NDN FOR DATA INTENSIVE SCIENCE - SANDIE

Susmit Shannigrahi, Edmund Yeh, Harvey Newman, Christos Papadopoulos, Catalin Lordache, and others
susmit@colostate.edu
The “big-data” problem

- The LHC is world’s largest data intensive application (1 Exabyte data by the end of 2018)
- LHC network connects CERN to ~500 tiered sites worldwide
- Demand is growing in multiple dimensions
  - Number of datasets
  - Complexity of data
  - Global collaborations
  - Network traffic
- Present infrastructure can not scale to meet the needs of LHC Run3 (2021-23)
The SANDIE Project

- Uses NDN principles to enhance LHC network and data distribution capabilities
- Deploy caches at strategic points (both at the core and the edge)
- Integrate NDN-based backend with existing software infrastructure
  - Simplify and add new capabilities
- Deploy large scale testing infrastructures
  - SSD based caches + large disk based caches
  - Extend CSU testbed, add 40G/100G networks at 7 sites
  - Support transactions that may require upto 10TB or more data
A Global Testbed

- Expand CSU climate testbed and CalTech SDN testbed
- Deploy few more sites across the USA, Europe, and India
- Caches each with
  - Several terabytes of SSDs
  - 40-60 Terabytes of SAS disks
  - 10G to 100G network interfaces
  - NDN and Xrootd software systems
- New nodes already deployed at CalTech/NEU
Testing the Potential for NDN’s Improvements
CMS Topology Generation

- Topology: include all the Tier 1 and Tier 2 sites in the US:
  - PerfSONAR: average throughput between sites
  - 10s - 100s Gbps typically
Data Distribution and Access Performance Improvements

- VIP distributed caching algorithm developed at Northeastern
  - Set of distributed and adaptive caching algorithms
  - Minimizes the total network cost (e.g., link delays)
  - Intelligent multipath routing and caching

- Simulated over the derived CMS topology
  - Reduced total network cost by a 1000 times compared to stock routing and caching
  - Average data retrieval delays reduced by 50%
Integrating with current workflow
Integration with the mainstream data production and analysis tools of CMS

- Goals - Increased efficiency and reduced complexity for both applications and the network
- Create value for the HEP communities
- Integrate with xRootD, the de-facto data management software
- **First step** – an NDN based filesystem plugin for XRootD and a suitable NDN Producer
XrootD

4: Try open() at A
1: open("/my/file")

2: Who has "/my/file"?

Data Servers
Architecture and flow diagram

**xrootd server**
Handles xrootd protocol specific requests using its filesystem implementation:

1. Current implementations: (cephs, hdfs, etc.) for servers with access to `/mnt/hadoop` (fuse-dfs)

   - **xrootd-cephs-plugin**

2. XrdNdnFS plugin for xrd servers that can’t access `/mnt/hadoop`

**XrdNdnFS plugin**
Handles filesystem calls through NDN network. Next to the implementation required by xrootd API for file-system handling, it will also contain a NDN Consumer.

The consumer will generate interest packages with names derived from system calls, e.g.:

`/ndn/xrootd/open/store/mc/foo`

**NDN Producer - /mnt/hadoop**
Listens for NDN xrootd specific requests, e.g., url with `/ndn/xrootd` prefix.

**CMSSW**
Uses xrootd-protocols.
The NDN Consumer for XRootD plugin

• Support for: **open, fstat, read** and **close** system calls

• Composes **Interests** for these system calls over NDN:

```
/ndn/xrootd/....../root/path/for/ndn/xrd/foo?ndn.MustBeFresh=true
```

- **ndn prefix**
- **path to file of interest.**
- **ndn interest specific info**
The NDN Consumer for XRootD plugin

```
/ndn/xrootd/........./root/test/path/for/ndn/xrd/foo?ndn.MustBeFresh=true
```

- ndn prefix
- path to file of interest.
- ndn interest specific info

```
/ndn/xrootd/........./root/test/path/ndn/xrd/foo/%00%00?ndn.MustBeFresh=true
```

- ndn prefix
- path to file of interest.
- seg. no
- ndn interest specific info

- For read system call, it breaks down the request into segments.

- Handles: Interest validation, timeout and Nack with a maximum of 32 retries
XrdNdnFs xrootd plugin

• Packaged as a dynamic library (.so)

• Compiles the NDN consumer to handle the file system calls over NDN

• Handles: open, fstat, read and close file system calls
An NDN Producer for XRootD plugin

• Registers prefixes to NFD

• Performs **open**, **fstat**, **read** and **close** system calls as required
  - No need to explicitly define operations

• Keeps track of opened files

• Sends data as strings and non-negative integers

/ndn/xrootd/......./root/path/for/ndn/xrd/foo/00/?ndn.MustBeFresh=true&ndn.Nonce=825012545

*ndn prefix*  *path to file of interest.*  *ndn data specific info*
Demo

Using xrdcp tool to copy file:

[29B/29B] [100%] [================================================================]==] [29B/s]

Consumer tracing:

```
1536076342.646256 TRACE: [xrdndconsumer] Received data for fstat: /ndn/xrootd/fstat/root/test/path/for/ndn/xrd/test.txt?ndn.MustBeFresh=true&ndnNonce=2114664675
1536076342.652151 TRACE: [xrdndconsumer] Sending read file interest: /ndn/xrootd/read/root/test/path/for/ndn/xrd/test.txt/%0%07ndn.MustBeFresh=true
1536076342.658508 TRACE: [xrdndconsumer] Received data for read: /ndn/xrootd/read/root/test/path/for/ndn/xrd/test.txt/%0%07ndn.MustBeFresh=true&ndnNonce=3348515939
188904 11:52:22 3997 root.4033:19[@:ffff:192.168.56.101] XrdPoll: FD 10 detached from poller 0; num=0
188904 11:52:34 4802 cms_Finder: Waiting for cms path /tmp/.olb/olbd.admin
```
Demo - Producer tracing

1536076215.220843 TRACE: [xrdndnproducer] Allocate xrdndn::Producer

1536076215.220843 TRACE: [xrdndnproducer] Register prefixes.

1536076215.223160 INFO: [xrdndnproducer] Successfully registered prefix for: /ndn/xrootd/open

1536076215.223198 INFO: [xrdndnproducer] Successfully registered prefix for: /ndn/xrootd/close

1536076215.223214 INFO: [xrdndnproducer] Successfully registered prefix for: /ndn/xrootd/fstat

1536076215.223262 INFO: [xrdndnproducer] Successfully registered prefix for: /ndn/xrootd/read

1536076215.231777 INFO: [xrdndnproducer] Successfully registered prefix for: /ndn/xrootd


1536076342.633427 TRACE: [xrdndnproducer] Sending integer.

1536076342.635789 INFO: [xrdndnproducer] Sending: Name: /ndn/xrootd/open/root/test/path/for/ndn/xrd/test.txt?%FD%00%00%0e%55%65

MetaInfo: ContentLength: 128
Content: (size: 1)
Signature: (type: SignatureSha256WithRsa, value_length: 256)


1536076342.643215 TRACE: [xrdndnproducer] Sending string.

1536076342.644782 INFO: [xrdndnproducer] Sending: Name: /ndn/xrootd/root/test/path/for/ndn/xrd/test.txt?%FD%00%00%0e%55%65s

MetaInfo: ContentLength: 0
Content: (size: 144)
Signature: (type: SignatureSha256WithRsa, value_length: 256)

1536076342.654514 TRACE: [xrdndnproducer] onReadInterest: /ndn/xrootd/read/root/test/path/for/ndn/xrd/test.txt?%0%0%0?ndn.MustBeFresh=true&ndnNonce=3348515939

1536076342.655736 TRACE: [xrdndnproducer] Sending string.

1536076342.657288 INFO: [xrdndnproducer] Sending: Name: /ndn/xrootd/read/root/test/path/for/ndn/xrd/test.txt?%0%0%0%FD%00%00%0e%55%65%7F

MetaInfo: ContentLength: 0
Content: (size: 29)
Signature: (type: SignatureSha256WithRsa, value_length: 256)


1536076342.667473 TRACE: [xrdndnproducer] Sending integer.

1536076342.668575 INFO: [xrdndnproducer] Sending: Name: /ndn/xrootd/close/root/test/path/for/ndn/xrd/test.txt?%FD%00%00%0e%55%65%68

MetaInfo: ContentLength: 128
Content: (size: 1)
Signature: (type: SignatureSha256WithRsa, value_length: 256)

1536076350.556802 TRACE: [xrdndnproducer] Deallocate xrdndn::Producer. Closing all files.
Other ongoing work

• Improve performance of producer/consumer
• Further integration with existing workflow and tools
• Modifying naming schemes for supporting next generation event forms such as NANO-AOD
• Adaptation of NDN to take advantage of SDN enabled infrastructures
• Tuning and optimizing NDN nodes for performance
Thank You!