1173
1175 | 1174
1176 | 1175
1177 | 1176
1178 | 1177
1179 | 1178
1180 | 1179
1181 | 1180
1182 | 1181
1183 | 1182
1184 | 1183
1185 | 1184
1186 | 1185
1187 | 1186
1188 | 1187
1189 | 1188
1190 | 1189
1191 | 1190
1192 | 1191
1193 | 1192
1194 | 1193
1195 | 1194
1196 | 1195
1197 | 1196
1198 | 1197
1199 | 1198
1200 | 1199
1201 | 1200
1202 | 1201
1203 | 1202
1204 | 1203
1205 | 1204
1206 | 1205
1207 | 1206
1208 | 1207
1209 | 1208
1210 | 1209
1211 | 1210
1212 | 1211
1213 | 1212
1214 | 1213
1215 | 1214
1216 | 1215
1217 | 1216
1218 | 1217
1219 | 1218
1220 | 1219
1221 | 1220
1222 | 1221
1223 | 1222
1224 | 1223
1225 | 1224
1226 | 1225
1227 | 1226
1228 | 1227
1229 | 1228
1230 | 1229
1231 | 1230
1232 | 1231
1233 | 1232
1234 | 1233
1235 | 1234
1236 | 1235
1237 | 1236
1238 | 1237
1239 | 1238
1240 | 1239
1241 | 1240
1242 | 1241
1243 | 1242
1244 | 1243
1245 | 1244
1246 | 1245
1247 | 1246
1248 | 1247
1249 | 1248
1250 | 1249
1251 | 1250
1252 | 1251
1253 | 1252
1254 | 1253
1255 | 1254
1256 | 1255
1257 | 1256
1258 | 1257
1259 | 1258
1260 | 1259
1261 | 1260
1262 | 1261
1263 | 1262
1264 | 1263
1265 | 1264
1266 | 1265
1267 | 1266
1268 | 1267
1269 | 1268
1270 | 1269
1271 | 1270
1272 | 1271
1273 | 1272
1274 | 1273
1275 | 1274
1276 | 1275
1277 | 1276
1278 | 1277
1279 | 1278
1280 | 1279
1281
```

```

272 + torch.cuda.nvtx.range_push(f"step[{self.__step_id}]:Fetch current module")
247 273 for param in params_to_fetch:
248 274     debug_rank0(f"-fetch: {param.ds_summary()}")
275 +
249 276 self.__all_gather_params(params_to_fetch)
250 277
251 278 # wait for parameters in the immediately needed submodule to become available
259 286     if len(self.__ongoing_fetch_events) > self.__max_ongoing_fetch_events:
260 287         self.__ongoing_fetch_events.popleft().synchronize()
261 288
262 289 - self.__inflight_param_registry.pop(param).wait()
289 + self.__inflight_param_registry.pop(param).wait() #
current_stream().synchronize()
263 290
264 291 - event = get_accelerator().Event()
291 + event = get_accelerator().Event() # torch.cuda.Event
265 292     event.record()
266 293     self.__ongoing_fetch_events.append(event)
267 294
268 295     assert param.ds_status == ZeroParamStatus.AVAILABLE, param.ds_summary()
269 296     get_accelerator().current_stream().wait_stream(self.__allgather_stream)
297 + torch.cuda.nvtx.range_pop()
298 +
299 + # ## print Embedding weights into .txt
300 + # my_print_params_info('paramsEmbedding_afterGather_withoutPrefetch_skip_1.txt',
"Embedding", current_submodule)
301 + # ## print LayerNorm weights into .txt
302 + # my_print_params_info('paramsLayerNorm_afterGather_withoutPrefetch_skip_1.txt',
"T5LayerNorm", current_submodule)
303 +
304 + ## -----
305 + # save/load T5LayerNorm and Embedding weights
306 + from my_debug_utils import my_skip_2_enabled
307 + if (my_skip_2_enabled == True):
308 +     my_saveload_module_individually(current_submodule, 'load', print=False)
309 + ## -----
310 + # # print weights into .txt
311 + # import os
312 + # threshold_size = 1000 * 1024
313 + # filename = '/home/mark/Research/a_MoE_experiments/weights_skip_1_3200.txt'
314 + # with open(filename, 'a+') as f:
315 + #     # file_size = os.path.getsize(filename)
316 + #     # if file_size < threshold_size:
317 + #     if module_index < 3320:
318 + #         f.write("-----")
319 + #         f.write(f"Name[{current_submodule.__class__.__name__}][{module_index}]\n")
320 + #         f.write(f"Name[{current_submodule.state_dict()}]\n\n")
321 +
322 + module_index = module_index + 1
323 + ## -----
270 324
271 325 # kick off parameter prefetches for upcoming modules
272 326 # don't prefetch if we dont have a completed model trace
273 327 if self.is_complete_trace():
274 328 + torch.cuda.nvtx.range_push(f"step[{self.__step_id}]:Prefetch current module")
275 329     # go through the parameters we need for the current module and pop them
276 330     # off the fetch queue so that they aren't prefetched later.
277 331     # if params have already been popped off the fetch queue by earlier
333 388     if self.__prefetch_nvme:

```

```

334 389         self.__prefetch_nvme_param_partitions()
335 390
391 +         torch.cuda.nvtx.range_pop()
392 +
336 393         self.__step_id += 1
337 394
338 395         @instrument_w_nvtx
375 432         self.__n_available_params += param.ds_numel
376 433
377 434         if partitioned_params:
378 -             with get_accelerator().stream(self.__allgather_stream):
379 -                 handle = partitioned_params[0].all_gather_coalesced(partitioned_params)
435 +             with get_accelerator().stream(self.__allgather_stream): #ACUTALLY: with
torch.cuda.stream(torch.cuda.Stream object instance):
436 +                 handle = partitioned_params[0].all_gather_coalesced(partitioned_params) # SEARCH
`def _convert_to_deepspeed_param`
437 +                 # SEARCH: def all_gather_coalesced(params: Iterable[Parameter], safe_mode: bool
= False) -> AllGatherCoalescedHandle:
380 438
381 439         for param in partitioned_params:
382 440             assert param.ds_status == ZeroParamStatus.INFLIGHT, param.ds_summary()
387 445             p for p in partitioned_params if p.ds_persist and p.ds_tensor.final_location ==
OffloadDeviceEnum.nvme
388 446             ]
389 447             if swap_persisted_params:
390 -
- swap_persisted_params[0].nvme_swapper.remove_partition_and_release_buffers(swap_persisted_params
)
448 +
swap_persisted_params[0].nvme_swapper.remove_partition_and_release_buffers(swap_persisted_params
) #SEARCH def remove_partition_and_release_buffers(self, params):
391 449
392 450         @instrument_w_nvtx
393 451         def __release_param(self, param: Parameter) -> None:

```

1 deepspeed/utils/debug.py

```

11 11     param_names = {}
12 12
13 13
14 14 +
14 15     def debug_extract_module_and_param_names(model):
15 16         # extract the fully qualified names as soon as the model is acquired
16 17         global module_names

```

11 deepspeed/utils/nvtx.py

```

4 4     # DeepSpeed Team
5 5
6 6     from deepspeed.accelerator import get_accelerator
7 7 + from torch.cuda import nvtx
7 8
8 9
9 10     def instrument_w_nvtx(func):
17 18         return ret_val
18 19
19 20         return wrapped_fn
21 21 +

```

```
22 + def my_nvtx_wrapper(func):
23 +
24 +     def wrapped_fn(*args, **kwargs):
25 +         get_accelerator().range_push(func.__name__)
26 +         ret_val = func(*args, **kwargs)
27 +         get_accelerator().range_pop()
28 +         return ret_val
29 +
30 +     return wrapped_fn
```