# Olio/PHP
# Install and Setup Guide from Tarball

## Overview

Olio is a macro-level toolkit consisting of the following components:

1. The web application
2. The main database
3. Distributed storage servers (NFS or MogileFS)
4. The caching server (memcached)
5. Storage metadata database (for MogileFS only)
6. Geocoder emulator
7. Workload driver

If your primary interest is in setting up the application alone, you need items 1-5 above and they can all be setup on a single system. If on the other hand, you would like to drive load against the application, you will need at least 2 systems. At higher loads, you may need multiple systems. At a minimum, we need to separate the SUT (System Under Test) components and the non-SUT components to get valid results. The non-SUT components are the Geocoder emulator and the workload driver. It is best to connect the driver machine to the SUT machine on a local private network. This ensures that latencies measured do not include arbitrary delays.

For a horizontally scaled workload, or to measure the performance of the individual components, you can deploy the SUT components on separate physical or virtual machines. Keep in mind though that the bulk of the CPU is consumed in the web application tier (apache/php).

In the following sections, we will go over the steps needed to configure each component :

Testing the Web Application

Starting a Performance Test

# Extracting the Contents

The Olio/PHP kit is packaged as a gzipped tar file. The first task is to unzip the file and untar it to extract it's contents. Because of the long pathnames, please use GNU tar to extract. This is the default tar on Linux but on Solaris, use '/usr/sfw/bin/gtar'. This can be done as follows:

```
# tar -xvzf olio-php-<version>.tar.gz (on Linux)
# gzcat olio-php-<version>.tar.gz | /usr/sfw/bin/gtar xvf - (on Solaris)
```

This will cause a directory named olio-php-<*version*> to be created whose contents should look like:

```
# ls olio-php-0.2*

LICENSE OlioDriver.jar RELEASE-NOTES-php-0.2.txt geocoder.war php_setup_kit.html

NOTICE oliophp/ release

#
```

We will use $OLIO_HOME to designate the name of this directory. A description of the contents follows :

- The *oliophp* directory contains the web application. This application will be deployed on the webserver.
- *OlioDriver.jar* contains the load generator/driver (which we typically refer to simply as *driver*). The driver is implemented using *Faban* – an open source benchmarking toolkit.
- *geocoder.war* contains the simple servlet used to emulate the geocoder.
- *php_setup_kit.html* is this document.
- *release* contains the release name. LICENSE and NOTICE contain the licenses.

# Setting up the Driver

Even if you don't plan to drive load against the application, steps 1 and 2 are required as the database and file loaders are part of the workload driver – feel free to install the driver on the same system as the web application.

1. See http://faban.sunsource.net/docs/guide/harness/install.html for Faban installation instructions. Note that faban needs to be installed on all the machines used for the test. Please also read the *Getting Started Guide* to get a high-level understanding of Faban terminology and how it works. From now on, we will refer to the faban install directory as $FABAN_HOME.
2. Install the required Faban services:
    - `cd $FABAN_HOME`
    - `cp samples/services/ApacheHttpdService/build/ApacheHttpdService.jar services`

- `cp samples/services/MysqlService/build/MysqlService.jar services`
- `cp samples/services/MemcachedService/build/MemcachedService.jar services`

3. Copy $OLIO_HOME/OlioDriver.jar to the $FABAN_HOME/benchmarks directory. Also copy the OlioDriver.jar file to the $FABAN_HOME/benchmarks directory on the web server system.

4. For the driver to work, you will need JDK 1.6. Set JAVA_HOME to the path of the JDK in the faban user's environment.

5. Start the faban master on the master driver machine :

   $FABAN_HOME/master/bin/startup.sh

6. Test that you can connect to the master by pointing your browser at http://<*driver_machine*>:9980.

3. Download MySQL Connector/J (JDBC Driver for MySQL) and install the jar in $FABAN_HOME/benchmarks/OlioDriver/lib.

## Installing the Web Application

The web application is a PHP application. It requires the following components:

1. A web server such as Apache2 or Lighttpd
2. PHP 5 with extensions: apc.so, curl.so, gd.so, pdo_mysql.so
3. MySQL 5
4. Memcached

All of these applications are included in OpenSolaris. If you're running on any other operating system, please install the above applications.

Once you have the application stack installed, follow the steps below to set up the application.

1. Decide where you want to install the web application. The default for apache is the htdocs sub-directory but a common location is /var/www. Create a sub-directory *oliophp* where the php application will reside and copy the contents of $OLIO_HOME/oliophp to this directory. We will use $APP_DIR to refer to this location. Ensure that all files in $APP_DIR are readable by the world.

2. Edit the httpd.conf used by your system's apache installation. Set the **Listen** parameter to the hostname or ip address and set the **DocumentRoot** to $APP_DIR/public_html. See the *httpd.conf* file in $APP_DIR/etc for additional settings.

3. See the php.ini provided in $APP_DIR/etc and copy the settings appropriately to the php.ini for your installation.

## Setting up the Database

1. If you plan to run MySQL on a separate machine, install MySQL on that system. We will refer to the

MySQL installation directory as MYSQL_HOME.

2. Setup the mysql user/group and permissions for it's directories:
```
# groupadd mysql
# useradd -d $MYSQL_HOME -g mysql -s /usr/bin/bash mysql
# chown -R mysql:mysql $MYSQL_HOME
```

3. Create the database:
```
# su - mysql
$ cd bin
$ ./mysql_install_db
```

4. A sample MySQL configuration file (my.cnf) is included in the $APP_DIR/etc. Please read the embedded comments and edit the file appropriately after copying it to /etc.

5. Start the mysql server using the my.cnf file. Substitute your own password for *pwd* (we typically use *adminadmin*)
```
$ ./mysqld_safe –defaults-file=/etc/my.cnf &
$ ./mysqladmin -u root password pwd
```

6. Create the olio user and grant privileges:
```
$ ./mysql -uroot -ppwd
mysql> create user 'olio'@'%' identified by 'olio';
mysql> grant all privileges on *.* to 'olio'@'%' identified by 'olio' with
grant option;
```

   In some cases the wildcard '%' does not work reliably as a substitution for all hosts. You need to grant the privileges to 'olio'@'<hostname>' individually, where hostname is localhost, the name of the database system and any other name that's used to access the database system.

7. Create database
```
mysql> create database olio;
mysql> use olio;
```

8. To create the database schema, you first need to copy the schema script from the driver system. It is located in $FABAN_HOME/benchmarks/OlioDriver/bin/schema.sql. Copy the script anywhere on the database server and create the schema:

```
$ mysql -uroot -ppwd
mysql> \. <location>/schema.sql
mysql> exit
$
```

   Now, if you login as the user *olio*, you should be able to see the database created by the root user.

## Loading the Database

It is best to load the database manually the first time so that we can test the web application. However, while doing performance tests, the load driver can be configured to automatically re-load the database before the run.

1. Login to the machine running the Faban master driver. Only this machine has the loader at this time.

2. Go to the directory containing the loader script:

```
# cd $FABAN_HOME/benchmarks/OlioDriver/bin
```

3. Ensure the script has execute permissions. Faban takes care of this for the runs, but since we have not yet started the first run, we will need to change that ourselves:
   ```
   # chmod +x dbloader.sh
   ```

4. Run the loader script:
   ```
   # ./dbloader.sh <dbserver> <load_scale>
   ```
   You can start small with a SCALE of 50 for initial testing.

5. Edit the $APP_DIR/etc/config.php and set the database host name, replacing *localhost* in the dbTarget entry.

   ```
   $olioconfig['dbTarget'] = 'mysql:host=localhost;dbname=olio'; //
   PDO target.
   ```

## Setting up the Filestore

Olio can be configured to use either a local filesystem or MogileFS for the object data. Our initial testing with MogileFS found some severe performance issues, so for now we advice using a local filesystem or network file system such as NFS. You will need about 50GB of space for the data, as the data does grow over runs. Using a single spindle does work but may create performance bottlenecks. We recommend striping the filesystem across at least 3 spindles to avoid such bottlenecks. A local file system needs to be setup on the same machine as the web application. A network file system can reside on a separate server but needs to be exported and mounted on the system running the web application.

1. Create a directory (or mount a filesystem) designated for storing the image and binary files. This directory is referred to as $FILESTORE. Any valid name for the OS should be fine. Ensure that everyone has read and write access to it:
   ```
   # mkdir -p $FILESTORE
   # chmod a+rwx $FILESTORE
   ```

2. Now load the filestore. You should have the $FABAN_HOME/benchmarks/OlioDriver.jar file on this system.

   ```
   # JAVA_HOME=<java_install_dir>; export $JAVA_HOME

    # cd $FABAN_HOME/benchmarks; mkdir olio

    # cd olio; jar xvf ../OlioDriver.jar; chmod a+x bin/*
    # $FABAN_HOME/benchmarks/olio/bin/fileloader.sh <load_scale> $FILESTORE
   ```

   This loads files for use for up to *load_scale* number of active users.

2. Edit the $APP_DIR/etc/config.php parameter *localfsRoot* and set it to $FILESTORE.

## Setting up the Geocoder Emulator

The Geocoder Emulator is a simple servlet deployed on Tomcat. It is typically run on a driver machine. The following steps describe how to install it :

1. Donwload and install Tomcat (from http://tomcat.apache.org) on the driver machine. The install directory doesn't matter – we will refer to it as $TOMCAT_HOME.

2. Copy $OLIO_HOME/geocoder.war file to $TOMCAT_HOME/webapps.

3. Start Tomcat using $TOMCAT_HOME/bin/startup.sh.

4. Edit $APP_DIR/etc/config.php parameter geocoderURL and replace *GEOCODER_HOST:8080* with the host and port where Tomcat is running.

## Setting up Memcached

The Olio application uses memcached for caching the home page which dramatically reduces the load on the database. If you are running Olio to stress MySQL, you may not want to have the caching tier. In this case, set the *CacheSystem* to *NoCache* in $APP_DIR/etc/config.php. This will eliminate the use of memcached and all requests will directly go to the database. However, if you do want to use memcached (scaling will be very difficult without it), you need to setup an instance of memcached. This can typically run on the same system as the web server.

1. Download and install memcached on the system you plan to use.

2. Start memcached : "memcached -u mysql &"

3. Edit $APP_DIR/etc/config.php and replace MEMCACHED_HOST with the host name where memcached is running.

## Testing the Web Application

1. Start apache. This can be done by executing "$APACHE_HOME/bin/apachectl start". Check that you can connect to it from your browser at the port apache is listening on (http://*host*:80).

2. Check the home page (HomePage) . If there are no error messages and all images get loaded, that's a great start ! If instead, the server does not display anything or the images don't get loaded, check the apache error_log and double-check that the application's etc/config.php file has been edited correctly. Remember that you will need to re-start apache after any edits to this file for them to take effect.

3. Click on an event (EventDetail). Make sure the whole page looks OK.

4. Click on an attendee (PersonDetail) to see a person's profile.

5. Go back to the home page and click on a tag in the tag cloud. Choose a big tag and check that we have good results and images get loaded OK.

6. Click on the sign up tab. Fill in the form and create a user. Make sure you find some jpeg images to upload. If not, take them from $FABAN_HOME/benchmarks/olio/resources.
   Submit the form. Make sure the form goes through. This completes the AddPerson transaction.

7. Login using your new login name you just created. The top right of the screen should show that you're logged on.

8. Select an event, go back to the EventDetail page but this time as a logged on user.
   Add yourself as an attendee. This is the EventDetail transaction with attendee added (about 8-9% of all EventDetail views).

9. Click on the add event tab and add an event. Make sure to include an image and some literature. You can also use the files from $FABAN_HOME/benchmarks/olio/resources. Fill in the form and submit. This is the AddEvent transaction.

## Starting a Performance Test

Now that we know that the web application is running and the faban harness is up, it is time to kick off a test.

1. Shutdown apache and kill memcached. Memcached is always started by the driver before the run to ensure a clean cache and will cause port conflicts if it is already running. The apache server is also automatically started at the start of a run and shutdown at the end of a run.

2. Point your browser at http://<*driver_machine*>:9980

3. Click on the **Schedule Run** link.

4. Under the JAVA tab, set the JAVA_HOME. You can accept the default value for JVM options. **DO NOT** click on the OK button yet!

5. Select the Driver tab.

   Enter a Description for the run (say 'First test run' for this case). In general, the Description field is very useful to get a quick idea of what a particular run is testing.

   Enter the name of your driver(s) machine for Host (when using more than one machine, simply separate them by a space).

   Enter 10 for 'Concurrent Users' (we want to start small).

   Enter 'vmstat 10' for Tools. This indicates the measurement tools that will be run on the driver machine. It's a good idea to keep an eye on the driver cpu utilization.

   Now enter 30, 30, 30 for the Ramp up, Steady State and Ramp down times. This is a very short test run. For normal runs, you may need a ramp up of 200 seconds and a steady state of at least 600 seconds during which measurements are made.

   For current systems, the time between client startup of 200 milliseconds is good enough. Some web servers or slower systems may not be able to accept connections very frequently. In that case we may want to increase this value to 1000 milliseconds.

The number of Agents needs to be set based on the number of concurrent users - we start with 1 and add an agent for every 500 users.

6. Select the Web Server tab.

The Host:Port Pairs field takes the host port pairs where the web applications are running. The host and port is separated by a colon.

Each pair is separated by space.

For the Webserver type field, enter either "apache" or "lighttpd" depending on which web server

you're using.

Leave the field blank if you're using servers other than these two. Only these two servers are managed by faban.

For any other servers, you must manually start/stop the servers.

Enter the webserver's bin, log, and config directories, and the directory containing the php.ini file in the respective fields.

In the tools box, type the tools you want to run. Here are the tools we typically run : vmstat 10; mpstat 10; nicstat 10; iostat -x 10

7. Select the Data Servers tab.

   For the database server, enter the Host name. (Olio also supports multiple hostnames if MySQL replication is being used).

   Edit the hostname part of the JDBC Connection URL. This is used by the loader program to reload the database before a run.

   Set the 'Load for Concurrent Users' to 25 (this is the minimum number of users we can load for and is good for up to 25 concurrent users).

   Set the Data Storage server. For local storage this is the same host as the web server.

   Set the memcached server instances to the servers you've configured in config.php of the web application.

   The memcached server instances are given as host:port pairs, separated by space. If a port is not given, the default port of 11211 is assumed.

8. That's it. Click OK and the run should be scheduled. You can click on the View Results link on the left to monitor the run.