4/25/11

Condor and the Hadoop File System with a Packing Problem

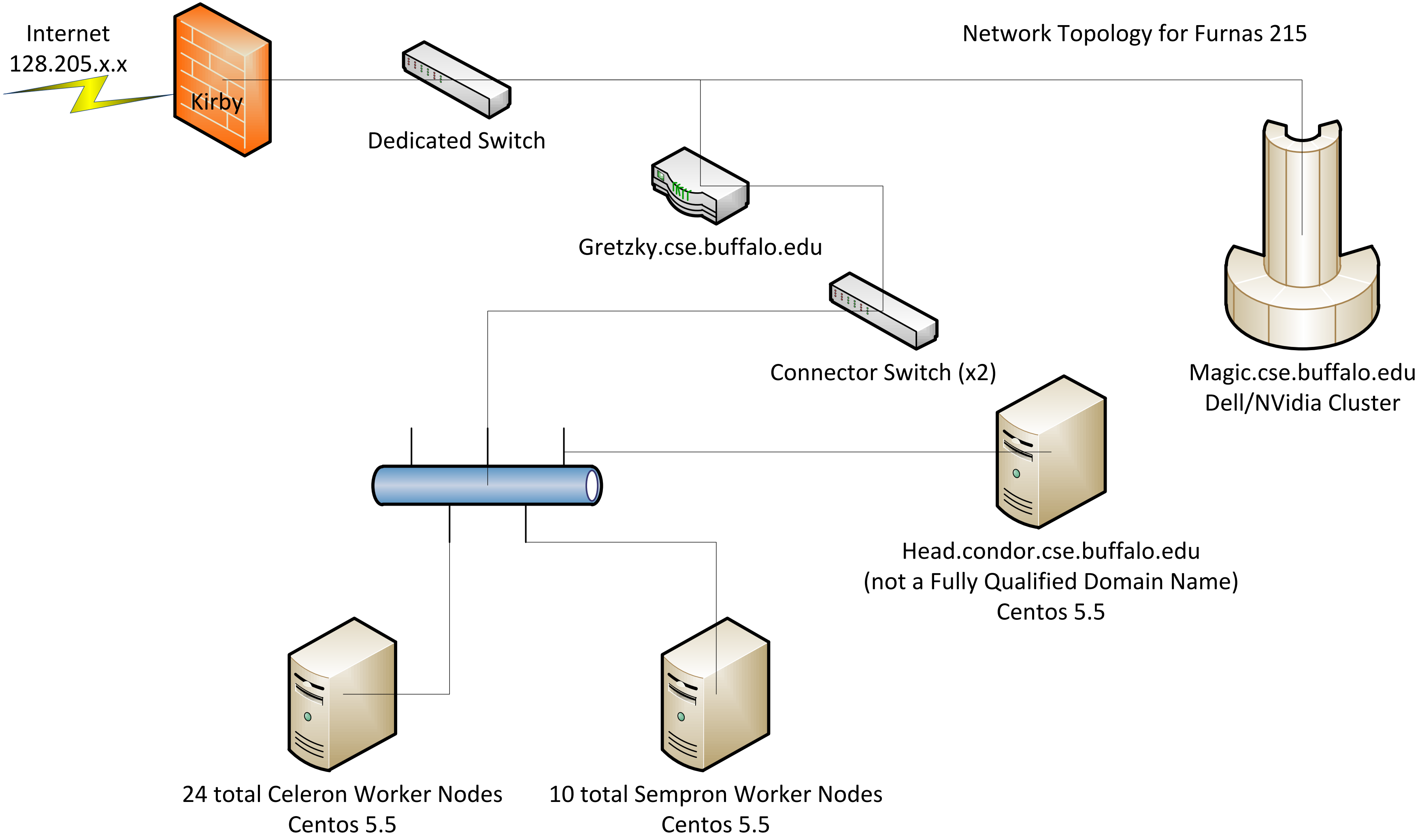
CSE 704 – Parallel Computing Seminar

Presenter: Philip Matuskiewicz

The Lab (Furnas 215)

- 35 personal computers
  - Will now be referred to as nodes
  - 10 AMD Semprons with 1GB of ram
  - 20 Intel Pentium Dual Core Processors w/ 1GB or 2GB of ram
- Connected to a router (Gretzky.cse.buffalo.edu)
  - performs NAT
  - Internal nodes connected on 192.168.0.0/24 subnet
- 1 node dedicated as head node
  - Head.condor.cse.buffalo.edu
  - Runs necessary services to support the flock
  - Contains a network file system and the hadoop file system namenode
- 34 nodes dedicated as worker nodes (for processing)

The Lab



Internet
128.205.x.x

Kirby

Dedicated Switch

Network Topology for Furnas 215

Gretzky.cse.buffalo.edu

Connector Switch (x2)

Magic.cse.buffalo.edu
Dell/NVidia Cluster

Head.condor.cse.buffalo.edu
(not a Fully Qualified Domain Name)
Centos 5.5

24 total Celeron Worker Nodes
Centos 5.5

10 total Sempron Worker Nodes
Centos 5.5

Video from the lab

The head node functions as:

- Network File System – “/network”
- NIS (Network Information Services)
  - Central login / password management
- Internal DNS (Domain Name Services)
  - Lookups such as head -> 192.168.0.10
- Internal DHCP (For ghost cast)
- Condor management node
  - Handles submissions of Condor Jobs to nodes on the Condor Flock
- Hadoop Namenode
  - Contains a lookup table mapping files to the nodes that hold those files
- Central Email Manager for the nodes on the flock

What is Condor

- Grid computing
  - Installed to existing nodes
    - Nodes may be on multiple networks
      - Nodes on a single network is referred to as a Condor Pool
      - One Condor flock can contain many pools
      - A central manager node does the match making to put jobs onto free nodes within a Condor flock
    - Runs jobs during idle time on these workstations
    - Cost efficient – dedicated compute nodes not required!
    - Implements Failover Mechanisms Well
  - High Throughput Computing
    - High data movement over a long period of time
    - Network latency isn't a major factor
- Developed at the University of Wisconsin-Madison

Condor Commands

- Condor_submit – submit a job to the condor queue
- Condor_q – view jobs submitted to the condor queue
- Condor_rm – remove jobs from the condor queue
- Condor_compile – compile a program with the Condor libraries for submission as a Condor Job
  - For status checking during a job execution
  - Supports many variants of C/C++ and Fortran
- Can be ran from any node in a Condor Flock

Condor Services

Installed on all nodes

- MASTER – Monitors the other daemons
- STARTD – responsible for starting / running a job
- SCHEDD – responsible for queuing jobs to be ran on the node

Installed on the Head node:

- COLLECTOR – collects the status information of all the nodes on a Condor flock
- NEGOTIATOR – Places jobs onto nodes as availability permits

Example Condor Submission File

```
Universe = standard
Executable = pack
Output = pack.out
Log = pack.log
arguments = 11
Requirements = ( Memory > 0 && TotalMemory >= (512) )
Notification = Never
queue
```

Condor results

- Condor executes a job on other nodes
  - Scripts that have dependencies on certain local files are impacted (hence NFS)
- The job runs on the next available node
- Output from STDOUT is placed into the output file
- An email is sent to the user if notification is enabled

What is Hadoop

- Distributed by the Apache Foundation
- Software framework that supports data-intensive distributed computing
  - Inspired by Google's File System and MapReduce
- Implemented on Sun's version of Java
- Highly Scalable

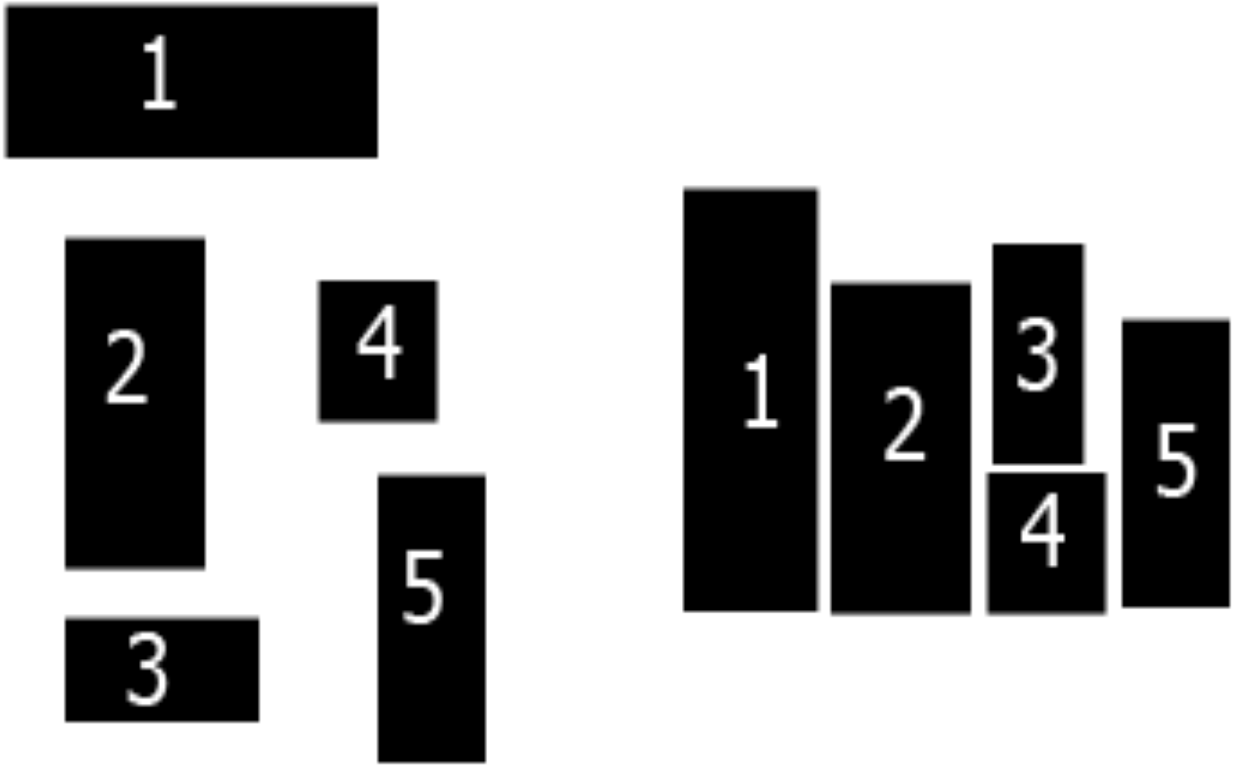
Hadoop File System (HDFS) in the lab

- NFS at "/network" is centralized
  - Bottleneck on the network, Condor is writing / reading from NFS concurrently
- The Hadoop File System is a Distributed File System
  - Files are replicated to all of the worker nodes
    - All the files are locally available
  - Technically, the network bandwidth should be eliminated!
- The HDFS is mounted as "/hdfs" using fuse-dfs
  - Enables normal directory access to the HDFS

The 2-D Bin Packing Problem

- Take many rectangles of random sizes and place into bins of a uniform size
  - Minimize the number of bins (optimization)
  - Parallel and Orthogonal trajectory only
    - 0 and 90 degree rotation
- Problem:
- Take 340,000 rectangles and optimize them
  - 10,000 rectangles per node
  - Rectangles generated randomly

The 2-D Bin Packing Problem Pictured



Algorithm Overview For the Lab

- Generate 340,000 rectangles and place 10,000 per file (34 files)
- On each worker node, optimize 10,000 rectangles as follows
  - For each rectangle (represented as an object), get the width and height
    - Take the smaller of the width and height and iterate through the bins, if a bin can hold the rectangle, place the rectangle in the bin
    - Update the rectangle's orientation and the bin's availability
    - Move onto the next rectangle
- Algorithm described is the First Fit algorithm from Garey and Johnson's book – The Thory of NP-Completeness

Results on NFS vs. HDFS

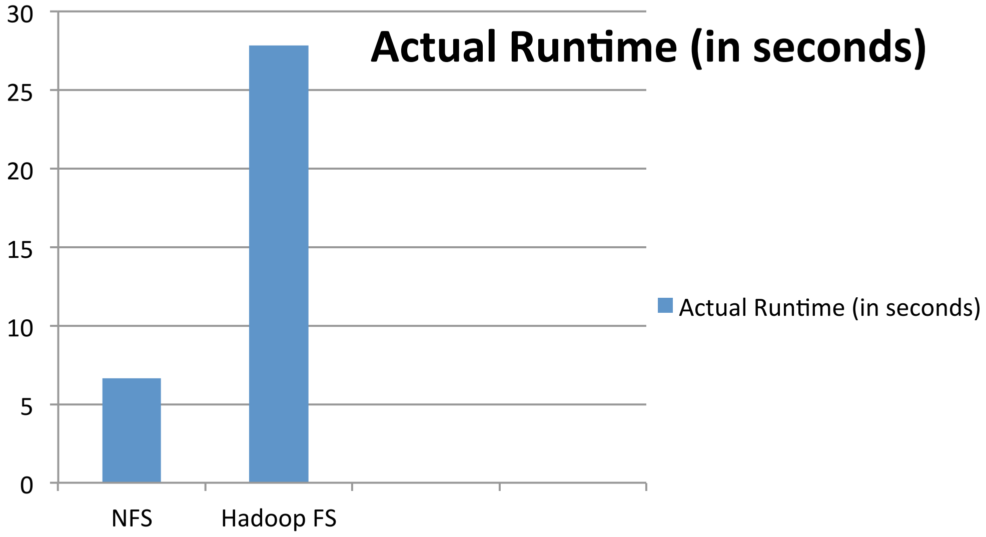
NFS

- Bins: 42118
- Width: 1868356
- Height: 50
- Area: 93417800
- Time (Seconds): 6.665628

HDFS

- Bins: 41644
- Width: 1747826
- Height: 50
- Area: 87391300
- Time (Seconds): 27.86374

-Time calculated using C timer accurate to milliseconds.
- Files and algorithm executable stored on the file system directly



Unexpected Results?

- All the files are local!
  - The problem is that Fuse-DFS was accessing them
    - Hadoop has internal mechanisms that schedule jobs to run on nodes where the files are present and those mechanisms instruct the machines to locally access the required files for the job runs
    - There was an abnormal amount of network traffic from the worker nodes, logs show massive file requests across the entire Condor flock during the HDFS test run
    - Hadoop is ideal only when programming and submitting using its built in mechanisms
- HDFS does not support the executable bit, applications cannot be ran directly from HDFS.
  - Work around used for this project was to copy the packing algorithm executable to the /tmp directory and execute it there, all the other files were directly accessed on the HDFS.

Suggestions

- Copy the required files into the /tmp or / scratch directory and run from there, then transfer back to NFS or HDFS

Sources

- Phil’s Paper Titled - Using Condor to solve a Bin Packing Problem
- http://hadoop.apache.org/
- http://www.cs.wisc.edu/condor/

Questions?