zcollection_trim_feature_request

April 21, 2023

```
[1]: import fsspec
     import numpy

     import dask.distributed
     import zcollection
     import zcollection.tests.data
```

```
[2]: zcollection.__version__
```

```
[2]: '2023.3.2'
```

Zcollection creation :

```
[4]: zds = next(zcollection.tests.data.create_test_dataset_with_fillvalue())
     fs = fsspec.filesystem('memory')

     partition_handler = zcollection.partitioning.Date(('time', ), resolution='M')
     collection = zcollection.create_collection('time',
                                                zds,
                                                partition_handler,
                                                '/my_collection',
                                                filesystem=fs)
```

```
[6]: cluster = dask.distributed.LocalCluster(processes=False)
     client = dask.distributed.Client(cluster)
     collection.insert(zds)
```

```
/work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/site-
packages/distributed/node.py:182: UserWarning: Port 8787 is already in use.
Perhaps you already have a cluster running?
Hosting the HTTP server on port 46109 instead
  warnings.warn(
```

Update overlap : The expected function usually return an array that has the same shape as the overlapped regions :

```
[12]: def callback(zds, partition_info: tuple[str, slice]):
          print("Update")
          return dict(var1 = zds["var1"].values)
```

```
collection.update(callback, depth=1)
```

```
Update
Update
Update
Update
Update
Update
Update
```

However, in some cases, we would like to submit a function that returns only the non-overlapped parts. In the current zcollection version, this raises an error when storing the result

[14]:
```
def callback_trimmed(zds, partition_info: tuple[str, slice]):
    sl = partition_info[1]
    return dict(var1 = zds["var1"].values[sl])

collection.update(callback_trimmed, depth=1)
```

```
2023-04-21 09:48:13,350 - distributed.worker - WARNING - Compute Failed
Key:       callback_trimmed-f4a02b3f806f1c7f09a342b5112a20ec
Function:  wrap_function
args:      (('/my_collection/year=2000/month=01',
'/my_collection/year=2000/month=02', '/my_collection/year=2000/month=03',
'/my_collection/year=2000/month=04', '/my_collection/year=2000/month=05',
'/my_collection/year=2000/month=06'))
kwargs:    {}
Exception: 'ValueError("parameter \'value\': expected array with shape (9, 25),
got (0, 25)")'
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[14], line 5
      2     sl = partition_info[1]
      3     return dict(var1 = zds["var1"].values[sl])
----> 5 collection.update(callback_trimmed, depth=1)

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
  ↪site-packages/zcollection/collection/__init__.py:925, in Collection.
  ↪update(self, func, depth, filters, partition_size, selected_variables, *args,
  ↪**kwargs)
    920 client = dask_utils.get_client()
    922 batches = dask_utils.split_sequence(
    923     tuple(self.partitions(filters=filters, lock=True)), partition_size
    924     or dask_utils.dask_workers(client, cores_only=True))
--> 925 storage.execute_transaction(
    926     client, self.synchronizer,
    927     client.map(local_func, tuple(batches), key=func.__name__))

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
  ↪site-packages/zcollection/storage.py:65, in execute_transaction(client,␣
  ↪synchronizer, futures, **kwargs)
     63     with synchronizer:
     64         awaitables = client.compute(futures, **kwargs)
---> 65         return client.gather(awaitables)
     66 except:  # noqa: E722
     67     # Before throwing the exception, we wait until all future scheduled
     68     # ones finished.
     69     dask.distributed.wait(awaitables)

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
  ↪site-packages/distributed/client.py:2313, in Client.gather(self, futures,␣
  ↪errors, direct, asynchronous)
   2311 else:
   2312     local_worker = None
-> 2313 return self.sync(
   2314     self._gather,
   2315     futures,
   2316     errors=errors,
   2317     direct=direct,
   2318     local_worker=local_worker,
   2319     asynchronous=asynchronous,
   2320 )

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
  ↪site-packages/distributed/utils.py:338, in SyncMethodMixin.sync(self, func,␣
  ↪asynchronous, callback_timeout, *args, **kwargs)
    336     return future
    337 else:
--> 338     return sync(
    339␣
  ↪        self.loop, func, *args, callback_timeout=callback_timeout, **kwargs
    340     )

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
  ↪site-packages/distributed/utils.py:405, in sync(loop, func, callback_timeout,␣
  ↪*args, **kwargs)
    403 if error:
    404     typ, exc, tb = error
--> 405     raise exc.with_traceback(tb)
    406 else:
    407     return result

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
  ↪site-packages/distributed/utils.py:378, in sync.<locals>.f()
    376         future = asyncio.wait_for(future, callback_timeout)
    377     future = asyncio.ensure_future(future)
--> 378     result = yield future
    379 except Exception:
    380     error = sys.exc_info()

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/tornado/gen.py:769, in Runner.run(self)
    766 exc_info = None
    768 try:
--> 769     value = future.result()
    770 except Exception:
    771     exc_info = sys.exc_info()

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/distributed/client.py:2176, in Client._gather(self, futures,␣
 ↪errors, direct, local_worker)
   2174         exc = CancelledError(key)
   2175     else:
-> 2176         raise exception.with_traceback(traceback)
   2177     raise exc
   2178 if errors == "skip":

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/zcollection/collection/detail.py:281, in wrap_function()
    277 for partition in partitions:
    278     ds, indices = _load_dataset_with_overlap(depth, dim, fs, immutable,
    279                                              partition, partitions,
    280                                              selected_variables)
--> 281     update_with_overlap(func, ds, indices, dim, fs, partition, *args,
    282                         **kwargs)

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/zcollection/collection/detail.py:108, in update_with_overlap()
    106 for varname, array in dictionary.items():
    107     slices = _get_slices(ds[varname], dim, indices)
--> 108     update_zarr_array(
    109         dirname=join_path(path, varname),
    110         array=array[slices],  # type: ignore[index]
    111         fs=fs,
    112     )

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/zcollection/storage.py:300, in update_zarr_array()
    297     array = array.filled(store.fill_value)
    299 # store[:] = array
--> 300 store.__setitem__(Ellipsis, array)
    302 # Invalidate any cached directory information.
    303 fs.invalidate_cache(dirname)
```

```
File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/zarr/core.py:1373, in __setitem__()
   1371     self.vindex[selection] = value
   1372 else:
-> 1373     self.set_basic_selection(pure_selection, value, fields=fields)

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/zarr/core.py:1468, in set_basic_selection()
   1466     return self._set_basic_selection_zd(selection, value, fields=fields
   1467 else:
-> 1468     return self._set_basic_selection_nd(selection, value, fields=fields

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/zarr/core.py:1772, in _set_basic_selection_nd()
   1766 def _set_basic_selection_nd(self, selection, value, fields=None):
   1767     # implementation of __setitem__ for array with at least one dimensi n
   1768
   1769     # setup indexer
   1770     indexer = BasicIndexer(selection, self)
-> 1772     self._set_selection(indexer, value, fields=fields)

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/zarr/core.py:1800, in _set_selection()
   1798     if not hasattr(value, 'shape'):
   1799         value = np.asanyarray(value, like=self._meta_array)
-> 1800     check_array_shape('value', value, sel_shape)
   1802 # iterate over chunks in range
   1803 if not hasattr(self.store, "setitems") or self._synchronizer is not Non␣
 ↪\
   1804    or any(map(lambda x: x == 0, self.shape)):
   1805     # iterative approach

File /work/scratch/chevrir/.conda/envs/proto_duacs_karin/lib/python3.10/
 ↪site-packages/zarr/util.py:547, in check_array_shape()
    544     raise TypeError('parameter {!r}: expected an array-like object, got
 ↪{!r}'
    545                     .format(param, type(array)))
    546 if array.shape != shape:
--> 547     raise ValueError('parameter {!r}: expected array with shape {!r},␣
 ↪got {!r}'
    548                      .format(param, shape, array.shape))

ValueError: parameter 'value': expected array with shape (9, 25), got (0, 25)
```

It would be great if we could have a trim argument similar to dask.map_overlap, that allows us to specify if our function returns an array which has already been trimmed

[ ]:
```
collection.update(callback_trimmed, depth=1, trim=False)
```