HDF5 Performance Framework User Manual

Hyo-Kyung Lee

The HDF Group

HDF5 Performance Framework User Manual
by Hyo-Kyung Lee

Table of Contents

Chapter 1. Introduction to HPF

This HDF5 Performance Framework (HPF) user manual was created to help users quickly learn how to test HDF5 library performance. It covers the basic concept and usage of HPF.

1.1 What is HDF5 Performance Framework(HPF)?

HPF is a free performance testing system for HDF5 library and its applications. Although this project is aimed for HDF5, HPF is a very general-purpose benchmarking tool and can be applied to any applications.

It measures the time that custom benchmark programs took via `gettimeofday()` function call. It doesn't measure any disk or memory space usage yet. And it has a web-based graphical user interface for analysis of the measured time data.

The HPF testing model is to run benchmark programs daily. Any changes in HDF5 library may or may not affect the daily performance of the existing benchmark programs and the goal of HPF is to monitor the long term performance of HDF5 library. Also, since HDF has different versions of libraries and runs on different platforms, we made HPF allow testing multiple versions on multiple platforms.

1.2 On what systems does HPF run?

As of September 2009, HPF runs on the following platforms:

- 32-bit Linux:kernel 2.6.x
- Solaris 5.10
- 64-bit Linux:kernel 2.6.x

In general, it can run on any UNIX platform that supports HDF5 library.

1.3 Why should I use HPF?

HPF can provide a quick way of identifying not only HDF library compilation problems on many platforms but also potential performance issues of the library. HPF generates a daily warning report if there's a compilation error or a benchmark takes longer than 20% or more. Users can quickly locate the problem by looking at the daily performance graph. HPF also records the subversion revision number used in the library so the library developers can quickly find the source code that caused the performance degradation.

Thus, HPF helps HDF5 developers to improve HDF5 library performance by providing an interface for quickly spotting the performance issue in their new codes. It is much better to know the potential performance problems as soon as possible instead of realizing a significant performance problem after a couple of weeks, months or years.

1.4 What is included with HPF?

The HPF has three parts: back-end, core and front-end.

The back-end part is about how to run the tests and how to store the results. It has a shell script and several PHP scripts and all scripts run as as UNIX cron jobs daily. It uses a MySQL database for the long term storage of performance result data.

The core part is C/C++ performance framework library. This is a collection of wrapper functions that benchmark programs should be compiled and linked together.

The front-end part is about performance data analysis and visualization. It has PHP and JavaScript scripts including jpgraph package for plotting graphs.

HPF requires a number of third-party software products, including:

- Apache
- jpgraph
- MySQL
- PHP
- gd
- subversion
- bash

The jpgraph is included in the HPF distribution.

Chapter 2. Getting to know HPF

2.1 Web Pages

The official website for the HPF project is located at: http://hdfdap.hdfgroup.uiuc.edu/h5perf/index.html.

2.2 Reporting Bugs

Before crying "Bug!", please make sure that the problem is really relevant to HPF. We cannot fix bugs that belong to third-party software products like MySQL and PHP. Also, please try to run **svn update** on your source tree to see if the bug is already fixed.

Proper bug reporting is one of the most important responsibilities of end users. Very detailed information is required to diagnose most serious bugs and please identify and include the subversion revision number when you send a bug report to: <help@hdfgroup.org>

Chapter 3. HPF Installation Guide

3.1 Pre-installation checklist

Before installing HPF, you need to make sure that the following programs are already installed:

- svn
- MySQL database
- PHP-enabled web server with gd library
- PHP with command line interface and MySQL module

Of course, you need all the necessary software including GNU autotools and a C/C++ compiler to build HDF5 library.

The first step of using HPF is to get the HPF source codes from subversion repository.

```
%svn co http://svn.hdfgroup.uiuc.edu/hdf5perf
```

3.2 Creating Database and Tables

HPF uses a MySQL database as a storage option for HDF5 performance measurement results and system environment variables. The `hdf5perf/hdf5perflib/lib/cpp/DB.sql` file contains a set of SQL commands that create necessary tables.

Issuing a command like

```
%mysql -u root -p < hdf5perf/hdf5perflib/lib/cpp/DB.sql
# Replace 'root' with your DB admin username
```

and entering your database password will create a new database called `hdf5perfsimple` and all 9 tables necessary for HPF under the new database.

3.3 Modification of Source Codes

HPF distribution comes with a default database username `root` and no password in all its source codes and it is necessary to change them if you use a different username/password pair on your MySQL database.

Here is the list of files that you need to modify:

- `hdf5perf/hdf5perflib/libtest/db.h`
- `hdf5perf/hdf5perflib/examples/db.h`
- `hdf5perf/hdf5perflib/raw-data/db.h`
- `hdf5perf/hdf5perflib/freespace/db.h`
- `hdf5perf/hdf5perfphp/hdf/includes/data.inc.php`

3.4 Customization of Shell Scripts

Three shell scripts need customization to ensure that they can perform daily tests successfully on time. Here are the three shell scripts you need to modify:

- `hdf5perf/hdf5perfphp/run_1.6.sh`
- `hdf5perf/hdf5perfphp/run_1.8.sh`

- hdf5perf/hdf5perfphp/run_1.9.sh

Please pay attention to the choice of compiler, compiler options and paths. Different operating systems may use different command line options for the same command.

In HPF, all shell script cron jobs should finish within a day (i.e., before 11:59pm) so that analysis and report PHP scripts can run on the next day in sync. The PHP scripts look for the data that have a time-stamp of specific range in MySQL DB. Although you can submit a cron job that starts early to ensure the job to be completed on time, it's also possible to save some time by adjusting the INTERVAL variable within the shell script.

Please look for the lines that say:

```
# This script will run tests 3 times to record the best performance.
# Please set the interval in seconds between trials.
# 600 seconds = 10 minutes
INTERVAL="600"
```

and make it shorter if necessary.

Finally, it is not recommended to modify the lines below the following comments:

```
#####################################################################
###########
# Please DO NOT edit lines below.
#####################################################################
###########
```

It is also a very good idea to test the customized script at the command prompt before submitting the cron job.

3.5 Cron Job Installation

Once all DB creation, modification and customization are done, you are ready to submit cron jobs for the following scripts:

- hdf5perf/hdf5perfphp/run_1.6.sh
- hdf5perf/hdf5perfphp/run_1.8.sh
- hdf5perf/hdf5perfphp/run_1.9.sh
- php hdf5perf/hdf5perfphp/hdf/sync.php
- php hdf5perf/hdf5perfphp/hdf/mailer.php

Here's an example cron job table listing on hdfdap.hdfgroup.uiuc.edu machine at The HDF Group:

```
00 18 * * * /hdf5perf/hdf5perfphp/run_1.6.sh
00 20 * * * /hdf5perf/hdf5perfphp/run_1.8.sh
00 22 * * * /hdf5perf/hdf5perfphp/run_1.9.sh
15 01 * * * /usr/bin/php /hdf5perf/hdf5perfphp/hdf/sync.php
55 01 * * * /usr/bin/php /hdf5perf/hdf5perfphp/hdf/mailer.php
```

The above example tells us that the HDF5 library version 1.6 testing starts at 6:00pm, HDF5 library version 1.8 at 8:00pm and version 1.9 at 10:00pm on hdfdap.hdfgroup.uiuc.edu machine.

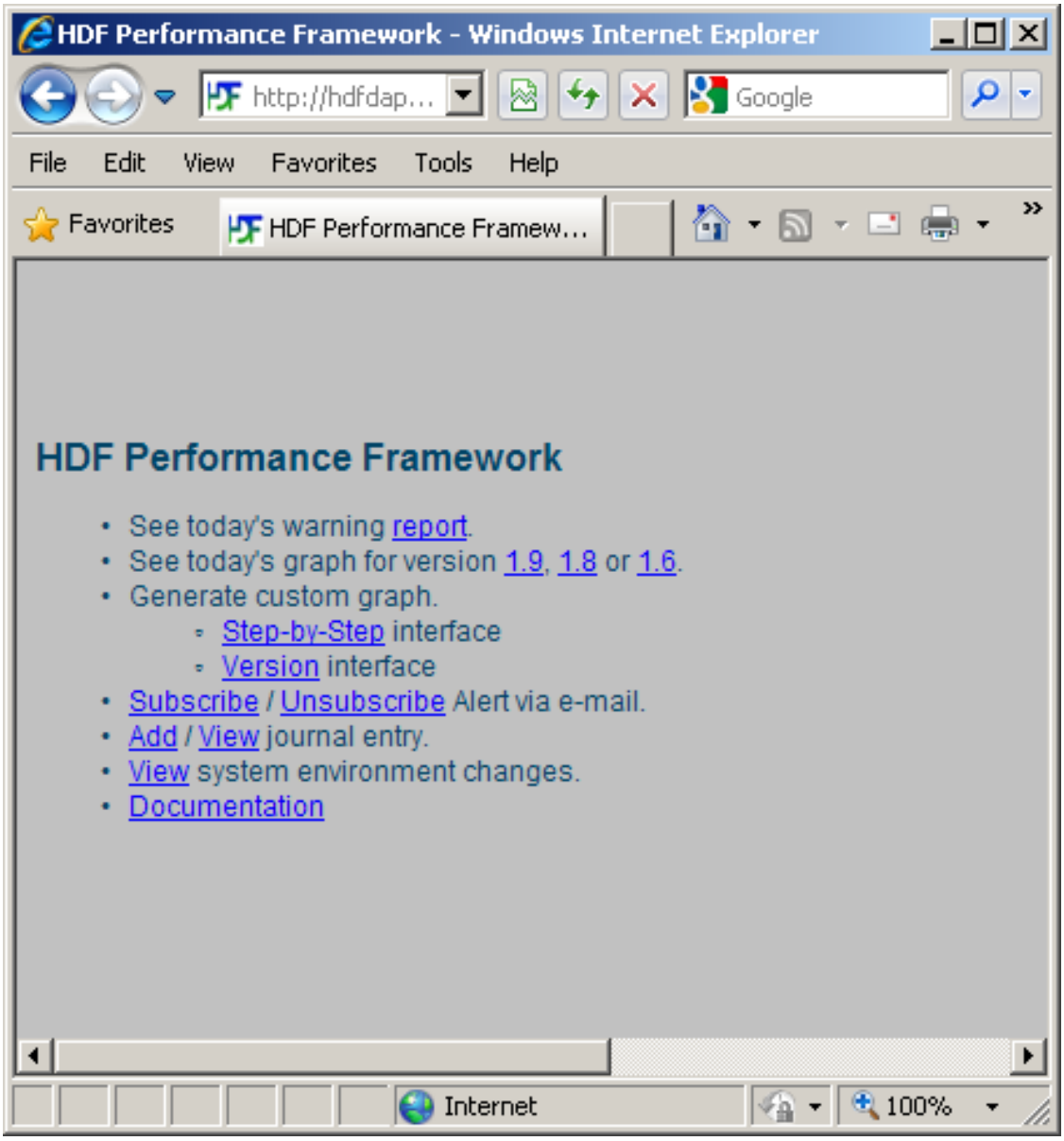
You should not install the sync.php and mailer.php cron jobs for the machines (e.g., linew, smirom and jam) that do not run the HPF MySQL database since they need to be run only once from the machine that collects all data using a MySQL database.

The sync.php cleans up any duplicate entries from the routine and action tables in the database and ensure that the front-end system can run smoothly.

The mailer.php script sends an e-mail notification to subscribers if there is a significant increase in performance.

3.6 Web Page Installation

Simply point your web server's document root to `hdf5perf/hdf5perfphp/hdf/`. Some web servers are configured to look for `index.php` first instead of `index.html`. The main web page of HPF is `hdf5perf/hdf5perfphp/hdf/index.html` so you should browse it through http://your_web_server/index.html. If it's installed successfully, the web page should look like the below:



Chapter 4. Writing Custom Benchmark Program

In this chapter, we provide a detailed instruction about creating a custom benchmark program. Writing a custom benchmark program requires some knowledge about HPF C-API.

4.1 Overview

To write a custom benchmark program, the following steps are needed:

Create a directory `mytest` under `hdf5perflib` source tree.

Copy files under `examples/*` into `mytest` using **cp -r examples/* mytest**.

Review `mytest/c/write.c` and make your own program by modifying benchmark names.

Make sure that your functions are wrapped around the `H5Perf_startTimer(&start)` and the `H5Perf_endTimer()` in your program.

If you've renamed `write.c` or created another file, please update `mytest/c/Makefile.am` file to include your source codes.

Modify `configure.ac` to include `mytest/Makefile` and `mytest/c/Makefile` and run **autoreconf**.

You may skip step 1, 2, 5, and 6 if you are in a real hurry and minimize the hassle of modifying several makefiles. In that case, you need to modify `examples/c/write.c` file directly. In the next two subsections, we'll focus on step 3 and 4 only since the rest of steps are more relevant to GNU autoconf/automake topic.

4.2 Creating Custom Routine, Action and Instance Name

HPF uses a hierarchical model to group benchmark programs. The model has three levels of abstraction - routine, action and instance. One routine can have one or more actions and each action can have one or more instances. Thus, to write a custom benchmark program, the first thing you'll need to do is to decide at least 3 names for each level.

For example, if you are interested in measuring "write" performance, you may name it as a "write" routine. Then, you may name two actions like "write groups" and "write datasets" under the "write" routine. Finally, you can add two instances like "write 1,000 groups" under the "write groups" action and "write 100 datasets" under the "write datasets" action.

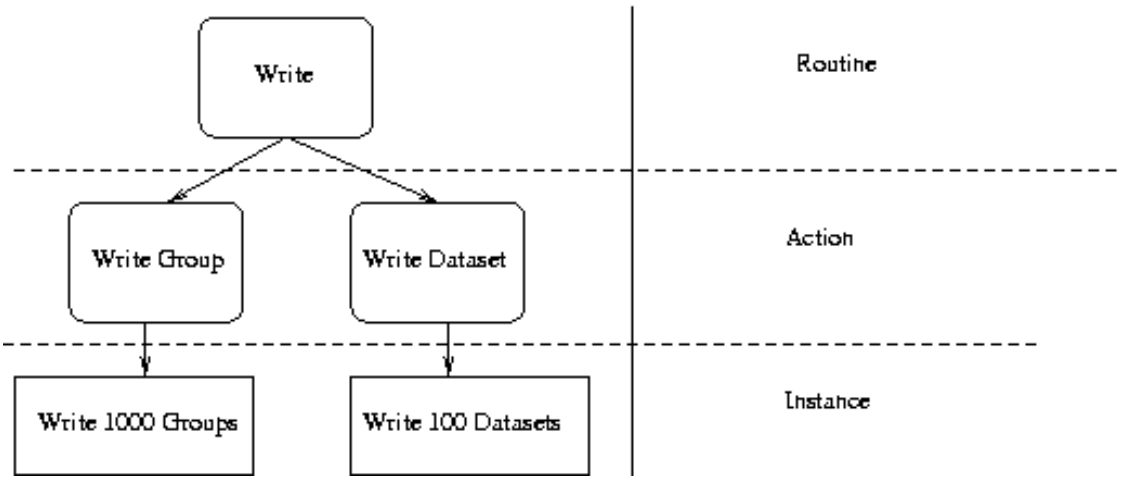


Figure 4-1. An example HPF hierarchical benchmark model

Once you've decided your names for routine, action and instance, it's time to modify the part of example code with your custom names. At the beginning of `examples/c/write.c` program, you can find the main() function. Below the main() function declaration, there are character array variables dedicated for the custom names as shown below:

```
int
main (int argc,char *argv[])
{
  /* BEGIN: Modify the following variables with your own names. */
  /* Names for Test Routine */
  char TestRoutine_Name[] = "Write";
  char TestRoutine_Desc[] = "A tutorial benchmark program";
  char TestRoutine_Version[] = "1.0";
  /* Names for Test Action */
  char TestAction_Name[] = "Write Dataset";
  char TestAction_Desc[] = "H5D Write function test";
  /* Names for Test Instance */
  char DatasetName[] = "Write 10000 Data";      /* This must be less than
50 characters. */
  char DatasetDesc[] = "Writing an array of 10000 integers"; /* This must
be less than 250 characters. */
  /* END  */
```

You are welcome to modify the content of the variables like `TestRoutine_Name(TestRoutine_Desc)`, `TestAction_Name(TestAction_Desc)` and `DatasetName(DatasetDesc)`. The `TestRoutine_Version` can be anything you want it to be since it does not play any significant role in HPF system.

4.3 Measuring Custom Benchmark Program Performance

Once you've modified variables for your routine, action and instance names, it's time to measure the performance of your benchmarking program. In this section, we will not cover the details of how to write programs with HDF5 library since there are plenty of tutorial examples on The HDF Group web site. Rather, we'll focus on how to inject your HDF5 program into our tutorial example.

Adding your own program into HPF is as simple as putting a letter into an envelop. All you have to do is to insert your own program between two API pairs called `H5Perf_startTimer()` and `H5Perf_endTimer()`. In the `write.c` example that we used in the previous subsection, look for the above APIs near line 110. It'll look like below program listing:

```
H5Perf_startTimer(& start);

/* BEGIN: Insert your custom benchmark program here */
status = H5Dwrite(dataset, H5T_NATIVE_INT, H5S_ALL, H5S_ALL,
                  H5P_DEFAULT, data);
/* END */

time_used = H5Perf_endTimer(start);
```

In the above example, you're measuring the "H5Dwrite" performance of writing "data" which is initialized at the beginning of the `write.c` file:

```
/*
 * Data and output buffer initialization.
 */
data = (int*)malloc(sizeof(int)*dx*dy);
tempdata = data;

for (j = 0; j < dy; j++) {
  for (i = 0; i < dx; i++){
    *tempdata = i + j;
    tempdata++;
  }
}
```

As you can see, the "data" initialization is not a part of performance measurement and is done at the beginning of the program. It is important to wrap only the target HDF5 API function(s) when measuring the performance. Don't wrap around other parts of source code since those code may take long time and it may affect the benchmark results accordingly. In our example programs, irrelevant parts like HDF5 file creation are always outside of the two wrapper HPF APIs -- `H5Perf_startTimer()` and `H5Perf_endTiemr()`.

Chapter 5. Testing HPF

5.1 Testing Back-end

HPF testing can be done by running a back-end script like `run_1.6.sh`, `run_1.8.sh` or `run_1.9.sh`. It's a good idea to set `INTERVAL="0"` in the script to get a quick result of your test.

If your script exits without any error, it's important to examine database tables. To examine your data, issue the following command on the MySQL server machine:

```
%mysql -u root -p hdf5perfsmiple
```

and then press enter. No password is required by default.

You can inspect newly recorded data from database by issuing an SQL query like:

```
mysql>SELECT * from TestInstance WHERE TestInstance_Date > 20091001;
```

In the above example, it's important to change the date `20091001` to today's date. After all your testing is done, you can clean up today(=`20091001`)'s data:

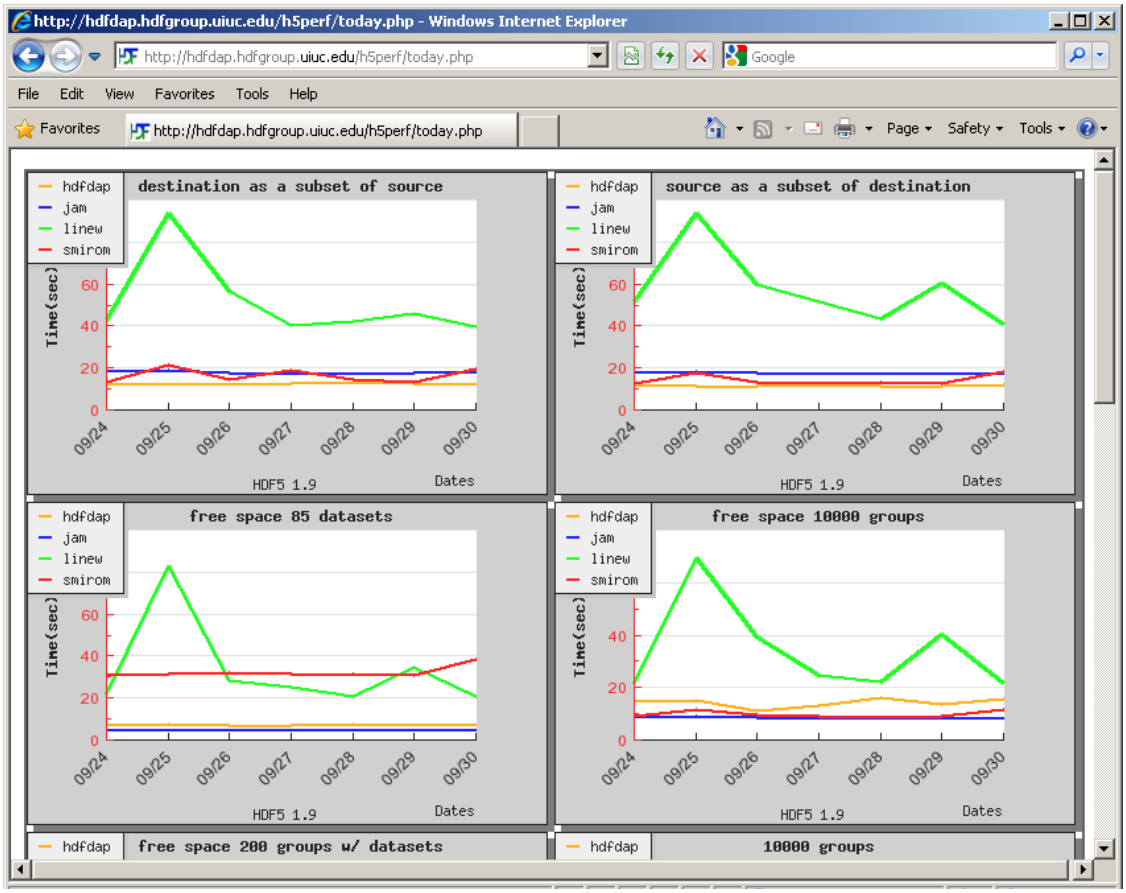
```
mysql>DELETE FROM subversion WHERE checkout_date > 20091001;
mysql>DELETE FROM TestInstance WHERE TestInstance_Date > 20091001;
mysql>DELETE FROM TestInstanceBest WHERE TestInstance_Date > 20091001;
mysql>DELETE FROM environment WHERE env_date > 20091001;
```

Please don't forget to set `INTERVAL="600"` back in your test script. Finally, submit your script as a cron job.

5.2 Testing Front-end

After two days of your cron job installation, it is possible to visualize your performance data. The reason is that any graph rendering by jpgraph requires two or more data points.

From the HPF main web page, click either "1.9", "1.8" or "1.6" in "See today's graph for version 1.9 1.8 or 1.6" list item. You'll see a line graph that looks like below:



The graph above shows the results of four of eleven example benchmarks used in The HDF Group. (The fifth one is cut due to the screen size.) It shows the last 7 days of benchmark results from the four different machines in The HDF Group - hdfdap, kagiso, linew and smirom. As you can guess from the last date "09/30" at the bottom of the legend, the screenshot was taken on October 1.

Also from the HPF main web page, click "report" in "See today's warning report." You'll see a lot of warning messages in a table since you have neither 7 nor 30 days of data to compare against in MySQL database. After 7 days of your installation, you'll see that half of the warnings disappear. After one month, you'll see almost no warning messages in the table.

Chapter 6. HPF Maintenance

HPF requires a periodic maintenance like manual data entry and version change.

6.1 Recording Missing Data

Sometimes, the current HPF fails to record the best instance from the three trials. In that case, one can inspect the `TestInstance` table and import the best record into `TestInstanceBest` table.This can be done using a utility script called `record_manually.php`.

```
%php record_manually.php <hostname> <version>
```

This will fill in the missing records in TestInstanceBest from TestInstance.

Check the graph to see if it's recorded properly:

```
%firefox http://hdfdap.hdfgroup.uiuc.edu/h5perf/today.php
```

6.2 Version Change

In case there is a version change in HDF5 library from 1.x to 1.y where x is the old version number and y is the new version number, perform the following steps:

- Update the back-end shell script.

```
# HDF5 version - 1.9 1.8 or 1.6
# VERSION="1.x"
VERSION="1.y"
```

- Rename the temporary directory for environment variables defined in the back-end script.

```
mv /tmp/chicago_1.x /tmp/chiago_1.y
```

- Update entries in tables if the version change was not planned in advance.

```
%php replace_subversion.php
```

- Update entries in the `TestInstanceBest` and `TestInstance` table using the `replace_version.php` script. Please modify `replace_version.php` and then execute.

```
%php replace_version.php
```

- Fill in the missing record for each host.

```
%php record_manually.php hdfdap 1.y
%php record_manually.php kagiso 1.y
%...
```

- Update `index.html`, `mailer.php`, and `today.php`. For each file, look for the lines that have version numbers.

6.3 Journal Entry

Finally, it's good to add a journal entry whenever you did some maintenance work. Although this journal feature is optional in HPF, it can be a very useful tool for documenting what affected the benchmark results. For example, HPF may fail due to a network interruption, a MySQL DB failure, and a full file system. If such extra ordinary event occurs, you can write a log easily with the journal form provided by HPF.

In HPF, there is a dedicated MySQL table called "journal" and you can easily add a new entry and view it later. To add a journal entry, click the "Add" link in the "Add / View journal entry" on the HPF main web page. An HTML form will appear and it'll ask a confirmation with preview page once you filled out the form. To view the entire content of the "journal" table, click the "View" link.