Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: microsoft/DeepSpeed | base: master | ... | head repository: Markshilong/DeepSpeed | compare: master

✕ Can't automatically merge. Don't worry, you can still create the pull request.

Discuss and review the changes in this comparison with others. Learn about pull requests

Create pull request

15 commits **12** files changed **2** contributors

Commits on May 15, 2023

a
 **Markshilong** committed on May 15

Commits on May 25, 2023

save
 **Markshilong** committed on May 25

Commits on May 31, 2023

update
 **Markshilong** committed on May 31

Commits on Jun 1, 2023

update
 **Markshilong** committed on Jun 1

Commits on Jun 7, 2023

my_update
 **Markshilong** committed on Jun 7

Commits on Jun 11, 2023

update
 **Markshilong** committed on Jun 11

update

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

27 deepspeed/runtime/pipe/engine.py

```
4     4   # DeepSpeed Team
5     5
6     6   from types import MethodType
7           -
      7   + from deepspeed.utils.nvtx import my_nvtx_wrapper, my_nvtx_wrapper
      8   + from torch.cuda import nvtx
8     9   import torch
9     10  from deepspeed import comm as dist
10    11
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          -
58          -         assert self.zero_optimization_stage() < 2, "ZeRO-2 and ZeRO-3 are incompatible with
      pipeline parallelism"
      58  +         # my comment:
      59  +         print("!!!!! DISABLE: ZeRO-2 and ZeRO-3 are incompatible with pipeline parallelism")
      60  +         # assert self.zero_optimization_stage() < 2, "ZeRO-2 and ZeRO-3 are incompatible with
      pipeline parallelism"
59    61
60    62          # We schedule the all-reduces, so disable it in super().backward()
61    63          self.enable_backward_allreduce = False
222   224         pipe_dataloader = RepeatingLoader(pipe_dataloader)
223   225         self.set_dataloader(pipe_dataloader)
224   226
      227 +     @my_nvtx_wrapper
225   228       def _exec_reduce_tied_grads(self):
      229 +
226   230         # We need to run this first to write to self.averaged_gradients;
227   231         # since this class turns `enable_backward_allreduce` off,
228   232         # `self.overlapping_partition_gradients_reduce_epilogue()` defined in the
      DeepSpeedEngine
239   243             grad = weight._hp_grad if self.bfloat16_enabled() else weight.grad
240   244             dist.all_reduce(grad, group=group)
241   245
      246 +     @my_nvtx_wrapper
242   247       def _exec_reduce_grads(self):
243   248         self._force_grad_boundary = True
244   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 +         # nvtx.range_push("Execute_schedule")
336   342         self._exec_schedule(sched)
      343 +         # nvtx.range_pop()
337   344         self.agg_train_loss = self._aggregate_total_loss()
338   345
339   346         self.timers('train_batch').stop()
360   367         # TODO: should return precisely what loss returned and allow others to be queried?
361   368         return self.agg_train_loss
362   369
```

```
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_recv_activations(self, buffer_id):
1006 1021          if self.wall_clock_breakdown():
1007 1022              self.timers('pipe_recv_input').start()
1045 1060          if self.wall_clock_breakdown():
1046 1061              self.timers('pipe_recv_input').stop()
1047 1062
1063 +     @my_nvtx_wrapper
1048 1064      def _exec_recv_grads(self, buffer_id):
1049 1065          if self.wall_clock_breakdown():
1050 1066              self.timers('pipe_recv_grad').start()
1102 1118          if self.wall_clock_breakdown():
1103 1119              self.timers('pipe_recv_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 +    @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)
     1324 +                # nvtx.range_push(f"{self._exec_instr.__name__}")
1307 1325                  self._exec_instr(**cmd.kwargs)
     1326 +                # nvtx.range_pop()
```

2 deepspeed/runtime/pipe/p2p.py

```
61 61      else:
62 62
63 63          if can_send_recv():
64      -            return dist.send(tensor, dest_rank)
   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      -    def __init__(self, ds_config, model_dtype):
   38 +    def __init__(self, ds_config, model_dtype, my_version):
   39 +
   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 +        if self.my_version:
   80 +            self.finished_flag = False
   81 +        else:
   82 +            self.finished_flag = True
   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)
          96  +
          97  +         if (self.my_version == False):
          98  +             shutil.rmtree(self.swap_folder, ignore_errors=True)
          99  +             os.makedirs(self.swap_folder, exist_ok=True)
91        100
92        101           self.swap_element_size = torch.tensor([], dtype=self.dtype).element_size()
93        102
178       187               print_rank_0(f"param {param.ds_id} is assigned swap in buffer id {buffer_id}  ")
179       188               self.param_id_to_buffer_id[param_id] = buffer_id
180       189               aligned_swap_numel = self._io_aligned_numel(self.param_id_to_numel[param_id])
181       -                 swap_buffer = self.buffers.narrow(0, int(buffer_id *
          self.aligned_elements_per_buffer), aligned_swap_numel)
          190 +             swap_buffer = self.buffers.narrow(0, int(buffer_id *
          self.aligned_elements_per_buffer), aligned_swap_numel) # torch.Tensor.narrow(dim, start, length)
          → Tensor
182       191
183       192               self.param_id_to_swap_buffer[param_id] = swap_buffer
184       193               compute_buffer = swap_buffer.narrow(0, 0, self.param_id_to_numel[param_id])
191       200       def synchronize_writes(self):
192       201           if self.pending_writes == 0:
193       202               return
194       -         assert self.pending_writes == self.aio_write_handle.wait()
          203 +         # original
          204 +         # assert self.pending_writes == self.aio_write_handle.wait()
          205 +
          206 +         # my start
          207 +         if (self.my_version):
          208 +             if self.finished_flag == True:
          209 +                 assert self.pending_writes == self.aio_write_handle.wait()
          210 +         else:
          211 +             assert self.pending_writes == self.aio_write_handle.wait()
          212 +         # my end
          213 +
195       214           self.pending_writes = 0
196       215           self.remove_partition_and_release_buffers(self.swap_out_params)
197       216           self.swap_out_params = []
201       220           if self.pending_reads == 0:
202       221               return
203       222
204       -         assert self.pending_reads == self.aio_read_handle.wait()
          223 +         assert self.pending_reads == self.aio_read_handle.wait()  #SEARCH: int
          deepspeed_aio_handle_t::wait()
205       224
206       225           self.pending_reads = 0
207       226
249       268           swap_out_params = self._get_swap_buffers(params)
250       269           self._track_numel(params)
251       270
252       -         swap_out_tensors(self.aio_write_handle, swap_out_params, swap_out_paths)
          271 +         # my start
          272 +         if (self.my_version):
          273 +             swap_out_tensors(self.aio_write_handle, swap_out_params, swap_out_paths,
          self.finished_flag)
          274 +         else:
          275 +             swap_out_tensors(self.aio_write_handle, swap_out_params, swap_out_paths, True)
          276 +         # my end
          277 +
```

```
278 +         # original
279 +         # swap_out_tensors(self.aio_write_handle, swap_out_params, swap_out_paths, True)
280
281           self.pending_writes += len(swap_out_params)
282           self.swap_out_params += params
334           self._update_inflight_swap_in(params, swap_in_buffers, inflight_numel)
335
336           if not async_op:
-                 self.synchronize_reads()
337 +               self.synchronize_reads() # lsl: After this, all self.pending_reads == 0, copy from buffer to param.ds_tensor.data
338
339       # Enables swapping into buffer that is out the control of swapper. This is always synchronous
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 def swap_out_tensors(swap_handle, tensor_buffers, swap_paths):
+   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):
-                assert (swap_handle.async_pwrite(buffer, path) == 0)
26 +         # my next line is mine
27 +         if finished_flag == True:
28 +             assert (swap_handle.async_pwrite(buffer, path) == 0)
29  29
30  30
31  31   def print_object(obj, name, exclude_list=[]):
```

8 deepspeed/runtime/zero/parameter_offload.py

```
246 246
247 247           self.forward_hooks = []
248 248           self.backward_hooks = []
-   249           self.setup_zero_stage3_hooks()
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           if non_zero_params:
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               if zero_params:
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               else:
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

```
@@ -40,7 +40,7 @@
 class NoGatherHandle:

+    @instrument_w_nvtx
     def __init__(self, param: Parameter) -> None:
         if param.ds_status != ZeroParamStatus.INFLIGHT:
             raise RuntimeError(f"expected param {param.ds_summary()} to be available")
-
+
         param.data = param.ds_tensor.data.to(device=get_accelerator().current_device_name(),
                                              non_blocking=True).view(param.ds_shape)
         self.__param = param

+    @instrument_w_nvtx
     def wait(self) -> None:
         get_accelerator().current_stream().synchronize()
         self.__param.ds_status = ZeroParamStatus.AVAILABLE
@@ -287,7 +289,13 @@ class InsertPostInitMethodToModuleSubClasses(object):

     def __init__(self, enabled=True, mem_efficient_linear=True, ds_config=None, dtype=None):
+        import sys
+        sys.path.append('/home/mark/Research/a_MoE_experiments/my_debug_utils')
+        from my_debug_utils import my_skip_1_enabled
+        self.my_version = my_skip_1_enabled  # True to skip  stage 1 (write to NVMe .swp)
+        self.my_print = True
+
         self.mem_efficient_linear = mem_efficient_linear
         self.enabled = enabled
         self._set_dtype(ds_config, dtype)
         self.wrapped_cls = set()

     def __enter__(self):
+        if(self.my_print):print("!!!!!!!! executing Init.fatherClass - __enter__()!!!!!!")
         if not self.enabled:
             return

         zero_init_context.append(self)

     def __exit__(self, exc_type, exc_value, traceback):
+        if(self.my_print):print("!!!!!!!! executing Init.fatherClass - __exit__()!!!!!!")
         if not self.enabled:
             return

         zero_init_context.pop()
         if self.nest_level == 0:
             if dist.get_rank() == 0:
+                if(self.my_version):self.set_finished_flag_True()
                 logger.info("finished initializing model with %.2fB parameters", param_count / 1e9)
-
         # Now that we cleaned up the metaclass injection, raise the exception.
         if exc_type is not None:
             return False

     # To be implemented by inheriting classes
     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       -
       838  +              # if get_accelerator().on_accelerator(param):
       839  +              #     # lslmark: for Pipeline, temporarily comment
       840  +              #     dist.broadcast(param, 0, self.get_dp_process_group())
       841  +              # else:
       842  +              #     if dist.get_rank() == 0:
       843  +              #         logger.warn(f"param `{name}` in {module.__class__.__name__} "
       844  +              #                     f"not on GPU so was not broadcasted from rank 0")
       845  +
822  846                  param.partition()
823  847          see_memory_usage(
824  848              f"Param count {param_count}. After converting and partitioning parmas in
                      {module.__class__.__name__}",
1035 1059      def _partition_numel(self, param):
1036 1060          return param.ds_tensor.ds_numel
1037 1061
       1062  +    @instrument_w_nvtx
1038 1063      def _ensure_availability_of_partitioned_params(self, params):
1039 1064          swap_in_list = []
1040 1065          swap_in_flight = []
1046 1071                  assert param.ds_tensor.final_location == OffloadDeviceEnum.nvme and
                              param.ds_status == ZeroParamStatus.NOT_AVAILABLE
1047 1072                  swap_in_flight.append(param)
1048 1073          if len(swap_in_list) > 0:
1049       -            swap_in_list[0].nvme_swapper.swap_in(swap_in_list, async_op=False)
       1074  +            swap_in_list[0].nvme_swapper.swap_in(swap_in_list, async_op=False) # SEARCH: def
                          swap_in(self, params, async_op=True, swap_in_buffers=None):
1050 1075          elif len(swap_in_flight) > 0:
1051       -            swap_in_flight[0].nvme_swapper.synchronize_reads()
       1076  +            swap_in_flight[0].nvme_swapper.synchronize_reads() # SEARCH: def
                          synchronize_reads(self):
1052 1077
1053 1078      @instrument_w_nvtx
1054 1079      def _all_gather(self, param_list, async_op=False, hierarchy=None):
1112 1137              #    print(f"Releasing {param.data.numel()}")
1113 1138              if param.ds_tensor is not None and not has_been_updated:
1114 1139
       1140  +
1115 1141                  #param.data = param.ds_tensor.data
1116 1142
1117 1143                  see_memory_usage(f'Before partitioning param {param.ds_id} {param.shape}',
                              force=False)
1160 1186              start = partition_size * self.get_partition_rank()
1161 1187              end = start + partition_size
1162 1188
```

```
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      1191              if start < param.ds_numel and end <= param.ds_numel:
1166      1192                  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      1685          for h in handles:
1660      1686              h.wait()
          1687  +
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    + # my
          19    + # from deepspeed.utils.debug import countt, module_index, my_print_params_info, my_saveload_module_individually
          20    + import sys
          21    + sys.path.append('/shared_ssd_storage/shilonglei/OOC/a_MoE_experiments')
          22    + from my_debug_utils import countt, module_index, my_print_params_info, my_saveload_module_individually, forward_prehook_time_output
19        23
20        24    def debug_rank0(message: str) -> None:
21        25        if dist.get_rank() == 0:
243       247
244       248            params_to_fetch = frozenset(iter_params(current_submodule))
245       249
          250   +        global module_index
          251   +        ## ----------------------------------
          252   +        # # # print weights into .txt
          253   +
          254   +        # import os
          255   +        # threshold_size = 1000 * 1024
          256   +        # filename = '/home/mark/Research/a_MoE_experiments/weights_before_hookGather_withoutPrefetch_ori_1_inputchange.txt'
          257   +        # with open(filename, 'a+') as f:
          258   +        #     # file_size = os.path.getsize(filename)
          259   +        #     if module_index < 3320:
          260   +        #         f.write("-------------------------------------")
          261   +        #         f.write(f"Name[{current_submodule.__class__.__name__}][{module_index}]\n")
          262   +        #         f.write(f"Name[{current_submodule.state_dict()}]\n\n")
          263   +        ## ----------------------------------
          264   +
          265   +        # ## print Embedding weights into .txt
          266   +        # my_print_params_info('paramsEmbedding_beforeGather_withoutPrefetch_skip_1.txt', "Embedding", current_submodule)
          267   +        # ## print LayerNorm weights into .txt
          268   +        # my_print_params_info('paramsLayerNorm_beforeGather_withoutPrefetch_skip_1.txt', "T5LayerNorm", current_submodule)
          269   +
246       270           # kick off all gather for params in the immediately required submodule
          271   +
```

```diff
+        torch.cuda.nvtx.range_push(f"step[{self.__step_id}]:Fetch current module")
         for param in params_to_fetch:
             debug_rank0(f"-fetch: {param.ds_summary()}")
+
         self.__all_gather_params(params_to_fetch)

         # wait for parameters in the immediately needed submodule to become available
                     if len(self.__ongoing_fetch_events) > self.__max_ongoing_fetch_events:
                         self.__ongoing_fetch_events.popleft().synchronize()

-                    self.__inflight_param_registry.pop(param).wait()
+                    self.__inflight_param_registry.pop(param).wait()  # current_stream().synchronize()

-                    event = get_accelerator().Event()
+                    event = get_accelerator().Event()  # torch.cuda.Event
                     event.record()
                     self.__ongoing_fetch_events.append(event)

             assert param.ds_status == ZeroParamStatus.AVAILABLE, param.ds_summary()
         get_accelerator().current_stream().wait_stream(self.__allgather_stream)
+        torch.cuda.nvtx.range_pop()
+
+        # ## print Embedding weights into .txt
+        # my_print_params_info('paramsEmbedding_afterGather_withoutPrefetch_skip_1.txt', "Embedding", current_submodule)
+        # ## print LayerNorm weights into .txt
+        # my_print_params_info('paramsLayerNorm_afterGather_withoutPrefetch_skip_1.txt', "T5LayerNorm", current_submodule)
+
+        ## ---------------------------------
+        # save/load T5LayerNorm and Embedding weights
+        from my_debug_utils import my_skip_2_enabled
+        if (my_skip_2_enabled == True):
+            my_saveload_module_individually(current_submodule, 'load', print=False)
+        ## ---------------------------------
+        # # print weights into .txt
+        # import os
+        # threshold_size = 1000 * 1024
+        # filename = '/home/mark/Research/a_MoE_experiments/weights_skip_1_3200.txt'
+        # with open(filename, 'a+') as f:
+        #     # file_size = os.path.getsize(filename)
+        #     # if file_size < threshold_size:
+        #     if module_index < 3320:
+        #         f.write("-------------------------------------")
+        #         f.write(f"Name[{current_submodule.__class__.__name__}][{module_index}]\n")
+        #         f.write(f"Name[{current_submodule.state_dict()}]\n\n")
+
+        module_index = module_index + 1
+        ## ---------------------------------

         # kick off parameter prefetches for upcoming modules
         # don't prefetch if we dont have a completed model trace
         if self.is_complete_trace():
+            torch.cuda.nvtx.range_push(f"step[{self.__step_id}]:PreFetch current module")
             # go through the parameters we need for the current module and pop them
             # off the fetch queue so that they aren't prefetched later.
             # if params have already been popped off the fetch queue by earlier
                 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 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 + 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 +
```

```
+ def my_nvtx_wrapper(func):
+
+     def wrapped_fn(*args, **kwargs):
+         get_accelerator().range_push(func.__name__)
+         ret_val = func(*args, **kwargs)
+         get_accelerator().range_pop()
+         return ret_val
+
+     return wrapped_fn
```