T3 data access via BitTorrent

Charles G Waldman
USATLAS/University of Chicago
cgw@hep.uchicago.edu

USATLAS T2/T3 Workshop Aug 19 2009

Introduction

- BitTorrent is an interesting protocol which is used to move large amounts of data across the Internet
- Can we use the p2p features of BT to:
  - Improve performance of downloads to T3?
  - Improve reliability by accessing multiple T2s for a single file?
  - Reduce effects of transient SRM and network loads at the T2s?
  - Easily share data among T3s?
- http://www.mwt2.org/~cgw/talks/t2t.html

BitTorrent protocol

- Metadata file ('.torrent') contains list of files, sizes and block checksums, and a “tracker” address.
- Client contacts tracker and is connected with one or more peers. Tracker does not move data.
- As soon as any blocks are downloaded and checked, client shares with other peers.
- Distinction between client and server blurred. A client which has all blocks is a 'seed'.
- Works well for large files (ISOs, movies, etc). Failed downloads can be resumed.
- Spec: http://wiki.theory.org/BitTorrentSpecification

Client software

- Very little to install/configure on client side
- .torrent file fetched using Web browser, curl, etc
- Many clients available, GUI (nice monitoring) and command-line (scriptable), interfaces for Python, Java, C++
- ctorrent: CLI client with nice features: select individual files for download, remote-control, daemon mode
- http://www.rahul.net/dholmes/ctorrent/

Possible advantages

- Less to install/configure, site does not need to be DQ2 endpoint
- Reduced load at T2s, since load is spread across multiple sites, and T3's can share data
- Resilient: Servers or sites can go down and transfers will continue
- All data integrity-checked (sha1)
- Possibly faster, depending on # seeds, CPU, network and disk speed

Disadvantages

- Uses more CPU
- Does not take advantage of multi-core hosts
- Ports may be blocked

Differences from common BitTorrent scenario

- Fewer peers / more files – cannot assume that data will be seeded
- Faster networking
- Multi-core/clustered hosts

Implementation

- Modified tracker: instead of returning empty list of peers, it requests seeds.
- Daemon process (bt-server) runs at T2 sites, listens for messages and starts seeds.
- Dynamic creation of metadata (about 4 minutes for 25GB dataset, so good to create these in advance)
- use LD_PRELOAD to read directly from storage (/pnfs).

Implementation, cont'd

- Server creates directory populated with symlinks, so that:
  - Dataset looks like single directory
  - Filename = LFN (no suffixes)
  - Optimization: symlink to file on local disk pool when possible

Example

DPD_IDCOMM.051203._00001.pool.root.1 ->
/pnfs/uchicago.edu/atlasdatadisk/data08_cos/DPD_IDCOMM/data08_cos.00091949.physics_IDCosmic.merge.DPD_IDCOMM.o4_r653_p26_tid051203/DPD_IDCOMM.051203._00001.pool.root.1

DPD_IDCOMM.051203._00002.pool.root.1 ->
/pnfs/uchicago.edu/atlasdatadisk/data08_cos/DPD_IDCOMM/data08_cos.00091949.physics_IDCosmic.merge.DPD_IDCOMM.o4_r653_p26_tid051203/DPD_IDCOMM.051203._00002.pool.root.1

DPD_IDCOMM.051203._00003.pool.root.1 ->
/pnfs/uchicago.edu/atlasdatadisk/data08_cos/DPD_IDCOMM/data08_cos.00091949.physics_IDCosmic.merge.DPD_IDCOMM.o4_r653_p26_tid051203/DPD_IDCOMM.051203._00003.pool.root.1

DPD_IDCOMM.051203._00004.pool.root.1 ->
/pnfs/uchicago.edu/atlasdatadisk/data08_cos/DPD_IDCOMM/data08_cos.00091949.physics_IDCosmic.merge.DPD_IDCOMM.o4_r653_p26_tid051203/DPD_IDCOMM.051203._00004.pool.root.1

DPD_IDCOMM.051203._00005.pool.root.1 -> /dcache/pool5/data/00060000000000000017F53C0

DPD_IDCOMM.051203._00006.pool.root.1 -> /dcache/pool3/data/00060000000000000019D5970

DPD_IDCOMM.051203._00007.pool.root.1 -> /dcache/pool6/data/00060000000000000019D5A18

DPD_IDCOMM.051203._00008.pool.root.1 ->
/pnfs/uchicago.edu/atlasdatadisk/data08_cos/DPD_IDCOMM/data08_cos.00091949.physics_IDCosmic.merge.DPD_IDCOMM.o4_r653_p26_tid051203/DPD_IDCOMM.051203._00008.pool.root.1

DPD_IDCOMM.051203._00009.pool.root.1 -> /dcache/pool1/data/00060000000000000017F5308

DPD_IDCOMM.051203._00010.pool.root.1 -> /dcache/pool5/data/00060000000000000017F5680

DPD_IDCOMM.051203._00011.pool.root.1 ->
/pnfs/uchicago.edu/atlasdatadisk/data08_cos/DPD_IDCOMM/data08_cos.00091949.physics_IDCosmic.merge.DPD_IDCOMM.o4_r653_p26_tid051203/DPD_IDCOMM.051203._00011.pool.root.1

Tests

- Test dataset: 25 GB, 29 files
- 3 seeds running on uct2-s[1-3] (Dell 2950 storage servers, 8 core, 32GB, 10Gb NIC)
- Fetch to uct3:
  - dq2 takes 22 minutes (19MB/sec)
  - bittorrent takes 29 (15 MB/sec)
- Dq2 is faster, but:
  - It's using more cores (2 lcg-cp's active)
  - It's not validating data (other than file size)

More tests at uct3

- How to get multiple transfers running on same host?
- Fetch to 4 WNs, each receiving ¼ of the files
  - Finished in 3,5,5 and 6 minutes
  - Overall rate: 70MB/sec (using longest time)

Tests at IllinoisHEP (UIUC)

- Fetch to single host:
  - dq2-get: 22 minutes (19MB/sec)
  - Bittorrent: 26 minutes (16MB/sec)
  - Disk write speed: ~33MB/sec
- 4-host test
  - Finished in 7,9,15,18 minutes (24MB/sec)

Next steps

- Deployment and more testing of bt-server
- Repository for .torrent files, browse/search for datasets
- Investigate slowdowns, bottlenecks, tuning
- Multi-threaded transfers on single host
- Splitting of transfers across cluster
- Single command 'bt-get' that gets .torrent file and starts client(s)