NDN-TR68: Utilizing NDN for Domain Science Applications - a Genomics Example

Rini Pauly, Cameron Ogle, Cole Mcknight, David Reddick, Justin Presley, Susmit Shannigrahi, Alex Feltus
December 21, 2020

Abstract

Advanced imaging and DNA sequencing technologies now enable the diverse biology community to routinely generate and analyze terabytes of high resolution biological data. The community is rapidly heading towards the petascale in single investigator laboratory settings. As evidence, the single NCBI SRA[1] central DNA sequence repository contains over 45 petabytes of biological data. Given the geometric growth of this and other genomics repositories, an exabyte of mineable biological data is imminent. The challenges of effectively utilizing these datasets are enormous as they are not only large in the size but also stored in geographically distributed repositories in various repositories such as National Center for Biotechnology Information (NCBI), DNA Data Bank of Japan (DDBJ), European Bioinformatics Institute (EBI), and NASA's GeneLab. NDN[2] has several useful properties such as name based data access, data from anywhere, in-network caching that can address data management and cyberinfrastructure challenges faced by the genomics community. This document outlines how we integrated NDN with a contemporary workflow, point out the methods used, and the challanges. This tech report will serve as a starting point for other communities trying to integrate NDN in their workflows.

1 Purpose

- Communicate to the NDN team how an end user installed NDN and published/pulled data with a description of challenges.
- Communicate to science communities how to utilize various tools and software stacks needed for successfully utilizing NDN for science workflows.
- Act as a documentation for new users as they are onboarding.

2 Resources

A number of software libraries are needed for basic integration with domain workflows. This is a list meant as a starting point. Depending on the particular requirements, more tools may need to be developed or the existing tools and libraries need to be extended.

1. Named Data Networking Website - https://named-data.net
2. Named Data Library - https://github.com/named-data/ndn-cxx
3. Named Data Forwarding Daemon - https://github.com/named-data/NFD
4. NDN Python Library - https://github.com/named-data/python-ndn
5. NDN Python repo - https://github.com/JonnyKong/ndn-python-repo
6. NDN basic tools - https://github.com/named-data/ndn-tools
7. Modified basic tools for genomics workflow - https://github.com/susmit85/ndn-tools
8. NDN Docker - https://hub.docker.com/r/dereddick/ndn-docker
9. NDN tools docker - https://hub.docker.com/r/cbmckni/ndn-tools

3 How to use NDN from an end user perspective

3.1 Install the required software

For basic installation, one needs to install two primary pieces - NFD and ndn-cxx. These can be installed from the source (refer to the respective README files) or through a package manager (*apt* for Ubuntu or *dnf* for Fedora). For

example, in Ubuntu, NFD can be installed with the following steps:

```
$sudo add-apt-repository ppa:named-data/ppa
$sudo apt-get install nfd
```

Once these software are installed the forwarder can be started with:

```
$ nfd-start
```

The staus of the forwarder can be checked with

```
$nfd-status
```

3.2 Install the required tools and test

Once the basic installation is done, the actual tools need to installed. For this use case, we wanted to transfer files over long distance links, so we needed applications that can publish and retrieve files.

The first step was to install the ndn-tools. This installes the *ndncatchunks and ndnputchunks* tools that we used for data retrieval. Note that ndnputchunks publishes data in memory and therefore, limited by the amount of RAM on a machine.

These can be installed as

```
$git clone https://github.com/named-data/ndn-tools
$cd ndn-tools
$./waf configure
$./waf
$./waf install
```

For this excise, we modified ndn-tools to work with ndn-python-repo and installed it from https://github.com/susmit85/ndn-tools.

3.3 Initial Testing

After the download, make sure to successfully configure, compile and install the repos/tools. Uninstall all previously installed versions. This is critical to avail several updated options. An example has been provided under section 'Pulling data onto a local file system'.

For testing, one can simply publish a file and download it on the same machine.

```
$nfd-status
$echo "Hello World!" | ndnputchunks /test (on one terminal)
$ndncatchunks -v /test (on another terminal)
```

3.4 Publishing a large amount of genomics data

Making NDN the work-horse for genomic workflows would make big data analysis and management much more simpler and faster. The hierarcial genomics names can be easily translated and published into NDN names. These published datasets could then be used to represent evolutionary relationships with some additional metadata. Here, we have published an actual dataset from the National Center for Biotechnology Information's (NCBI) Sequence Read Archive (SRA) database into an NDN testbed.The purpose of this experiment is to show that NDN can do data retrieval and insertion from any location.

As we mentioned earlier, *ndnputchunks* is limited by the amount of memory of a machine. For publishing a large amount of data, we utilized *ndn-python-repo* since it provides persistent disk storage. This software can be installed as:

```
$git clone https://github.com/JonnyKong/ndn-python-repo
$cd ndn-python-repo && /usr/bin/pip3 install -e .
```

Below, we have expanded the steps to publish datasets to an NDN testbed.

- **Create interface**
  Depending on if TCP or UDP overlay is desired, the corresponding faces need to be created.

  ```
  nfdc face create tcp4:<url\_to\_server\_location>
  or
  nfdc face create udp4:<url\_to\_server\_location>\\
  ```

- **Add named route**
  nfdc route add / tcp4://atmos-nwsc.ucar.edu
  This command must be used every time the data is pulled. A single user can have multiple routes.
  'nfdc route add /test tcp4://atmos-nwsc.ucar.edu'
  'nfdc route add /BIOLOGY tcp4://atmos-nwsc.ucar.edu'

- **Publish Method A:**
  putfile.py [-h] -r BASE_REPO_NAME -f FILE_PATH -n NAME_AT_REPO
  e.g. 'python3 putfile.py -r testrepo -f /raid/rpauly/10K -n /test1/test1'
  -n cannot have larger than 32 octets currently.

- **Publish Method B:**
  There is another method for publishing, that might be still in the testing phase:
  curl <ftp> | ndnputchunks <ndn name> -
  The last dash after ndn names should read from FTP output.
  This can be used in a bash redirect script to publish multiple files.

- **Errors:**
  .curl: (78) RETR response: 550
  This means that the file doesn't exist (code 550). Check the curl ftp link separately before piping to ndnputchunks.
  .Error if no route is added: 'Received Nack with reason NoRoute for Interest'.

  Before publishing the data, it important to know the storage space available. A quick status check reveals that the jobs are running even after the publication - this is by design.Finally, after publishing the data to check the job status use: ps aux|grep rpauly

3.5 Pulling data

To pull data from NDN network, the user must create an interface, add a route to the requested data, then use one of the methods outlined below to initiate the transfer.

- **Create interface**
  nfdc face create tcp4:<url_to_server_location>
  This command must be used every time the data is pulled.

- **Add named route**
  nfdc route add / tcp4://atmos-nwsc.ucar.edu
  A single user can have multiple routes.
  'nfdc route add /test tcp4://atmos-nwsc.ucar.edu'
  'nfdc route add /BIOLOGY tcp4://atmos-nwsc.ucar.edu'

- **Pull Method A:**
  # ndncatchunks method

```
ndncatchunks -vTD NAME\_AT\_REPO>write\_file\_to\_local\_computer
-v turn on verbose output (per segment information)
-T [ --typed-encoding] use typed name component for segment number
-D [ --no-version-discovery] skip version discovery, even if the supplied name does n

For example,
$ ndncatchunks -vTD /BIOLOGY/SRA/9605/9606/NaN/RNA-Seq/ILLUMINA/TRANSCRIPTOMIC/
PAIRED/Kidney/PRJNA359795/SRP09590/SRX2458154/SRR5139394/SRR5139394_1 >
SRR5139394_1.fastq.gz
```

Ensure that the latest ndn-tools repo. The option -vTD is not available is some older versions. Also, make sure the repo is in the current PATH.

- **Pull Method B:**
  Getfile.py method
  python3 getfile.py -r BASE_REPO_NAME -n NAME_AT_REPO
  The user needs to make sure to create a route and an interface before pulling the data. If not, we get the below error:
  INFO:Timeout
  INFO:Manually shutdown
  As the time of writing this, getfile was slower than ndncatchunks.

3.6 Deploying NDN with Docker

In order to perform a NDN transfer, the user must have access to the ndn-tools software. This can be achieved by installing the sofware directly on the machine, which is outlined above in sections 3.1 and 3.2. Many use cases will benefit by instead pulling the Docker image and deploying a ndn-tools container to the machine using Docker or Singularity. Using the ndn-tools Docker image eliminates the need to install the software directly on the machine, while only taking a minor loss in performance. Using a ndn-tools container with a mount to the file system of the machine is reccommended for all use cases aside from those where containers cannot be deployed, or where optimal performance is necessary.

To deploy NDN to a machine using Docker:

- **Install Docker**
  Install Docker: https://docs.docker.com/get-docker/

  Docker must be installed on the machine. If Docker cannot be installed on the machine, look into using Singularity to pull and deploy ndn-tools: https://singularity.lbl.gov/quickstart

- **Create a Docker Volume**
  docker volume create persistent-data

  This command creates a Docker Volume named “persistent data” for the container to mount to.

- **Run the Image**
  docker run -it –mount source=persistent-data,target=/workspace cbmckni/ndn-tools

  This command pulls the ndn-tools image and deploys it as a container mounted to the “persistent data” volume.

3.7 Integrating NDN with a workflow

To add an NDN tranfer to a workflow, the workflow manager must have access to the ndn-tools software on the same filesystem the workflow is using. To achieve this, ndn-tools may be installed directly to the workflow’s execution environment by following the steps in sections 3.1 and 3.2, or the software can be accessed through a container deployed to the execution environment of the workflow. Integration of the ndn-tools container differs between workflow managers. Nextflow[3] is a highly scalable and portable workflow manager, that offers support for Docker

containers. To integrate NDN with a Nextflow pipeline, we created the necessary docker containers and followed the Nextflow documentation: https://www.nextflow.io/docs/latest/docker.html

4 Challenges from a user's perspective

4.1 Installation

1. Installing ndn-cxx was a challenge as multiple versions of the library exists in ndn-tools, nfd folder and ndn-python-repo. Perhaps clearer instructions would be helpful with information about the latest version, github location and instructions to uninstall previous versions.

4.2 Publishing Data

1. After ndn-python-repo installation, including a detailed example to demonstrate how to use the several options would be helpful. Also, details of potential errors and how to solve the problem.
2. Have a tool that determines storage and RAM available for data publishing. Put out a message for successful data publication that could be checked/verified at any consecutive logins without pulling the entire file.
    - Currently the best solution we have to verify published data without pulling the entire file to checksum is to pull only the first few bytes and verify that. While not a guarentee the entire file is published correctly, this is a work around.
3. Prompt if the exact same NAME_AT_REPO is used for publishing 2 files. Currently, there is no check in place.
    - While this is currently not possible, this is an upcoming feature that will be added to NDN in the future.
4. We upload several files at the same time, so if there is a way to point to the directory or ftp location that allows iterative publications of all files that would be great!
    - Currently we are using a bash script that is able to be set to a local directory and publish everything inside but it is currently only serially so publishing large data sets takes a long time and is currently unable to stream data into publishing to the repo but future work will be done to allow this. (Insert URL of code)

4.3 Pulling Data

1. Accessing the storage space needed to pull the data and confirm with the user that the space is available. Implementing checks to make sure the file is not corrupt and checksum.
    - While NDN has the ability to verify if data gets corrupted in transit reguardless if UDP or TCP is used by checking at the application level (Level 7 of the OSI), care should still be taken at the repo side to prevent corruption by utilizing technologies like ZFS to protect the generated packets at rest.

5 Summary

This report is aimed towards scientific communities trying to utilize NDN for data management. Integrating NDN with community workflows require understanding and deploying several software libraries and tools. This report outlines those tools and explains how each tool needs to be configured and used for a successful integration. This report will also help science communities to create a basic data publication and transfer mechanism over NDN.

References

[1] NCBI. Ncbi sequence read archive, 2019.

[2] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[3] Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316–319, 2017.