Item 313
git_comments:

1. ordinary numpy array
2. 0-dim
3. 0-size
4. Test zero_copy

git_commits:

1. **summary:** [MXNET-1398] Enable zero-copy from numpy to MXNet NDArray (#14733)
   message: [MXNET-1398] Enable zero-copy from numpy to MXNet NDArray (#14733) * Enable zero-copy from numpy to MXNet NDArray * should work * make lint happy * fix stupid typos * wip to address comments * Address comments * Address comments * Remove redundant code
   label: documentation

github_issues:

1. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
   body: Last time I did a benchmark and @szha pointed out [that](https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
2. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
   body: Last time I did a benchmark and @szha pointed out [that](https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
3. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
   body: Last time I did a benchmark and @szha pointed out [that](https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
4. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
   body: Last time I did a benchmark and @szha pointed out [that](https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
5. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
   body: Last time I did a benchmark and @szha pointed out [that](https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
6. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
   body: Last time I did a benchmark and @szha pointed out [that](https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
7. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
   body: Last time I did a benchmark and @szha pointed out [that](https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the

numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?

8. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
   body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
9. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
   body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
10. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
11. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
12. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
13. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
14. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
15. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
16. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
17. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) >

MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?

18. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?
    label: code-design
19. **title:** Feature request: share memory between numpy.array and mxnet.ndarray
    body: Last time I did a benchmark and @szha pointed out [that] (https://gist.github.com/SunDoge/59a8ff336703b45be30b46dc3ee8b4ab#gistcomment-2841120) > MXNet made the choice of not doing a zero-copy for numpy arrays, but instead making a copy of the numpy data. This is a safe choice but sometimes it can be a performance problem. I check the C API and find no function to share data from outside. Is it possible for MXNet to provide such api?

github_issues_comments:

1. I will take a look this weekend.
2. Hey, I did some research this weekend and am convinced it might be possible. So is it acceptable if we assume we are using cython (instead of ctypes) @szha
3. @junrushao1994 could you elaborate on the consideration?
4. The point is that we need transfer/share ownership of numpy's ndarray to mxnet's C++ backend, because we cannot guarantee the frontend object to exist forever. Therefore, we need a customized deleter of MXNet's NDArray, aka calling something `Py_DECREF` of the numpy's ndarray object from the C++ backend - It is possible to implement the deleter via ctypes or cython, and then pass it to something roughly like follows this chunk of code. https://github.com/apache/incubator-mxnet/blob/0f88f61379bd5f59fff6b825be1507d020bf2b7e/include/mxnet/ndarray.h#L131-L148 Per private discussion with @reminisce, his concern is how this could be compatible with the executor and memory planning, in which the executor may take over the ownership. I am not sure about this.
5. For the sharing from numpy to NDArray, we should use ctypes or weakref module add the inference of numpy object, and decrease the inference through NDArray::deleter. For the sharing from NDArray to numpy, I think we can add a deleter attribute for numpy object. https://docs.scipy.org/doc/numpy/user/basics.subclassing.html?highlight=deleter
6. Agreed with @wkcn
7. I prototyped a version that supports zero copy from numpy to DLManagedTensor in dlpack 0.2, but it turns out that MXNet hasn’t support dlpack 0.2 yet...
8. We can update the submodule dlpack.
9. @wkcn I opened a PR about this to https://github.com/dmlc/dlpack/pull/38.
10. @wkcn Sorry I made a mistake. @reminisce reminded me that we already got dlpack 0.2 in MXNet, so everything should be fine
11. In the submodule DLPack, DLPACK_VERSION is 010 rather than 020.
12. @wkcn this is because dlpack forgots to change its version number to 020 when releasing v0.2 lol. I check the commit hash [here](https://github.com/apache/incubator-mxnet/tree/numpy/3rdparty), and it is identical to tag [v0.2](https://github.com/dmlc/dlpack/releases/tag/v0.2).
13. Being lazy for a while, and now I am thinking of adding an API to mxnet. @wkcn @szha @SunDoge What do you guys think of names for this API? mx.nd.zerocopy_from() or anything else?
14. I think it is suitable to add a new argument. # NDArray to NumPy a = mx.nd.array([1,2,3]) b = a.asnumpy(shared=True) # NumPy to NDArray c = np.array([4,5]) d = mx.nd.array(c, shared=True)
15. I think adding a parameter to the existing API may introduce certain level of ambiguity which is not desirable. For example, `mx.nd.array` takes not just numpy arrays as arguments, but `shared=True` only works for numpy arrays. We can consider adding this parameter to a new API, for example: `mx.nd.from_numpy(zero_copy=True)`, and it falls back to `mx.nd.array` when `zero_copy` is `False`.
16. @reminisce Sounds good. Will do!
17. Thx.

github_pulls:

1. **title:** [MXNET-1398] Enable zero-copy from numpy to MXNet NDArray
   body: ## Description ## This PR allows users to convert numpy's NDArray without copying content data. This is achieved by converting numpy's NDArray to `DLManagedTensor`, which is introduced in https://github.com/dmlc/dlpack/pull/38. We expect that this would help users reduce major performance bottleneck when this conversion is conducted frequently. See also: #14244. ## Checklist ## ### Essentials ### Please feel free to remove inapplicable items for your PR. - [x] The PR title starts with [MXNET-$JIRA_ID], where $JIRA_ID refers to the relevant [JIRA issue](https://issues.apache.org/jira/projects/MXNET/issues) created (except PRs with tiny changes) - [x] Changes are complete (i.e. I finished coding on this PR) - [x] All changes have test coverage: - Unit tests are added for small changes to verify correctness (e.g. adding a new operator) - Nightly tests are added for complicated/long-running ones (e.g. changing distributed kvstore) - Build tests will be added for build configuration changes (e.g. adding a new build option with NCCL) - [x] Code is well-documented: - For user-facing API changes, API doc string has been updated. - For new C++ functions in header files, their functionalities and arguments are documented. - For new examples, README.md is added to explain the what the example does, the source of the dataset, expected performance on test set and reference to the original paper if applicable - Check the API doc at http://mxnet-ci-doc.s3-accelerate.dualstack.amazonaws.com/PR-$PR_ID/$BUILD_ID/index.html - [x] To the my best knowledge, examples are either not affected by this change, or have been fixed to be compatible with this change ### Changes ### - [x] Rewrite `NDArray::FromDLPack` to make it resource safe. - [x] Add DLManagedTensor and its deleter via ctypes that allows C++ side deletes it. ## Comments ## Nothing.

github_pulls_comments:

1. @reminisce @wkcn Could you help review? Thanks!
2. @reminisce Hey I have already addressed your comments. Could you review again? Thanks!
3. Merged. Thank you!

github_pulls_reviews:

1. In c_api.h, there are the three apis: ```c++ int MXNDArrayToDLPack(NDArrayHandle handle, DLManagedTensorHandle *out_dlpack); int MXNDArrayFromDLPack(DLManagedTensorHandle dlpack, NDArrayHandle *out_handle); int MXNDArrayCallDLPackDeleter(DLManagedTensorHandle dlpack); ``` MXNDArrayFromDLPack is the same as MXNDArrayFromDLManagedTensor.
2. It seems that FromDLManagerTensor is the same as FromDLPack, could we only use FromDLPack function?
3. We can test whether np_array and mx_array share the memory. For example, modify an element of np_array or mx_array, then check the other.
4. Could you please add an assertion to check whether str(array.dtype) in TYPE_MAP?
5. **body:** If this function is not user facing, better to make it private, i.e. `_make_managger_ctx`.
   label: code-design
6. Should we pass `dtype=ndarray.dtype`?
7. Better to raise instead of assert.
8. Same as above.
9. Same as above.
10. Remove `print`
11. **body:** Can you add more test cases including 1. scalar tensors. 2. zero-size tensors. 3. `zero_copy=False`. 4. Perturbing the elements of numpy ndarray will equivalently affect mxnet NDArray, and vice versa. 5. Deleting either one of them should not obliterate the content of the other one when `zero_copy=True` and `False`. 6. assertRaise for numpy ndarrays that are not c-contiguous.
    label: test
12. Do you need to increase the ref count of array?
13. **body:** There are asserts everywhere in MXNet's python package...Anyway, I will change it
    label: code-design
14. **body:** No, this is rather a temporary stack variable
    label: code-design
15. Oh, I meant where incref for numpy object happened. NVM, just saw line 4189, missed reading that before.
16. Good catch!

17. The most important reason that I didn't call the original API is its overly-strong underlying assumption made me scared. Correct me if I was wrong, but I feel that it asks the DLManagedTensor to be alive throughout the entire lifetime of the internal NDArray, i.e. the frontend should never delete the numpy ndarray before we delete the mx's one, otherwise the entire process would crash on the destructor. To be more specific, in the implementation of `NDArray::FromDLPack`, the deleter would assume the argument `tensor` always points to a valid address - this doesn't usually hold, especially for that in the frontend user may delete the numpy's ndarray. I was not aware why it is initially designed like this.
18. No. The `FromDLManagerTensor ` does not require the address `tensor` points to to be alive.
19. I wrote the function FromDLPack, and it is my mistake. Could you please replace FromDLPack with FromDLManagedTensor, but keep the function name FromDLPack? Thank you!
20. @wkcn No worries! I will try. Thanks!
21. I tried several hours unifying these two implementations, but not able to resolve some core dumps in the unittest `test_ndarray.test_dlpack` because `manager_ctx` points to an invalid address (something like 0x6). Maybe we can do it later.
22. Done
23. Done. Thanks :-)
24. Done per discussion with @wkcn. Now we unify `FromDLPack ` and `FromDLManagedTensor `. Thank you so much!
25. when doing zero-copy, should the ownership of the data pointer be transferred to the new ndarray? do we need to update the writable flag? what happens when the original data is updated by numpy?
26. @szha They share the ownership - which means numpy's modification is transparent to mxnet and vice versa. As requested by @reminisce in his review, I added a testcase for this transparency.
27. how does it work when using the asynchronous engine? since numpy calls happen immediately, allowing shared ownership may give unexpected results. consider the following case: ``` a = np.array(...) b = nd.from_numpy(a, zero_copy=True) c = nd.expensive_op(b, in_place=True) d = np.inplace_op(a) # this is called when c hasn't finished yet ``` given the asynchronous execution, it's hard to tell whether people should expect d to be based on the input before or after `expensive_op`
28. **body:** (We actually have such discussion long time before, but it's good to make things open and achievable to the community) My understanding is that your concern is a general question in multi-threading (see the code below). In any multi-threaded system, we may expect this issue to exist -- and MXNet has multi-threaded engine, so this issue exists, right? We have discussion that how about the content of array is changed somewhere inside the system but user don't know that, should we set up the `WRITEABLE` flag to be true to prevent users from writing, or should we completely disallow users to access this numpy array? The question, in the high-level, is all about trade-off. Restricting the freedom of users may make things less error-prone, while giving users full freedom may make the system more powerful. On one hand, in my humble opinion, restricting the freedom of users, like setting up `WRITEABLE` flag, does not completely resolve this issue, because users can still encounter pitfalls when they read the array using the numpy side API. The only helps in this case is to completely disable users from read or write the array, which grant them on access only the numpy side - This is more like guaranteeing safety by disallowing users to create threads/processes. On the other hand, giving users freedom enables them to do more things, and does not necessarily mean that they will definitely make a mistake, because we will encourage to use mxnet's numpy compatibility, which is on the way. Anyway, I am open to any discussion and open to possible changes setting up several flags on the numpy array.
    label: code-design
29. The API should have clear semantics regardless of users' knowledge on how MXNet execution works. If you worry about flexibility I'd recommend making it an option to not disable writable flag.
30. No. The point is that writable flag doesn't address the issue at all.
31. I think completely transferring the ownership from numpy and disabling access to the original array is preferred. If we are to expose numpy array for joint use, it should be done in asnumpy as another zero-copy creation feature, so that the wait_to_read can be done by us
32. **body:** Maybe we can add a new bool argument for `from_numpy`. The argument is `True` by default, and it disables access to the original NumPy array. When users know the asychronous risk, they can set this arugument to `False`, in order to access to the original NumPy array.
    label: code-design
33. Once we add a zero-copy option to asnumpy where the ownership is completely transferred back to numpy, I think there wouldn't be any need for co-ownership.
34. Consider the following case: ```python # the variable `a` is a `mx.nd.NDArray` object a = mx.nd.array([1,2,3]) b = a.asnumpy(zero_copy=True) # users call `asnumpy(zero_copy=True)` twice c = a.asnumpy(zero_copy=True) c[:] = 3 print(a) print(b) a[:] = 5 a.wait_to_read() print(b) print(c) ``` It's

necessary to keep co-ownership. We can add extra option in `from_numpy` and `asnumpy` to prompt users to call the function `wait_to_read()`. Alternatively, can we add a hook into `numpy.ndarray`? When users access `numpy.ndarray`, `wait_to_read()` will be called automatically. Reference: https://docs.scipy.org/doc/numpy/reference/arrays.classes.html

35. If zero-copy is on, then `a` should not be usable after the `asnumpy` call. The point is to let the framework deal with all synchronization, and co-ownership doesn't allow that. And for the same case, you can simply copy `c` from `b` (the new numpy array) instead of `a` (the old ndarray), so it's not necessary to keep co-ownership.
36. **body:** The hook in the context seems to refer to our own subclass of numpy array. Creating a new subclass seems like overkill.
    label: code-design
37. For reference, @szha's suggestion was addressed in https://github.com/apache/incubator-mxnet/pull/14948

jira_issues:

1. **summary:** Enable zero-copy from numpy to MXNet NDArray
   description: ## Description ## This PR allows users to convert numpy's NDArray without copying content data. This is achieved by converting numpy's NDArray to `DLManagedTensor`, which is introduced in https://github.com/dmlc/dlpack/pull/38. We expect that this would help users reduce major performance bottleneck when this conversion is conducted frequently. ## Checklist ## ### Essentials ### Please feel free to remove inapplicable items for your PR. - [x] The PR title starts with [MXNET-$JIRA_ID], where $JIRA_ID refers to the relevant [JIRA issue](https://issues.apache.org/jira/projects/MXNET/issues) created (except PRs with tiny changes) - [x] Changes are complete (i.e. I finished coding on this PR) - [x] All changes have test coverage: - Unit tests are added for small changes to verify correctness (e.g. adding a new operator) - Nightly tests are added for complicated/long-running ones (e.g. changing distributed kvstore) - Build tests will be added for build configuration changes (e.g. adding a new build option with NCCL) - [x] Code is well-documented: - For user-facing API changes, API doc string has been updated. - For new C++ functions in header files, their functionalities and arguments are documented. - For new examples, README.md is added to explain the what the example does, the source of the dataset, expected performance on test set and reference to the original paper if applicable - Check the API doc at http://mxnet-ci-doc.s3-accelerate.dualstack.amazonaws.com/PR-$PR_ID/$BUILD_ID/index.html - [x] To the my best knowledge, examples are either not affected by this change, or have been fixed to be compatible with this change ### Changes ### - [x] Rewrite `NDArray::FromDLPack` to make it resource safe. - [x] Add DLManagedTensor and its deleter via ctypes that allows C++ side deletes it. ## Comments ## Nothing.

jira_issues_comments: