distCVS: A Distributed Peer-to-Peer Versioning File Storage System

Andrew Logan

May 13, 2005

Advised by:
Professor Robert Signorile
Professor Elizabeth Borowsky

Thanks to:

My family, for always encouraging me to

Contents

1 Abstract

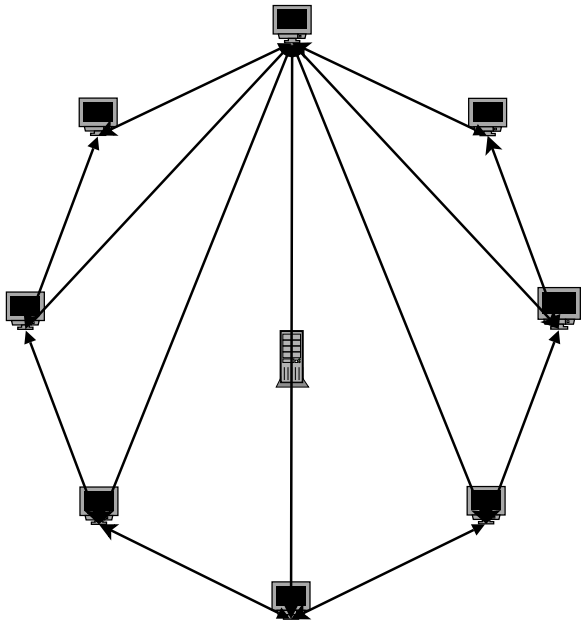
The current layout of resources on the internet su□ers from the fact that there is generally a single point of failure for data access. Peer-to-peer applications promise to change this by distributing resources, and therefore network routes and load, across the network itself. The problem is that the development and testing of such a system is often a hard and tedious chore, especially since the technology is still rapidly evolving. In this paper, I present distCVS, a peer-to-peer system for the storage of versioned data. distCVS is unique because it is an application built by using the Bamboo peer-to-peer networking framework to pass messages for an unmodified version of the CVS source controlun4TBT11.95520001220011.9552TcET.1804con

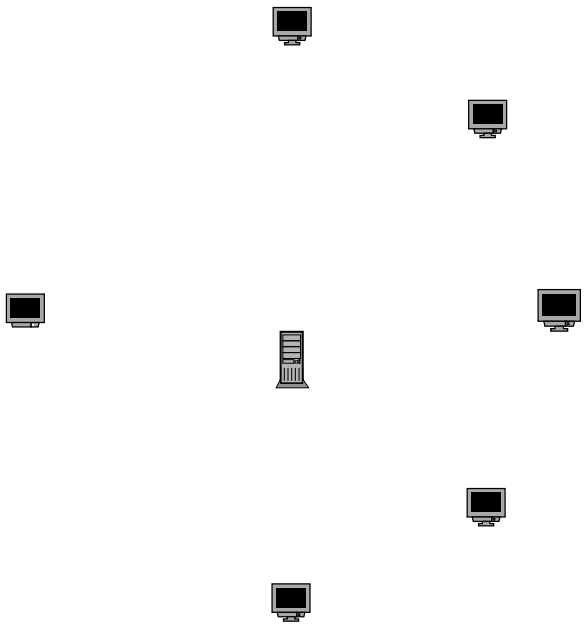
2 Introduction

The basic layout of the internet has changed very little since its inception in the late 1960's. In essence, the resources of the internet are laid out in a star topology. Client computers spread across the internet access resources contained

puter on a network to every other one on the network. Although this scheme does provide complete connectivity, it is not practical for several reasons. First, there is a limit to how many network connections a computer can have active at one time, which places a relatively small upper bound on the maximum size of the network.[1] This topology was attempted with the first version of the Gnutella peer-to-peer network, and the designers of that system soon discovered that the information that needs to be shared to index the data on this network imposed an enormous amount of networking overhead, which severely limited the amount of bandwidth that could be used for sharing data.

The popularity of Napster was partly due to the fact that it solved the indexing problem by using a central server to keep track of the resources available on the network.[2] Although this is a clever solution to the indexing problem, the addition of a central server makes this system not truly peer-to-peer. Therefore, this network still has some of the scalability and reliability problems that star networks do. This napster topology is better than a





easily recreated. Finally, since Bamboo and CVS are both freely av

3 System Overview

The distCVS system allows users to put, get and update files in a peer-to-peer network, and handles all versioning and concurrency enforcement in the background. The typical user of the system would provide it with the name of a project and a file that he or she wanted to work on, make changes to it, and then resubmit it into the network. If another user had submitted changes while this copy was being worked on, the system will notify the user that his or her file is out of date and needs to be updated before the changes will be accepted.

Each node in a

4 Implemen

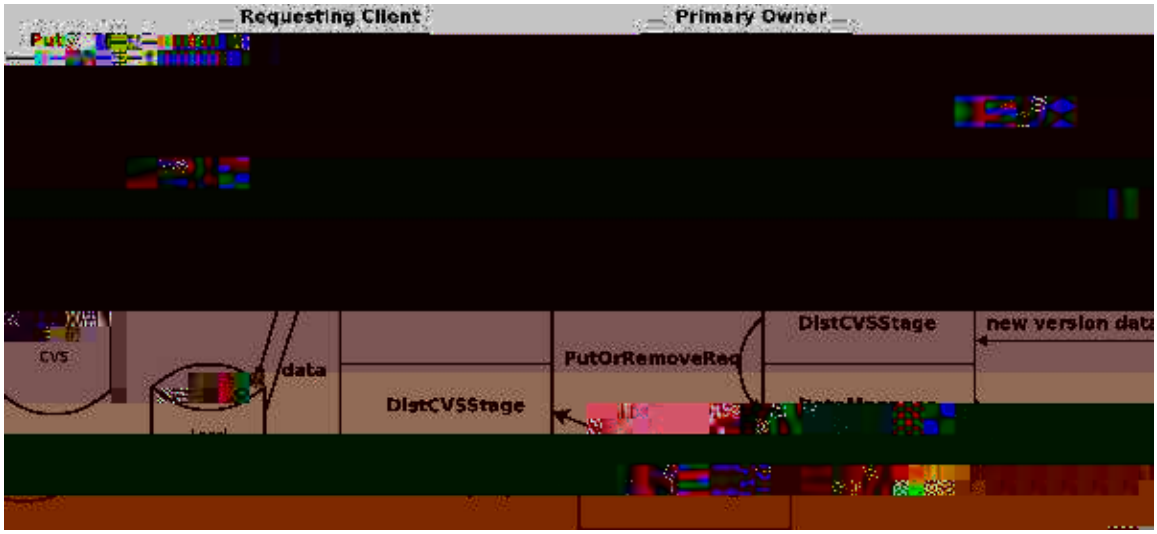


Figure 5: *An updated Coordinated Put operation*

implement deletion of old resources, extra steps need to be taken to iterate through every file referenced by a given GUID, to ensure that the most recently stored copy is retrieved from the primary owner's database. If no matching data is found, then the system assumes that the submitted data is being added for the first time. Otherwise, it is treated as an update.

The retrieved version data (if it

new version data from the CVS repository and indicates that the process completed successfully. DistCVSStage then retrieves this file from local storage

there is no way to ensure that any given node has a globally correct piece of data. Therefore, we were forced to assume that the correct version would quickly propagate

is

5 Testing

Our tests sought to answer several questions about the behavior of DistCVS. Specifically, we wanted to see how stable the system was under a high rate of file requests, or a high rate of nodes joining and dropping from the netw

In order to ensure that nodes received the correct copy of the data from the network, a master copy was kept on a file system shared by all of the test machines. Whenever a response was returned, it was compared against this master copy using the Linux di utility. If the files were identical, a GOOD_VERSION response was

number of nodes	total requests	%good	%bad	%not found	%unanswered
100					

6 Conclusions and Future Work

Due to the modular structure of Bamboo and the way that CVS interacts with data, distCVS demonstrates that it is possible to build a complex peer-to-peer system that can be rapidly upgraded to reflect any possible future advances in the technology. Although distCVS is a working system, there are still several areas where it could be improved.

distCVS is only a partial implementation of CVS, so although the frame-w

```
import bamboo.api.BambooRouteDeliver;
import bamboo.api.BambooRouteInit;
import bamboo.api.BambooRouterAppRegReq;
import bamboo.api.Ba
```

//////////

```
this.filesize = filedata.length;
System.out.println("PutCVSReqPayload: constructor");
debug_disp_file(this.filename, this.projectname, t
```

```
//Sent to primary owner to initiate retrieval of data from the
//storage net
```

```
//I keep track of the length here since I get an underflow
//error if I try to take more data fr
```

/

a y

Syst em.out.pri ntln(line); s
}

command = "./storehelper.pl ";
resultingFilename = initialFilename + ",v";

Process proc = runtime.exec(command + initialFilename + " > storehelperoutput");

//check for failure
if(proc.waitFor() != 0){
    Syst

a y t

/ a
/

```
        client = (GatewayClient) lookup_stage (config, client_stg_name);

        automatic_test = config_get_boolean(config, "automatic_test");
        updater_node = config_get_boolean(config, "updater_node");
        initial_data = config_get_boolean(config, "initial_data");
        show_gui = config_get_boolean(config, "show_gui");
        automatic_data_submission_hack = config_get_boolean(config,
"automatic_data_submission_hack");
        System.out.println("DistCVSStage: initial_data
```

```
System.out.println("DistCVSStage: got BambooRouterAppRegResp");
BambooRouterAppRegResp  resp = oR
```

```
                    classifier.dispatch_later(new BambooRouteInit(info.src, app_id, false, false,
new StatusPayload("Initializing database - your request will be processed shortly")), 0);
                }
            }

            //WarmupCompletedAlar
```

```
                        debug_disp_file(info.filename +",v", info.projectname, version_filedata);

                        //I don't need this, since I can just append it to the reconstructed data.
                        //version file
                        //write_file(info.filename + "_version", info.projectname, version_header);

                        //,v file
                        write_file(info.filename + ",v", info.projectname, version_filedata);

                        Runtime runtime = Runtime.getRuntime();

                        //does the requisite CVS commands

                        String command = "./reconstructhelper.pl ";

                        Process proc = runtime.exec(command + info.filename + ",v" + " >
helperoutput" );

                        //check for failure
                        if(proc.waitFor() != 0){
                            System.err.println("reconstructhelper exit value = " +
proc.exitValue());

                            //TODO: write new exception that passes error code.
                            throw new InterruptedException();
                        }

                        //So far so good.  Read the newly reconstructed file off of the disk.
                        byte
```

```
 */
else if(elem.user_data instanceof PutCVSReqInfoContainer){
    PutCVSReqInfoContainer info = (PutCVSReqInfoC
```

```
                        debug_disp_file(info.filename +",v_version",  info.projectname,
new_version_header);
                        debug_disp_file(info.filename +",v",  info.projectname, new_version_data);
                        debug_disp_file(info.filename +",v",  info.projectname,
submitted_version_filedata);

                        //Place the version data into
```

```
//Strip out the header that's included in info.filedata
byte[] master_version_hea
```

```
                    debug_disp_file(pay.filename + "_version", pay.projectname,
pay.version_filedata);

                    tr
```

```
    //done!
    fos.close();
}

protected byte[] read_
```

```
//Testing only.  Will modify a specific file and then place the
//updated data into the network.
protected void handle_submit_test_data_alarm(){
    try{
        Runtime runtime = Runtime.getRuntime();
        Process proc = runtime.exec("./appendfiledata.pl " + test_filename);

        if(proc.waitFor
```

```
{
    projectname = projectnameField.getText();
    filename = filenameField.getText();
    //If we're decvs-ifying the file, add the
    //
```

Listing 2: PrintDataStage.java

```
/*
 * PrintDataStage.java - a Bamboo stage that displays the GUIDs of
 * data currently stored on the node.
 *
 * Copyright (C) 2004,2005 Andrew Ian Logan
 * logana@bc.edu, andrewlogan@gmail.com
 *
 */

package thesis;

import java.math.BigInteger;

import java.nio.ByteBuffer;

import java.security.MessageDigest;

import java.util.Iterator;
import java.util.Random;
import java.util.LinkedList;

import ostore.util.ByteUtils;
import ostore.util.ByteArrayOutputBuffer;
import ostore.util.Pair;
import ostore.util.ConBu
```

```
int debug_level = config.getInt(
```

if

Listing 3: submithelper.pl

```
#
```

```
}
#force cop
```

Listing 4: reconstructhelper.pl

```
#!/usr/bin/perl
# Andrew Logan
# gethelper.pl
# 10/22/04
#
# Written because making a pile of system calls in Java is a hassle.
# This script will reconstruct a CVS repository, and then extract a
# file from it.  It leaves <filename> in the directory it was
# called from.

# Here is what we need to do:
# Make a CVSROOT
# run "cvs init" in it
# make a repository directory in it
# copy our input ,v files there (CVS only reconstructs files ending in ,v)
# run "cvs co repository
```

8 Appendix B: Test Scripts

Listing 5: go_big

```
#!/usr/bin/perl

# go_big
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com
# 3/14/05

# Written to automate the process of doing multiple tests of the
# DistCVS system.

$hostname = `hostname`;
chomp($hostname);

$start_experiment = 15;
$end_experiment = 15;

for ($i = $start_experiment; $i < $end_experiment+1; $i++){
    if ($i > $start_experiment){
        print "*****WAITING TO BEGIN EXPERIMENT $i*****\n";

        sleep(10 * 60);
    }

    print "*****BEGINNING EXPERIMENT $i*****TjETBT5.97758005.9775817556.2152468.4801Tm/Tc17758005.978005.97758328.1T5.97x
```

Listing 6: go

```
#!/bin/bash


# go
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com

# Starts a single test of the Bamboo system.


./run_local_nodes.pl -nodes=10 -with_updater -with_gateway
```

Listing 7: run_local_nodes.pl

```
#!/
```

```
#$gateway_pause = 1 * 60;
$gateway_pause = 15;
#$gateway_pause = 0;

#$inter_node_pause = 30;
$inter_node_pause = 15;
#$inter_node_pause = 0;

#in se
```

i

```
sleep (
```

```
        if($liveport != -1){
            #‘echo $message >> $hostname:$liveport/diff_results.log‘;
            log_to_node($message, $liveport);
        }
    }
}

sub log_to_node{
    my ($message, $portnum) = @_;
    #print "log_to_node: message: $message portnum: $portnum\n";
    ‘echo $message >> $hostname:$portnum/diff_results.log‘;
}
```

Listing 8: parse_cfg_and_run.pl

```
#!/usr/bin/perl

# parse_cfg_and_run.pl
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com
# 12/6/04
            n
# This perl script will parse through a Bamboo configuration fi
```

```
if($hide_gui){
    $show_gui_var = false;
}
else{
    $show_gui_var = true;
}

if($do_adsh){
    $automatic_data_submission_hack = true;
}
else{
    $automatic_data_submission_hack = false;
}

$data_manager_d
```

```
    cleanup();
    exit(1);
}

#clear the temp files from the experiments
#‘rm diff_re
```

Listing 9: modified_run_

```
debug_level 0
logfile_name          diff_results.log
automatic_test        true
u
```

Listing 11: make_global_experiment_stats.pl

```
#!/usr/bin/perl
#
# make_gk
```

Listing 12: process_and_send_results.pl

```
#!/usr/bin/perl

# process_and_send_results.pl
# Andrew Logan
# logana@bc.edu, andrewlogan@gmail.com
# 3/15/
```

9 Appendix C: Bibliography

References

[1] B. Berliner. CVS, The Concurrent Versioning System . http://www.gnu.org/manual/cvs1.9/html_chapter/cvs_2.html.

[2] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, and B. Wiley. Protecting free expression online with freenet. *n , , n Co p f n* , 6(1):40--49, 2002.

[3] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *o* , May 2001.

[4] S. P. Ratnasamy. *A c , Con , n . Add , , N o* . PhD thesis, Dept. Comp. Sci., University

and Internet Use in the United States: August 2000. http://www.census.gov/prod/2001pubs/p23-207.pdf, 2001.