JmeterMeter

Chinese Version

I(loadtester) have translated this document to Simple Chinese, and posted it to Bonoy QA.

Summary

I built this set of programs since I run jmeter almost always in non-interactive mode and the measures computed by jmeter itself were insufficient for my tests.

- Completely dependent on jmeter.log
- Has several metrics to analyze the performance of jmeter itself and "client-end" measures:
    1. number of active/terminated threads
    2. requests per minute
    3. average response time
    4. average response time across last X% of requests
    5. average response time across last X% of test duration
    6. mean time between consecutive requests (a nice way to know when things are slowing down)
    7. number of waiting requests (request sent but not received response yet)
    8. grouping response times based on URL regex matching (this would require a text file called RequestClasses in the dir where the server program in invoked with each line containing a PERL5 regular expression to match specific URL's)
- XML of results can be seen at anytime using a simple command: `telnet <host_running_jmserver> <port>`
- Results aggregation across hosts running jmeter
- Dynamic/near-realtime charting of results using a gnuplot script created dynamically

Dependencies

- XML::Simple
- Log::Log4perl
- Data::Dumper
- DateTime
- DateTime::Format::Strptime
- List::Util
- Gnuplot

Contents

jmeter.pm

Perl package containing the core class definition and algorithms for metrics computation

Gnuplot.pm

A perl wrapper for Gnuplot to automatically generate a gnuplot script from a perl hash.

jmserver.pl

A server program that invokes two threads: one that reads jmeter.log and computes all kinds of metrics; another that is listening to a socket and when a client attempts to read from the socket, spews out an XML containing the results.

jmclient.pl

A client program that reads XML containing measures from one or more servers, aggregates them, dynamically generates a gnuplot script, and runs gnuplot to generate charts.

Architecture and limitations

The core idea behind this is that the server running jmeter should do nothing except just that – generate load. So aggregation of the various performance measures is done by a single instance of `jmclient.pl` while there can be one instance of `jmserver.pl` per jmeter instance, gathering data from the jmeter log.

The code would only work on a linux distro once the dependencies have been installed. But anyone who knows a little perl can port it to other OS's. (A gnuplot port should ofcourse be available on that OS.)

Also, currently, the analysis is restricted to HTTP samples, but again, even that can be easily changed by changing a few lines in the server process.

It is necessary to have the following two settings in jmeter.properties.

```
log_level.jmeter.protocol.http.sampler.HTTPSampler=DEBUG
log_level.jmeter.threads.JMeterThread=INFO
```

The root loglevel should be WARN to save IO overhead within jmeter.

Ofcourse, if you are trying to gather metrics on other samplers (JDBC,..) you should set the corresponding class at DEBUG level.

Communication between the one or more `jmserver` processes and the single
`jmclient` process happens via an XML string streamed through a network socket on each host.

`jmclient` aggregates results across servers and can optionally be made to invoke several child processes each of which can gather measures from other sources. For eg., I have a database statistics collector that collects the number of active/idle oracle sessions and the memory used by them to monitor database performance; another program that is invoked as a child that can collect system memory and CPU usage of arbitrary hosts running perhaps the application/database server; and so on.

`jmclient` then creates a hash of the results and passes the hash to an instance of `Gnuplot.pm` that as evident, is a wrapper around the unix charting tool, gnuplot. It dynamically creates a gnuplot script the first time it is invoked with the results hash and from then on, keeps including the new results to dynamically generate charts.

Finally, you can create a simple html that shows the charts; I create a table in the html with each column/row corresponding to each test so that test results can be compared.

Usage

Make sure you install all dependencies.

There are some additional monitoring programs that I invoke as part of `jmclient.pl`. They should be commented out. Read the source code before using it.

On the host that is going to run jmeter:

1. `cd <dir in which jmeter.log will be written>`
2. `rm jmeter.log; touch jmeter.log`
3. `tail -100f jmeter.log | perl /path/to/jmserver.pl`

On the client that is going to aggregate and visualize charts (i strongly recommend this is a different machine)
1.#4 `cd </path/to/results/directory>`

1. `mkdir -p <test_name>/figs`
2. `cd <test_name>`
3. `perl /path/to/jmclient.pl`
4. Create a html pointing to the png files (I create 1 column/row for each test so that I can compare results across tests.)

There will be some files generated in the dir:

- `stats.dat` contains a space delimited text file containing all data which will be used by gnuplot
- `figs/` contains all .png's containing metrics
- `.jmclient.run` which contains the time at which the test started. this is useful if you kill the test and restart, but want the original stats to be included while charting the new measures
- `gnuplot` is a dynamically generated gnuplot script. this script is generated once per execution of jmclient.pl. For every new stats XML sourced from the servers running jmserver.pl, this is run once.

Once again, make sure you read the source in all files before using. The program will not compile until you install all the dependencies and may have runtime failures too; but they should be easy to fix if you know a little perl or any other language.

I am yet to optimize `jmserver` – perl's hashes are not easy on memory, and `jmserver` takes a tad more CPU than I would've wished. But I will update the source here whenever I have a newer version of the libraries. You can fire comments/questions to the jmeter user mailing list.

Source: JmeterMeter.tgz