GridPP

UK Computing for Particle Physics

Possibilities for Named Data Networking in HEP

D. Rand, S. Fayer & D. Colling

Imperial College London

Imperial College London

Introduction

- Currently two main methods whereby LHC experiment data is transferred to sites
- The first is pre-placement using data movement orchestrators such as PhEDEx (CMS) and Rucio (ATLAS)
  - need to pre-select data expected to be popular
  - distribution of popular data dynamically
- Remote access from federated storage using the XRootD protocol (e.g. AAA (CMS) and FAX (ATLAS))
  - currently no caching
- Overall solutions have evolved into complex overlay of software
  - considerable effort goes in monitoring and support

- Perhaps there is another, simpler, approach in which
  - no attempt is made to predict data popularity
  - every attempt is made to cache data once it has actually been requested

- We think that Named Data Networking (NDN) offers interesting possibilities in this regard

Named Data Networking

- NDN is a novel network architecture, one of five projects funded by the US NSF Future Internet Architecture Program
- In NDN data are named and secured thereby becoming location-agnostic
- Each data block is addressed by a unique name which consists of a hierarchical path, a name and attributes, all separated by "/".
  - e.g. /ndn/uk/ac/imperial/ph/hep/data/somefile/1
- Data are secured by requiring producers to cryptographically sign each data packet
- Access to confidential data can also be controlled through the use of encryption
- Rather than data being transferred between two hosts, data are retrieved out of the network by requesting by name
- Crucially, as the data packets traverse the network they are cached in routers
- It is this caching of data which should reduce read latencies and overall network usage

Data Transfer

- A consumer, interested in a named segment of data, sends out a request to the network for that data in the form of an 'interest packet'
- As the interest packet is routed towards a copy of the data
  - a check is made to see if that router has a copy of the segment in its Content Store (cache). If it does it serves the data back to the interface that the interest packet came from.
  - if not, it is stored in the NDN router's Pending Interest Table (PIT) and the interest packet is forwarded until it reaches a copy
- The resulting collection of PIT entries acts as a 'trail of crumbs' for the data packet to follow back to the consumer
- If two consumers request the same data segment from a router only one interest packet is forwarded
- The forwarding strategy is programmable,
  - e.g. it is possible to send interest packets to multiple routers
  - if one route fails the data can be server by the second one, thereby building in reliability

NDN realisation

- NDN collaboration (http://named-data.net)
  - "A Collaborative Effort to Promote and Sustain the NDN Future Internet Architecture"
  - "It aims to provide a practically deployable set of protocols replacing TCP/IP that increases network trustworthiness and security, addresses the growing bandwidth requirements of modern content, and simplifies the creation of sophisticated distributed applications."
- Produce NDN platform with ndn-cxx library and Named Data Networking Forwarding Daemon (NFD)
- NFD handles NDN packet transfers and acts as NDN router
- Available for Ubuntu and MacOSX

- We have built NFD RPMs for CentOS7, our preferred distribution

Repository

- Along with data caching in routers NDN has the concept of a more permanent store known as a repository
- We have written an application in C++ to provide repository services called repo-se ('repose')
- Application links against lib-ndncxx to connect it to NDN and to backend filesystem libraries librados (part of Ceph) & libcurl
- Backends allow files to be served into the NDN namespace from either a conventional POSIX filesystem, Ceph or HTTP source
- Designed with scalability in mind

Repository client

- Small static client library, librepoclient, is also included within the repose codebase
  - provides a minimal POSIX-like interface (open(), read(), close(), ...) and is designed to be linked into shared library plugins for other applications such as ROOT & GFAL2
- Also a minimal "getfile" application for testing, which fetches an entire file from NDN and writes it to the local disk using the client library
- Current implementation is limited to read-only access for basic demonstrations and testing
- Once we finalise a plan for authentication it can then be extended to include full remote I/O (read-write)
- Original version of client requested and retrieved data segments one at a time
- Curently working on improving the client to send out multiple interest packets simultaneously

IC HEP NDN testbed

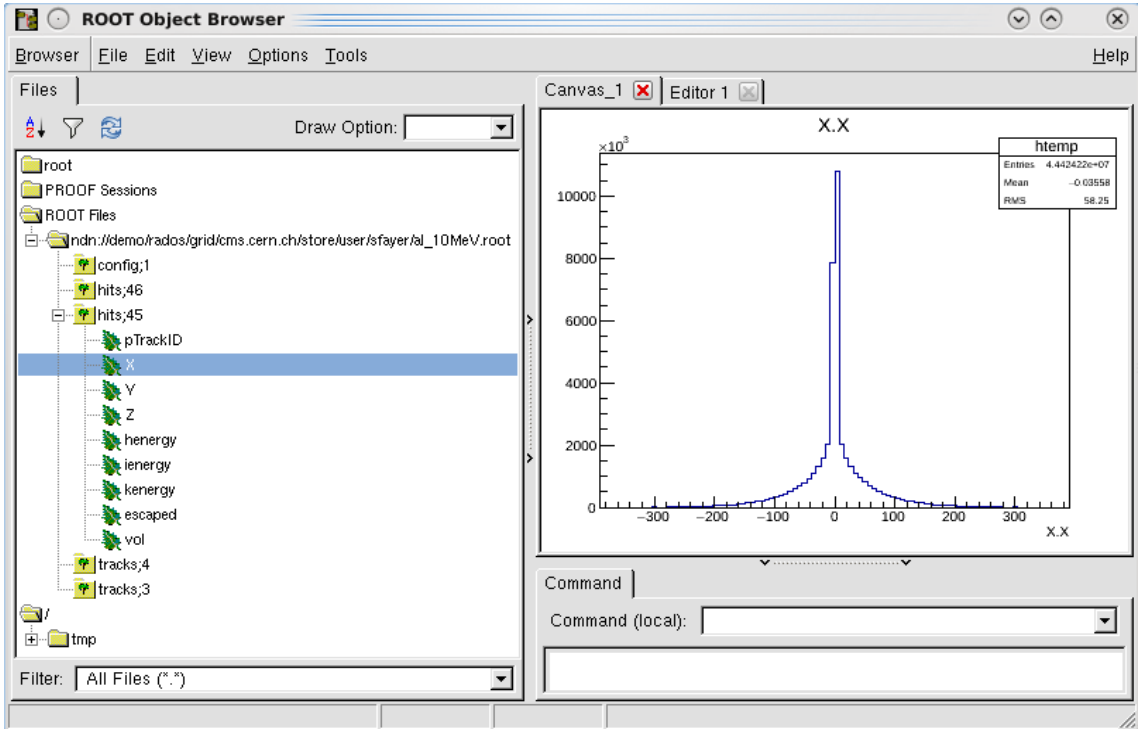
- Installed an NDN testbed running NFD software comprising:
  - a User Interface host
  - two NDN routers
  - two repositories running 'repo-se' for data storage
  - Ceph cluster storage with four 16 TB servers
- Routing paths are added in manually, e.g.

  nfdc register /demo/rados udp4://123.123.123.123
- The client on the UI requests data segments from the repo-se by sending out interest packets to the intermediate routers
- The passage of the interest packets and returning data packets can be monitored using the 'ndndump' software (https://github.com/zhenkai/ndndump)

- Currently using testbed to build up experience with NDN

ROOT plugin

- ROOT is an integral part of most HEP experiment analysis software
- We have coded up a prototype NDN ROOT plugin using C++
- On the NDN testbed we are now able to open a ROOT file stored in our Ceph cluster from the NDN UI via an NDN router
- It is currently unoptimised and rather slow



Imperial College London

Grid testing

- Testbed NFD software uses CentOS7 but our grid cluster worker nodes are running CentOS6
- We have built the NFD software on CentOS6 for grid testing
- Submit grid jobs to our production cluster which include a tarball of the NDN software in the input sandbox
  - in future will use CVMFS software area
- Such jobs are able to stage a file from the repository to the WN local disk
- Plan to test at scale with a well-resourced host acting as an NDN core router
- Would also like to test reading and caching of ROOT files over longer distances

Possible use with LHC VO

- Main use case is chaotic user analysis where it is difficult to predict what data is to be read
- A version of the experiment's analysis code would be built with the NDN ROOT plugin
- VO datasets could be encrypted using an individual symmetric key per dataset
- Accessing the data would involve two steps, the client would
  - request the encrypted data
  - request the encryption key from a DataSet Key Server (DSKS) with a signed interest packet
- The key server would check the user's membership of the VO and if valid would encrypt the dataset key with the user's public key and send it back
- On receipt of the encrypted key and data packets the application would decrypt the key with user's private key and use that to access the data

Scalable routers

- NDN routers mentioned in the literature discuss the caching of data in memory
- With LHC experiment ROOT files roughly 2 GB in size and increasing, the viability of such in-memory caching is questionable
- We envisage the need to build a scalable router similar to our scalable repository
- The router would likely have several layers of caching:
  - Memcached
  - SSD
  - Ceph base storage

- May end up with a hybrid router-repository at experiment sites

Conclusion

- We feel it makes sense to examine the viability for HEP of this new networking paradigm
- Clearly it is early days, there are many unsolved issues both in Named Data Networking itself and in its application to the distribution of high energy physics data

- We have made a start by coding-up a scalable repository and ROOT plugin
- We are building our experience with Named Data Networking on our local testbed
- Would like to expand this to include other interested sites

- We would like to thank Artur Barczyk for introducing us to NDN