Item 293
git_comments:

1. ! * \brief Seed the global random number generator of the given device. * \param seed the random number seed. * \return 0 when success, -1 when failure happens.
2. ! * \brief Seed the random number generator of the device. * \param seed the seed to set to global random number generators.
3. ! * \brief Seed the random number generators of the given context. * \param seed the seed to the random number generators.
4. set seed to a sampler using global_seed and device id
5. reset pointer to ensure the same result with the same seed.
6. set seed to a sampler
7. set seed to a PRNG using global_seed and device id
8. Tests that seed setting of parallel rng for specific context is synchronous w.r.t. rng use before and after.
9. Avoid excessive test cpu runtimes.
10. Create enough samples such that we get a meaningful distribution.
11. The samples should be identical across different gpu devices.
12. To flush out a possible race condition, run multiple times.
13. Currently python API does not provide a method to get the number of gpu devices. Waiting for PR #10354, which provides the method, to be merged. As a temporal workaround, try first and catch the exception caused by the absence of the device with `dev_id`.
14. Check imperative. `multinomial` uses non-parallel rng.
15. Check symbolic. `uniform` uses parallel rng.
16. Collect random number samples from the generators of all devices, each seeded with the same number.
17. Tests that seed setting of std (non-parallel) rng for specific context is synchronous w.r.t. rng use before and after.
18. Check symbolic. `multinomial` uses non-parallel rng.
19. Set seed for the context variously based on `start_seed` and `num_init_seeds`, then set seed finally to `final_seed`
20. Check imperative. `uniform` uses parallel rng.

git_commits:

1. **summary:** [MXNET-262] Implement mx.random.seed_context to seed random number generators of a specific device context (#10367)
   message: [MXNET-262] Implement mx.random.seed_context to seed random number generators of a specific device context (#10367) * Implement mx.random.seed_context `mx.random.seed_context` is to seed the random number generator of a specific device context. The random number sequence on the deivce is completly determined by the given seed and is independent of the device id. This is in contrast to `mx.random.seed` which seeds all generators using the device id implicitly. * Fix lint * Adding ctx argument to random.seed instead of new function random.seed_context * Doc fix

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [MXNET-262] Implement mx.random.seed_context to seed random number generators of a specific device context
   body: This PR is modified to add an optional argument `ctx` to `mx.random.seed` instead of introducing a new function `seed_context`. Seeding with `ctx` option produces random number sequence independent of device id. The below is the original description before the modification. --- ## Description ## This PR introduces a function `mx.random.seed_context` to seed random number generators of a specific device context. `mx.random.seed_context(seed, ctx)` seeds the parallel and non-parallel generators of `ctx` where `ctx` is optional and the default is the current context. The random number sequence generated on the device is completely determined by `seed`, differently from existing `mx.random.seed` which implicitly uses the device id of each context together with the given seed. Using device id is reasonable to seed all generators at once with a given seed, but to reproduce the same random number sequence we

need to set the running context besides seeding with the same number. Sometimes setting a context would be inconvenient or the running context could be not deterministic. `mx.random.seed_context` would be helpful in such cases. The implementation is simple. It just hands over the given seed to the underlying generators. The unit tests are an adaptation of the existing tests for `mx.random.seed`. Here is an example. ```python # Seeding with `mx.random.seed`. Different results on gpu(0) and gpu(1). >>> with mx.Context(mx.gpu(0)): ... mx.random.seed(99) ... print(mx.nd.random.uniform(0, 1, 3)) [0.29560053 0.07938761 0.29997164] <NDArray 3 @gpu(0)> >>> with mx.Context(mx.gpu(1)): ... mx.random.seed(99) ... print(mx.nd.random.uniform(0, 1, 3)) [0.8797334 0.8857584 0.3797555] <NDArray 3 @gpu(1)> # Seeding with `mx.random.seed_context`. Identical results on gpu(0) and gpu(1). # This seeds the generator of the current context. Other generators are not touched. # To seed a specific device context, set the optional argument `ctx`. >>> with mx.Context(mx.gpu(0)): ... mx.random.seed_context(99) ... print(mx.nd.random.uniform(0, 1, 3)) [0.29560053 0.07938761 0.29997164] <NDArray 3 @gpu(0)> >>> with mx.Context(mx.gpu(1)): ... mx.random.seed_context(99) ... print(mx.nd.random.uniform(0, 1, 3)) [0.29560053 0.07938761 0.29997164] <NDArray 3 @gpu(1)> ``` ## 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

github_pulls_comments:

1. I would add a `ctx=None` argument to seed instead
2. @piiswrong Omitting `ctx` argument usually means that it is the current context. So I think that adding `ctx=None` to `seed` could be potentially confusing.
3. I still think its better than adding an API. `seed_context` looks ugly
4. maybe you can make it defaults to 'all'
5. I removed `seed_context` and added `ctx="all"` argument to `seed`.
6. @piiswrong can this be merged. will add this to api changes in release notes.
7. Thanks a lot for this change, this really something we've been looking for! @asmushetzel @KellenSunderland

github_pulls_reviews:

jira_issues:

1. **summary:** Implement mx.random.seed_context to seed random number generators of a specific device context
   description: This PR introduces a function `mx.random.seed_context` to seed random number generators of a specific device context. `mx.random.seed_context(seed, ctx)` seeds the parallel and non-parallel generators of `ctx` where `ctx` is optional and the default is the current context. The random number sequence generated on the device is completely determined by the seed, differently from existing `mx.random.seed` which implicitly uses the device id of each context together with the given seed. Using device id is reasonable to seed all generators at once with a given seed, but to reproduce the same random number sequence we need to set the running context besides seeding with the same number. Sometimes setting a context would be inconvenient or the running context could be not deterministic. `mx.random.seed_context` would be helpful in such cases.

jira_issues_comments: