ANSE and PhEDEx

Use of Dynamic Circuits in PhEDEx, the CMS data distribution management framework
Overview

- Quick PhEDEx (101)

- Current development efforts regarding dynamic circuits
  - Where (& how) it can be integrated
  - Software architecture
  - Initial results

- Next steps
PhEDEx Overview

Loosely coupled set of agents written in Perl interacting via central DB
  • central agents (ex. FileRouter agent)
  • site agents running at various T1s and T2s (ex. FileDownload agent)
  • each agent performs a single task

*FileRouter* (central) agent builds transfer queue per destination
*FileDownload* (site) agent examines its queue, processes it & reports back

ANSE efforts in circuit awareness integration:
1. In the FileDownload agent:
   - Great compromise between desired functionality and complexity
     (chosen for prototyping)
   - Only has a local view

2. In the FileRouter agent:
   - Has a global view of the whole system
   - Harder to implement and optimize
Integrating circuit awareness in the FileDownload agent – inner workings

Standard FileDownload agent:
• Files from the transfer queue are grouped into transfer jobs
• Jobs are handed to the transfer backend (FDT, FTS, etc…) for execution
• Transfer backend reports back with transfer status
• FileDownload agent reports back to DB

Updated FileDownload agent (CircuitAgent):
• Determines whether a circuit is worthwhile and requests one if it is
• Circuit request goes via the CircuitManager
• When a new transfer job is ready to start
  • Checks if a circuit is available (via CircuitManager)
  • Updates job to use circuit instead of GPN

CircuitManager agent manages the lifecycle of the circuit
Class diagram

- FileDownload agent
  - Extends
  - Circuit Agent
    - CircuitManager
    - HTTPServer
  - Circuit
    - 0..n
  - Backends
    - External
    - DynesStates
    - HTTPClient
      - Dynes
      - Dummy
      - ML
      - Core
        - IDC
  - Extend
  - 1..1
  - 1..n
**Sequence diagram**

- **CircuitAgent**
- **CircuitManager**
- **Backend**

**checkWorkload**

60s

**shouldRequestCircuit**

for all paths which allow & benefit from circuits

**requestCircuit**

**canRequestCircuit**

**checkLinkSupport**

**backendRequestCircuit**

**requestTimeout**

**handleRequestResponse**

**teardown**

**backendTeardownCircuit**

**verifyStateConsistency**

60s

**transferTask**

**isCircuitAvailable**

**circuit (if available)**

if

- [Circuit exists]
  - Transfer of file takes place on circuit (circuit IPs used)
- [No circuit]
  - Transfer of file takes place on default route (TFC IPs used)

**CircularManager**

**Backend**
PhEDEx integration of BoD (initial results)

Using shared path

- Link throughput (data points @ 1 min)
- Moving average over 1h

Background Iperf traffic @ 5Gbit/sec

Link saturation
Using dedicated circuit

- Link throughput (data points @ 1 min)
- Moving average over 1h
PhEDEx integration of BoD (initial results)

- **Seamless path switch**
- Per job link rates with PhEDEx traffic
  - ~620MB/sec -> 1060 to 1250MB/sec
- Average link rates with PhEDEx traffic
  - ~570MB/sec -> ~1050MB/sec
- Differences between per job and avg. rates
  - Delay in starting jobs
  - Sustained 10G to disk sometimes challenging

**Phedex reported rates**

**PhEDEx Transfer Rates (in MB/sec)**

Limited by background traffic
Future plans

ANSE timetable:
• Official project deadline: 1 Jan 2015
• Potential extension: ~ August 2015

Plans on circuits:
• Finish circuit integration at the link level
  • Iron out remaining bugs
  • Continue testing on private testbed
  • Deploy in CMS for real-life testing *(subject to circuit availability)*
  • Knowledge transfer to ATLAS team
• Prototype circuit integration at the instance level
Thank you!
Backup slides
Integrating circuit awareness in the FileDownload agent – inner workings

More about the FileDownload agent:
• Agents are event driven (POE)
• Transfer tasks are assigned to it (one file = task)
• Each task contains the source and destination PFNs
• Tasks are bulked into transfer jobs (# of tasks/job varies)
• This job is handed to the transfer backend (FDT, FTS, etc...)
• Each task in the job goes through the following flow
  • start_task: starts the transfer workflow for a task
  • transfer_task: marks a task as ready to transfer
  • transfer_done: saves task status and transfer info
  • finish_task: marks task as completed
• When all tasks in a job pass “transfer_task”, the backend begins transfer
• Backend reports back with transfer status

If we want to use other paths for transfers, we need to modify the PFNs*
• best place to do that is to insert our code into transfer_task

*PFN = protocol://hostname[:port]/[partial_path_to_root]/LFN
Initial results

Two 10G links/circuits:
- **Link1**: shared path
  - Background traffic @ 5Gbit/sec UDP generated by Iperf
- **Link2**: dedicated circuit for PhEDEx

Sandy01 gva/ams:
- dual cpu E5-2670 (2x8 physical cores with HT)
- 64GB ram
- 2 LSI controllers with 8 SSDs each
  - 4 x 4 SSD RAID 0 arrays
  - 1 LSI controller (2x4 SSDs) used for ANSE tests

Monitoring link performance with MonALISA