Jmeter report in pdf

So the test is finally done and it's time for analysis and optimization, right? No? What should I do after the JMeter test to evaluate its results? You know as a result of the JMeter load test several XML or CSV files... And that's it. In practice, this often means that the tester looks at the numbers and evaluates important things. This also usually means that some other important things are missing, and the test is essentially worthless. This is not real reporting or analysis. You can try to visualize the results by filling the visitors with output files and saving them as a graph or table (but only if the listener saves the log on the disc, otherwise you won't see anything). But even if everything works as it should, you see only individual parts, not a holistic view of the results. This is a great job for a rather paltry result. It should be a lot easier. Want a one click test report? During the test, you can see the live status of everything you have to look for in a bunch of numbers in the JMeter results. Once the test is complete, the load test report is generated at the touch of the Generate report button. Check out this example of the report. You can easily compare multiple graphs using the crossing tool in the bottom right corner. Identify the exact moment on the graph and compare it to other graphs. This allows you to detect unexpected connections. The adoption criteria allow auto-evaluation of test results. A quick look at if they were executed or not. Red means failure. Take advantage of continuous integration tools to evaluate tests on your behalf! Error Review When an error occurs, you should see the full detail of it - requested URL, query and body headlines, response code, headliners and body, and message approval. All included in the SmartMeter.io report. Do you do the tests several times or as part of the CI pipeline? If so, you will rate the trend analysis page. No configuration required. trends are formed automatically based on the name of the test script file. Receiving a report from JMeter downloads a log from the JMeter (.jtl) file in XML to the report generator and receives the report with one click. SmartMeter.io require more than 1.5.0 or newer. An open SmartMeter.io editor (Create/Editing a script on the welcome screen). Click the right button on WorkBench (the last element in the component tree on the left). Select Add/Non-Test Elements /et@sm - Report Generator. Click View and select the JMeter file. Additionally, click View and select a test script file (to evaluate the criteria for accepting and receiving Trend analysis). Click Generate Report Alternatively, run the JMeter test script SmartMeter.io year and get a test report in the usual way or easy to create Check out the quick guide to get some starting tips. There's the most. log, and the test itself is backed up while the report is created. This means that you will never lose old results and can compare them with new ones. You can also customize the report because its template is a simple HTML file that makes it easy to add a corporate logo or use the right colors to make it stand out in the presentation. If you're missing something, or something doesn't matter, you can re-create it and install exactly what you want to see, but more about it another time. Now, get to testing! All properties must be prefixed by jmeter.reportgenerator. Description of the attribute report_title the name used in the generated report. Default: Apache JMeter Dashboard date_format not start_date the start date of the data range for use in the report. The date format is determined by date_format property. Default: Not filled, which means that the data range will be used from the beginning of No. end_date the end date of the data range for use for the report. The date format is determined by date_format property. Default: Not filled, which means that the data range will be used until the end of No. overall_granularity No. apdex_satisfied_threshold sets a satisfaction threshold for calculating APDEX (in ms). Default: 500 Apdex_tolerated_threshold sets a clearance threshold for APDEX (ms). Default: 1500 No jmeter.reportgenerator.apdex_per_transaction sets a threshold for satisfaction and tolerance to specific samples. Use sample names or regular expressions. The sample_name format: satisfaction and tolerance; Values are in milliseconds. Notice the colon between the sample name and the values, pipe between the rapids and the sharpened at the end to separate the different samples. Don't forget to run after the colonial to cover a few lines. Example: jmeter.reportgenerator.apdex_per_transaction sample (d):1000-2000; Samples12:3000-4000; scenar01-12:5000-6000 No sample_filter installs a sample filter to generate graphs and statistics. Empty value deactivates filtering. Format: Regular expression. Default: temp_dir installs a temporary directory used by the generation process if it needs I/O file operations. Default: the temperature statistic_window sets the size of the sliding window used to estimate the percentille. Warning: Higher value provides better accuracy, but requires more memory. Default: 20,000 No Percentile Used Summary Tables and Interest charts can be adjusted to different values using 3 properties: aggregate_rpt_pct1 : Default to 90 aggregate_rpt_pct2 : Default to 95 aggregate_rpt_pct3 : By default up to 99 Relative paths built from JMeter work catalog (default: default You can identify some common properties that are used by the generator configuration. These properties are loosely called, but you should use the set-top box jmeter.reportgenerator. to avoid overlapping properties. For example: Definition of property: property: Link to real estate: reports jmeter.reportgenerator.overall_granularity HTML are very intuitive. In most cases, the development team and company management need this report to evaluate the performance of a particular site or application. The main steps to create a JMeter (.jmx) file are: Step 1: Creating a thread group. Step 2: Sending a request (HTTP Request). Step 3: View the results with the help of listeners (results in a tree or table). Step 4: Run the jmx file into Non GUI mode and generate HTML reports. There are two ways to create HTML reports: Approach 1: to report creae at the end of the test———————————————————— jmeter-n-t (file path .jmx) -l (the way of the sample folder along with the name of the csv file, where you want to save results) -e-o (the path of the output folder, Where you want to save results)Approach 2: Create a report from the standalone csv file———————————————————— —jmeter-g (file resultant csv) -o (the way of the output folder where you want to save results)Non GUI command with an explanation: Jmeter -n : This team says the tip of the team I want to run my script Jmeter in non GUI mode.#Jmeter -t : This will choose the Jmeter file .jmx Here we give the way of the jmx file, which is under the default bin folder or may be in another folder. We give the way to the bin/example folder along with the same csv file name. It's best to create a folder called csv in the bin/example folder to separate your csv and HTML files. It's best to create a folder called html in the bin/example folder to separate csv and HTML files. Approach 1: To run the command first you have to go to the drive in which you held the Jmeter. Let's say I kept my Jmeter in E drive.Command: $e: Now go to your Jmeter.Command: $CD E: 'apache-jmeter-4.0'apache-jmeter-4.0 ' binNow perform full command.Command: jmeter-n-t E:' Jmeter_jmx_Files'rides-proxy-twin-Foods.postman_collection.j mmx-l E:'apache-jmeter-4.0'apache-j-meter-4.0'bin'examples'csv-food.csv-e-e:'apache-jmeter-4.0' bin'example-of-food-HTML-generation report from CMDNow and you'll find the Food.csv file then go to HTML'gt;Food (exit folder) and you'll get there index.html file, which is your essential HTML report. To share the report, copy the .csv report to this folder and zip it all together and share it. If you only share an index.html file, it won't work, you should share it all ReportApproach 2: Suppose you've already created your test, as well as your report in the csv file, now want to create HTML reports from csv file.Command: jmeter-g E:'apache-jmeter-4.0'apache-jmeter-4.0'bin examples:'apache-jmeter-jmeter-4.0'examples'bin'-example-foodStep 5: Open index.html file in your browser, and you'll find a dashboard and dashboard. From the dashboard you will get APDEX, Summery Result, Stats, Errors and Top 5 Sampler Errors. The HTML ReportFrom dashboard charts you get different types of charts according to the passage of time, bandwidth and response times.Charts HTML Report I'm sure you agree with that: There are so many ways to collect and interpret JMeter results, you feel lost. Well, it turns out, after reading this post, you'll know 12 different ways to collect and analyze the results! We're going to explore all possible ways to get deep metrics, including graphs, charts, tables, HTML reports and more. JMeter is such a sophisticated tool with so many amazing features that it becomes difficult to know what to do. This is the ultimate guide on how to analyze the results JMeter will jump to start their knowledge of JMeter. Preliminary Installation Requirements This tutorial assumes that you already have the following software installed: Java 8, JMeter 3.3 or above. The following example of JMX is used throughout the guide. This JMX is testing our Java-based demo JPetstore complete with Docker. Understanding JMeter Metrics JMeter Metrics is widely used in the following section, so it's best if you're comfortable with their definition: Past time: Measures the past time before submitting a request immediately after receiving the last response snippet, Delay: Measures the delay immediately before submitting the request immediately after receiving the first piece of response, Connect Time: Measures of the time it takes to establish a connection, including the SSL handshake, median: The number that divides the samples into two equal halves, 90% of the line (90th Percentile): past time below which 90% of samples fall, Standard deviation: Measuring data set variability. This is a standard statistical measure, the name of the stream: Derived from the name of the thread group and the thread within the group. The name has the format of groupAindex - - - - - threadIndex, where: groupName: the name of the element Thread Group, groupIndex: the number of the group of threads in the test plan, starting with 1, threadIndex: the number of threads in the group of threads, starting with 1. Bandwidth: Calculated as a query/time unit. The time is calculated from the beginning of the first sample to the end of the last sample. Formula: Bandwidth (number of requests) / (total time). Interpreting JMeter metrics How do you know if the metric is satisfying or awful? Here's Past time / Connection time / Delay: should be as low as possible, ideally less than 1 second. Amazon found every 100ms cost them 1% in sales, which means a few million dollars lost, Median: should be close to the average past response time, XX% line: should be as low as possible too. When this is much lower than the past time average, it indicates that the latest XX% queries have a much higher response time than the lower, standard deviation: should be low. A high deviation indicates discrepancies in response times, leading to irregular response times. You see, it's pretty simple! Most numbers should be as low as possible. However, depending on the context, your boss can provide you with the expected response time for this load. Use them to calculate apdex of each query: Apdex (Application Performance Index) is an open standard developed by an alliance of companies. It defines a standard method of reporting and comparing the performance of software applications in computing. To run JMeter in headless (non-GUI) mode, which means without a user interface, to run load tests use the following command: jmeter-n-t scenario.jmx-l jmeter.jtl Command line has the following parameters: -n: start in mode, No vviy, -t: determines the path to the source .jmx script to run, -l: points the way to the JTL file, which will contain See our blog How to optimize JMeter for large-scale tests to understand why running in mode, not in mode, not gui is vital. Launch a demo application To run a demo application on your own computer, you need: To run the JPetstore demo app, just run a command-linear docker run-d -p 8080:8080 jloisel/jpetstore6. Open the browser and go to the . He has to show the first page of JPetstore. The next test group configuration will be launched: 20 simultaneous thread groups, 120 seconds of build-up duration, 120 seconds of peak test duration. The test will run for a total of 4 minutes with 20 simultaneous peak load users. UI Listeners JMeter has a number of UI Listeners, which can be used to view the results directly in JMeter UI: View results like a tree: The view results tree shows a tree of all sample responses, allowing you to view the answer for any sample., Results of the graph: The listener's graph generates a simple graph that charts all the time of sampling, Aggregate report: The cumulative report creates a table line for each differently named request in the table As well as the tree of presentation results, this visualizer uses a lot of memory, this visualizer uses a lot of memory, the aggregate chart: similar to a cumulative report. The main difference of the consolidated graph provides an easy way to generate bar charts and save the graph as a PNG file, generate brief results: This test item can be placed placed in test terms. Generates a summary of the test run so far in the log file and/or standard output. Both running and differential results are displayed. Some listeners were omitted: these listeners only for debugging the purpose. These listeners help diagnose script problems, but are not designed to provide performance indicators, as are the following: As a general rule, avoid using the Listeners user interface. They consume a lot of memory. They are not suitable for real load tests. Some of these may even cause an Out Of Memory error with multiple parallel thread groups. Placing listeners depending on where the listener posts the results, it collects different metrics. The JMeter results listener collects results from all the elements at or below. For this reason, it's a good idea to place listeners on a test plan level to collect all the results of the thread groups. Presenting the results tree The View Results Tree is essentially a tool for debugging submitted requests and responses received. It's useful to see if the script works correctly. But, it's not very suitable for viewing results when many simultaneous users are working. It quickly you run out of memory because it keeps all the results in the main memory. Some metrics are available when clicking on each query as follows: Flow title: JPetstore 1-1 Example Start: 2017-10-06 10:42:09 CEST Download time: 30 Connection time: 30 0 Delay: 29 Bytes size: 1530 Sent bye :582 Heders size in bytes: 196 Body size in bytes: 1334 Graph example: 1 Error Count: 0 Data Type (text bin): text response code: 200 Reply message: OK I would suggest using this listener: Fix the script before scaling the test for more simultaneous users, Identify the baseline performance by running one thread group for one iteration, and/or using the responses received to fix/design the mail processor to extract dynamic settings. The aggregate Graph graph is the Listener user interface, which brings some useful metrics to the width of testing about each request and transaction controller. It also includes a bar chart that can be modified to please your needs with different settings. I have to say, there are too many tweaks, and worse, none of these settings are saved in JMX. You lose them when you close JMeter. Although, I have to admit that it's really nice to be able to export a schedule of both PNG and export tables as CSV for future use in a specially designed report. The metrics are test, which means that you get, for example, the average response time for the entire Metrics available: Tag: query name, samples: total execution, average: Average past time in milliseconds, Median: Median is a value separating a higher half of the sample, population, or probability distribution from the bottom half. For a dataset, this can be seen as Value, 90% Line: 90% Percentile, Percentile (or Centril) is a measure, used in statistics indicating below which this percentage of observations in the observation group are falling, 95% Line: 95% Percentile, 99% Line: 99% Percentile, Min: Minimum Past, Max: Maximum Past Time, Errors %: Error Percentage (Errors / (Errors and Samples) 100), Bandwidth: Number of Samples per second, and KB/sec: Capacity Like any other UI Listener, I wouldn't recommend using it for real load tests. The cumulative Report Cumulative Report is very similar to an aggregated graph containing only a metric table. This listener can be used to run user-free load tests (without running the user interface) because the stats can be stored in the CSV file for longer use. It contains exactly the same metrics as an aggregated graph. These metrics can be used to write a report using Word, for example. JMeter summarizes listener results during load testing in the JMeter console, as shown below. It displays only a few common metrics every few seconds: Create a Summary of Results No. 5 in 00:00:07 and 0.8 /s Avg: 159 Min: 29 Max: 238 Err: 1 (20.00%) Active: 1 Started: 1 Finished: 0 Creation Summary Results No 7 at 00:22 and 0.3/s Avg: 163 Min: 54 Max: 239 Err: 0 (0.00%) Active: 0 Started: 1 Finished: 1 Creation Summary Results No. 12 at 00:28 th 0.4/s Avg: 161 Min: 29 Max: 239 Err: 1 (8.33%) Create Summary Results No. 17 at 00:00:25 th 0.7 / with Avg: 185 Min: 28 Max: 524 Err: 3 (17.65%) Active: 3 Started: 3 Finished: 0 Creation Summary Results No 32 at 00:30 and 1.1/s Avg: 160 Min: 28 Max: 239 Err: 2 (6.25%) Active: 2 Started: 5 Finished: 3 Creation Summary Results No 49 at 00:00:55 and 0.9/s Avg: 169 Min: 28 Max: 524 Err: 5 (10.20%) Create Summary Results No. 29 at 00:00:30 and 1.0/s Avg: 164 Min: 28 Max: 246 Err: 3 (10.34%) Active: 3 Started: 8 Finished: 5 Creation Summary Results No 78 at 00:01:25 and 0.9/s Avg: 167 Min: 28 Max: 524 Err: 8 (10.26%) Create Summary Results No 31 at 00:00:30 and 1.0/s Avg: 165 Min: 28 Max: 242 Err: 2 (6.45%) Active: 2 Started: 10 Finished: 8 Creation Summary Results No. 109 at 00:01:55 0.9/s Avg: 166 Min: 28 Max: 524 Err: 10 (9.17%) Create Summary Results No. 4 at 00:00:05 0.8/s Avg: 168 Min: 138 Max: 181 Err: 0 (0.00%) Active: 0 Start: 10 Finished: 10 Creation Summary Results No. 113 at 00:02:00 0.9 /s Avg: 166 Min: 28 Max: 524 Err: 10 (8.85%) These log lines are already the default output when JMeter is launched in unprepared mode. JMeter Jenkins Plugin

is able to analyze these lines and output graphs when launching JMeter on Jenkins. Graph Results JMeter Graphs Results Displays Linear Charts for Common Metrics as well as Figures No samples: number of samples processed, Last example: Last time in milliseconds, Average past time: in milliseconds, Standard deviation: deviation: milliseconds, and bandwidth: in KB/sec. This leads the listener not worth it. The graphs are barely readable. And as JMeter explains: Graphic results should not be used during load testing because it consumes a lot of resources (memory and processor). Use it only for functional testing or during debugging and testing plan testing. To sum up, most UI listeners are perfectly connected to debugging/testing the target. Don't expect to hit high loads (500 simultaneous users) to use them sparingly. These listeners were designed to quickly get metrics while running load tests inside the JMeter user interface, for very light loads. You can even use them for average load (100 to 500 simultaneous users), but don't expect to run distributed JMeter tests with JMeter user interface. That's not the goal. Remember, JMeter is configured with 512MB heaps of memory by default, which is pretty low. While you can increase JMeter's dedicated memory, it feels like handling water from a boat that doesn't float anymore. Now that we've tested most of the user interface listeners available in JMeter, the question is obvious: What listeners can we use when running real load tests? Ready-to-listen JMeter (or not user interfaces) are specifically designed to work when JMeter starts from a command line. These listeners are the ones that are used when running real load tests because they use much less memory than UI Listeners. As? These listeners don't keep the results in mind, they are basically responsible for offloading the results to another environment. Existing non-GUI JMeter Listeners: Simple Data Writer: Listeners can be configured to save various items in the results log files (JTL), Backend Listener: Backend Listener asynchronous listener, which allows you to connect the custom implementations of BackendListenerClient. The default is the implementation of graphite. Simple Data Writer Is the Most Useful Listener in JMeter. This saves performance based on the configuration inside the external file: the JTL file. JMeter JTL files are the best way to analyze the results, but come from the other side: you need another tool to perform data processing. Currently, there are two types of JTL files: CSV (default, with or without heads) and XML. XML files may contain more types of information, but much more. Therefore, it is recommended to stick to the CSV format. Produced by jmeter.jtl contains the same data: allThreads,Latency,IdleTime,Connect 150728028585,221,Home page,Number of samples in transaction : number of failed samples: 1,JPetstore 1-1,,,false,59592,10154,1,1,50,1,23 1507280286687.29,signinForm,200 ,OK,JPetstore 1-1,text,true,,,1531,582,1,29,0,0 1507280286108.29,Login page,200Cole samples in the transaction : number of failing samples : 0,JPetstore 1-1,,true,,1531,582,1,1,29,580,0 1507280286819,147,viewCatalog,200,OK,JPetstore 1-1,text,true,,3460,11027,1,1,27,0,0 1507280287967,233,signinAccount,200,OK,JPetstore 1-1,text,true,,3719,13270,1,1,55,0,27 1507280286717,380,Signin,200,Number of samples in transaction : 2, number of failing samples : 0,JPetstore 1-1,,true,,7179,24297,1,1,82,1104,27 1507280292035,162,viewCategory,200,OK,JPetstore 1-1,text,true,,2600,6502,1,1,56,0,26 1507280288201,162,ViewCategory,200,Number of samples in transaction : 1, number of failing samples : 0,JPetstore 1-1,,true,,2600,6502,1,1,56,3834,26 1507280297083,174,viewProduct,200,OK,JPetstore 1-1,text,true,,2643,6804,1,1,55,0,26 1507280292198,174,ViewProduct,200,Number of samples in transaction : 1, number of failing samples : 0,JPetstore 1-1,,true,,2643,6804,1,1,55,4886,26 1507280301651,162,addItemToCart,200,OK,JPetstore 1-1,text,true,,2827,6824,1,1,54,0,25 1507280304617,169,newOrderForm,200,OK,JPetstore 1-1,text,true,,3026,6804,1,1,55,0,27 1507280306851,173,setBillingInfo,200,OK .JPetstore 1-1,text,true,,2759,8194,1,1,63,0.28 15072803101018,163,confirmOr We will see later in this guide how we can use the results, stored in the JTL file for further processing and drilling. JTL is the most powerful way to analyze JMeter results. Pros: JTL are easy to read CSV files, some web tools are able to analyze JTL files and visualize online reports, all raw results are stored with JTL files. Cons: JTLs are written by every load generator on your drive. Distributed testing requires to return them to the controller at the end of the test, JTLs can grow large (multiple GB) and clutter the drive, JTLs must be mined using tools such as Excel to get useful metrics out of them. Let's see how we can interpret these JTL files. JTL analysis with Excel %%APACHE_JMETER_HOME%/extras contains several xsl files that are specifically designed to process JTL files in XML format and to report good reports. Look for the following files: jmeter-results-detail-report_21.xsl: JMeter Detailed Report, jmeter-results-report_21.xsl: Basic JMeter Report. The procedure below explains how to get good reports using these XSL styles and Microsoft Excel. How to analyze JTL files using Excel Simple Data Writer Listener: Add them to your test plan, customize it to save results like XML in the JTL file, run the load test: from APACHE_JMETER_HOME, run the command ./bin/jmeter-n-t jpetstore.jmx-lmeter j.jtl, Creating a summariser Created a Tree successfully using the zlt'summary'gt; jpetstore.jmx Starting test - Fri Oct 06 15:03:42 CEST 2017 (1507295022425) Waiting for a possible outage / StopTestNow / Heapdump message on port 4445 summary 12 at 00:00:18 - 0.7 /s Avg: 187 Min: 30 Max: 418 Err: 2 (16.67%) Active: 2 Started: 2 Finished: 0 cv and 27 in the zlt;/summary 0.9/s Avg: 168 Min: 29 Max: 270 Err: 2 (7.41%) Active: 2 Started: 4 Finished: 2 summary No 39 at 00:00:47 0.8/s Avg: 173 Min: 29 Max: 418 Err: 4 (10.26%) Summary No 33 at 00:00:31 - 1.1/Aug: 163 Min: 28 Max: 259 Err: 3 (9.09%) Active: 2 Start: 7 Finished: 5 summary No 72 at 00:01:18 0.9/s Avg: 169 Min: 28 Max: 418 Err: 7 (9.72%) Summary No. 27 at 00:29 - 0.9/Aug: 165 Min: 29 Max: 246 Err: 2 (7.41%) Active: 2 Started: 9 Finished: 7 summary No 99 at 00:01:47 - 0.9/s Avg: 168 Min: 28 Max: 418 Err: 9 (9.09%) Summary No 14 at 00:00:13 - 1.1/s Aug: 163 mins: 28 Max: 246 Err: 1 (7.14%) Active: 0 Start: 10 Finished: 10 Summary No 113 at 00:02:00 0.9/s Avg: 167 Min: 28 Max: 418 Err: 10 (8.85%) Cleaning ... - Fri Oct 06 15:05:43 CEST 2017 (1507295143106) ... end-of-launch Edit JTL: add the qlt;?xml-style type text/xsl href'PATH_TO_jmeter-results-report_21.xsl?'gt; after the JTL file's 1.0 encoding version inside it. Please note that it does not work with Open Office. Only Microsoft Office is supported. With the recently available JMeter report dashboard, this outdated solution is no longer as attractive. The report looks old-fashioned compared to the new JMeter HTML report available since JMeter 3.0. HTML Report DashBoard HTML Report Can be generated at the end of the test using a separate command line. This report is quite rich and displays many different metrics. A full list of all customizable settings can be found on the JMeter website. If you have a JTL containing all the results, run: ./bin/jmeter -g JTL_FILE -o OUTPUT_FOLDER Where: -g JTL_FILE: a relative or complete path to the JTL file. Example: jmeter.jtl, -o OUTPUT_FOLDER: the folder in which the HTML report should be written. It may take some time for the command line to run, depending on the size of the JTL file. Once completed, the bug should not appear in the terminal. The report is ready in this output folder. Pros: HTML Report is easy to create, graphics and tables are well designed, you can select/select queries and/or transactions on each chart. Cons: Too many settings, where to start? The report cannot be fully configured by adding text, images, and more. It's a static report. Starting with JMeter 3.0, HTML Report Dashboard is a huge step forward by simplifying JMeter's analysis of test results. Summary of the report contains the following information: the start time of testing and the time of completion, the APDEX assessment for each request and container, a pie chart called Summary of Requests, which gives a share of successful/failed samples. The Statistics Table provides global test statistics for each query performed: Execution: Number of hits and errors, samples: Total number of samples executed, KO: Total Unfinished Samples, Errors %: Percentage Response time (ms): Response time in milliseconds, Average: Average Past, Min: Minimum Past, Max: Maximum Past, 90th pct: 90th Percentile, 95th pct: 95th Percentile, 99th pct: 99th Percentile, Throughtput: Number of hits a second, Network: bandwidth in KB/sec Received: KB received in second, Sent: KB sent KB: KB sent second. Lines can be ordered by any of the above statistics, making it easy to find requests that cause bottlenecks. Requesting orders by reducing the average, you should see the slowest query, being first in the statistics table. The error table provides more information about the errors that you encounter during load testing. For each type of error, you'll see: Number of errors: how many errors occurred, % in errors: percentage of requests by error, % in all samples: Percentage of errors compared to total sample. Response Time Over Time Chart This chart shows the average response time of each transaction throughout the test. Unfortunately, if you have a lot of transactions, the schedule can look cluttered because all transactions are displayed on it. Response Time Percentili Active Threads, Bandwidth Over Time Active Threads, Bandwidth Over Time There are many other charts available: Bandwidth: Hits per second (except built-in resources): number of hits per second over time, codes per second (excluding built-in resources): HTTP codes per second over time (200 OK, 500 internal errors, etc.) Transactions per second: transactions (associated with the transaction controller) per second over time, Response Time Vs Query: Response time vs. queries per second, Delay Vs Request: Delay compared to requests per second, Response time: Response time: Response time: Last time per percentage , Response Time Review: Gives a percentage of requests for apdex range (Satisfaction, Tolerance and Disappointment), Time Vs Topics: Past time on active topics to see how past time deteriorates with increased load, the time of distribution of HTML Report responses is certainly a good step to catch up with some expensive tools such as LoadRunner or NeoLoad. Of course, this could be a way more customizable to taylor report that fits your needs. Anyway, this is a huge leap forward in improving the analysis of JMeter test results compared to the integrated listeners of the user interface. Given JMeter is an open source load testing tool available for free, I'm impressed to see how many tools there are to analyze test results. And we're not even done yet! JMeter's Backend Listener's Backend Lets external database to store test results and performance indicators. In this section, we'll combine several open source tools to collect and visualize JMeter results in real time: InfluxData: a database used as a temporary repository of metric metrics Store Performance Metrics, Grafana: Grafana is an open source platform for time series analytics that allows you to create real-time graphics based on time series data, JMeter's Backend Listener: Backend listeners collect JMeter metrics and send them to a temporary repository of metrics. Exposed Metrics JMeter sends metrics to the time-series database. The list below describes the metrics available. Thread metrics: ROOT_METRICS_PREFIX_ test.minAT: Minimum active threads, ROOT_METRICS_PREFIX_ test.maxAT: Maximum active threads, test ROOT_METRICS_PREFIX_.meanAT: Average active threads, ROOT_METRICS_PREFIX_.startedT test: Started threads, test ROOT_METRICS_PREFIX_.endT: Ready streams. Response time metric: ROOT_METRICS_PREFIX__ SAMPLER_NAME .ok.count: Number of successful responses to sampler's name, ROOT_METRICS_PREFIX__ SAMPLER_NAME .h.count: Server hits per second, this metric cumulatives sampling results and sub results (when using a transaction controller, Create a parent sampler should be out of control), ROOT_METRICS_PREFIX__ SAMPLER_NAME.ok.min: M ROOT_METRICS_PREFIX__ SAMPLER_NAME. : Maximum response time for successful responses to sampler name, ROOT_METRICS_PREFIX__ SAMPLER_NAME .ok.avg: Average response time for successful responses to sample name, ROOT_METRICS_PREFIX__ SAMPLER_NAME .ok.pct_PERCENTILE_VALUE: Percentyl, calculated for successful responses to sample names. There will be one metric for each calculated value, ROOT_METRICS_PREFIX__ SAMPLER_NAME .ko.count: Number of unsuccessful responses to sampler's name, ROOT_METRICS_PREFIX__ SAMPLER_NAME .ko.min: M.M. Response time to failed responses to sample name, ROOT_METRICS_PREFIX__ SAMPLER_NAME .ko.max: Max response time to failed responses to sample sample sample, ROOT_METRICS_PREFIX__ SAMPLER_NAME.ko.avg: Average response time for failed sample responses ,ROOT_METRICS_PREFIX__ SAMPLER_NAME .ko.pct_PERCENTILE_VALUE: A percile computer calculated for failed responses to sampler's name. For each calculated value will be one metric, ROOT_METRICS_PREFIX__ SAMPLER_NAME.a.count: Number of responses to samplers name (amount ok.count and ko.count), ROOT_METRICS_PREFIX__ SAMPLER_NAME .a.min: Ming response time for answers to samplers name (min ok.count and ko.count), ROOT_METRICS_PREFIX__ SAMPLER_NAME.a.max: Maximum response time for answers to samplers (maximum ok.count and ko.count ROOT_METRICS_PREFIX__ SAMPLER_NAME) : Average response time for samplers (avg ok.count and ko.count), ROOT_METRICS_PREFIX__ SAMPLER_NAME .a.pct_PERCENTILE_VALUE: Percentile, calculated for answers to sampler name. For each calculated value will be one metric. on the outcome for OK and failed samples). The next constants of the constant ROOT_METRICS_PREFIX_: root metrics. There is no case in using InfluxBackendListenerClient, SAMPLER_NAME: the sample name in the JMX script, PERCENTILE_VALUE: 90, 95 or 99 by default. Depends on the configuration of the listener's backend. Setting up InfluxDB We are going to download and install influxDB: Download InfluxDB, Install InfluxDB, here's the setup for Ubuntu with the Debian package: ubuntu@desktop: $ wget ubuntu@desktop: s sudo dpkg -i influxdb_1.3'amd64.deb Selection of previously unselected inflow packages. (Reading the database ... 264577 files and directories installed now.) Preparing to unpack influxdb_1.3.6'amd64.deb ... Unpacking (1.3.6-1) ... Set up (1.3.6-1) ... Created a simlink from /etc/systemd/system/influxd.service to/lib/systemd/system/influxdb.service. Created a simlink from /etc/systemd/system/multi-user.target.wants/influxdb.service to /lib/systemd/system/influxdb.service. ubuntu@desktop: Setting up An influxDB can vary depending on your operating system. For more information, visit the InfluxDB installation. Start the influxdb service by launching the influxdb service ubuntu@desktop: s sudo, launch a command flow in the terminal to connect to the database, Create a database JMeter: ubuntu@desktop: '$ inflow Connected to version 1.3.6 Version of the Shell InfluxDB: 1.3.6 zgt; show the name of the databases: database name ---- _internal Grafana is a dashboard that will allow you to use to visualize metrics sent by JMeter to the InfluxDB database. Download Grafana and install it (Ubuntu installation here): wget sudo dpkg-i grafana_4.5.2'amd64.deb View to open the grafana dashboard. Use the admin as an entry and password. Select Add DataSource, then set up DataSource with the following settings: Name: influxdb, any name should work, Type: InfluxDB, how we connect to the InfluxDB database, Url: Access: Direct, because it is a direct connection to the database, Database: jmeter, previously created database. BackendListener Setup Now, let's add a listener backend to our test plan: Open JMeter, then open a sample of the JMX script, right click on the test plan, and select Add and Listener' Backend Listener, set up the listener's backend with the following settings: influxdbMetricsSender: Implementation class to send metrics to InfluxDB. According to JMeter 3.2, InfluxDB is available without the need to add any additional plugins, influxDbUrl: the URL of the InfluxDB database, the InfluxDB URL database_name in the format: influxdb_host: How we created the jmeter database and we run it on a local machine with the default port, it's in our ours URL will be: app: app name. This option allows you to group the metrics by name, allowing you to use the same database for several different measurement tests: the name of the measurement to be stored in InfluxDB (the text line of InfluxDB is an internal protocol for storing metrics). Use 'jmeter' by default for this property, summaryOnly: put false if you want to keep detailed metrics in the database, samplesRegex: allows you to filter the results stored by the name of sampler, percentili: determines the percentile processed and sent to InfluxDB, 90;95;99 by default, testTitle: we use JPetstore here, eventTags: list of tags that will be stored in the event measure. Running the test now, it's time to run the test in JMeter. Either run the test in GUI mode or in a mode that is not in the wrong place. To check that the results are properly sent to InfluxDB, Launch the following command: curl ' db'jmeter --data-urlencode q'SHOW SERIES - results statement_id: Events, jpetstore app, name ApacheJMeter, jmeter, applicationjpetstore, responseCode-0,responsess Let's set up the Grafana dashboard to visualize hits/sec. Create JMeter Dashboard Select Create your first dashboard, Select graph, click on the name of the panel then edit, Now, let's set up the metrics: Data Source: Select data source InfluxDB previously configured, from: default jmeter, WHERE the application _interval and jpetstore, SELECT: field stake means (), which is the average number of samples, GROUP BY: Time It should produce a graph shown on the screenshot below. Hits/sec chart in Grafana using JMeter BackendListener NovaTec APM Dashboard Setting up the grafana dashboard itself is a tedious and difficult task, especially if you don't have advanced knowledge in query metrics. They published a pre-configured JMeter load testing panel. This dashboard only works with the following backend listener plugin: JMeter InfluxDB Writer Install JMeter InfluxDB Writer Create a special database This installation requires a separate database: Creating a new database in The InfluxDB named Novatek using the following command: ubuntu@desktop: $ curl-i -XPOST q'CREATE DATABASE novatec http /1.1 200 OK Connection: Close Content-Type: App/Json Request-Id: b04edfe5-acd4-11e7-8647-00000000000000000-Version: 1.3.6 Date: Mon, 09 Oct 2017 09:31:54 GMT Transfer-Coding: Chunked Results: statement_id:0 InfussBR Writer Open JMeter, then open a sample of the JMX script, right click on the test plan, and select Add the Listener to the Listener, Set up the listener's backend with the following settings: testName: jpetstore, nodeName: Test-Node, influxDBPort: 8086, influxDBUser: jmeter, inflow, dbPassword: no. Let other settings with default settings. JMeter BackendListener using NovaTec InfluxDB Writer Plugin Create new data-source Novatek in Grafana Create a new data source displayed on the novatec database. Import Novatec Dashboard Please follow the documentation explaining how to import Grafana Dashboard in detail. Open Grafana, Select import new dashboard, enter ID 1152, which is the id of the monitoring panel Novatec, Select the data source pointing to the novatec database. You should be able to see animated graphics in the dashboard. JMeter Novatec Dashboard in Grafana This dashboard offers many interesting metrics across graphs, pie charts and more: Active users: current running threads, Total bandwidth: Operations per second, Success Rate: Percentage of queries that have succeeded, Request Count: total number of queries executed, Error rate: percentage of requests that failed, Metrics Review: Table mapping of all metrics , and more! InfluxDB InfluxDB Studio is a user interface management tool for InfluxDB. It works on Windows and lets you manage InfluxDB databases with a user-friendly interface. Recommended setting, we strongly recommend using the NovaTec plugin in conjunction with the NovaTec JMeter dashboard. It provides an out-of-the-box dashboard with many interesting metrics ready for use. Setting up Grafana yourself can be difficult and requires knowledge of how InfluxDB requests work. Saas JMeter Solutions As we've seen, a complete work installation using BackendListener can take quite a bit of time to set up. And we don't even talk about maintenance and upgrades. That's why cloud solutions like OctoPerf, Blazemeter Flood.io appear. These Saas tools provide the ability to run JMeter tests and collect metrics. Each tool has its own reporting system based on patented technologies. We're going to explore every tool here and compare their reporting capabilities. The goal is to get an overview of the reporting capabilities of each JMeter cloud solution. Each tool will be used to run the same test: 20 simultaneous users, 10 min test duration, from anywhere available with a free account Please keep in mind that we try to be as objective as possible. There are many other tools on the market allowing JMeter analysis results. As a result, we have selected only the most popular tools. Blazemeter Blazemeter is the first tool that has appeared on the market to allow users to scale their load tests in the cloud. Blazemeter is an American company founded by Alon Girmonsky in December Starting test on Blazemeter Blazemeter Summary Report Summary Report provides the following stats: Max Users: Maximum number of simultaneous users, Avg Throughput: hits per second, Errors %: error percentage, average response time: average response time in milliseconds, 90% Response time: 90% Response time, average bandwidth: average KiB duration per second during the test. It includes two graphs: Load graph: displays hits/sec, errors/sec and simultaneous user curves, response time graph: displays simultaneous users and average response time curves. Summary static: Metrics cannot be added or removed. TimeLine Report TimeLine Report Timeline Report provides a huge graph whose curves can be configured. Deals can be selected individually and built. It's a little sad that the sampler hierarchy doesn't persist: all transactions and queries are within the same list. The chronology can get pretty messy if many queries are drawn at the same time. Query Statistics Statistics of Queries provides a table that contains global statistics for each transaction or request. Available: Sampling: Sample number, Avg Response time (ms): Average past time in milliseconds, 90th line (ms): 90% centile in milliseconds, 95th line (ms): 95% centile for the past time in milliseconds, 99th line (ms): 99% centile in past time in milliseconds, Min Response Time (ms): Minimum past time in milliseconds, Max Response Time (ms): Maximum past in milliseconds, Average KiB/sec: Network bandwidth (loading) in KilotesBy per second, Percentage The entire table can be downloaded as a CSV file for external processing. Statistics can be filtered over time. The Error Report of this report displays all errors received during the test run, classified by labels (pages) and types of errors. JMeter magazines JMeter magazines JMeter magazines JMeter magazines on the engine are available. Entries can be downloaded or viewed directly in the browser. The original Config test Original test configuration This section is a reminder of the original configuration of the test. The Executive Summary is a printed version of the test report. It contains all of the previous sections (Summary, TimeLine and more). Flood Flood blazemeter contender. This Australian company was founded in September 2013 by Ivan Vanderbyl and Tim Kupsman. They offer almost the same features as BlazeMeter: download the JMX script, run the test and analyze the results. Starting the Test on Flood IO TimeLine TimeLine includes the main graph with selected transactions TimeLine gives an overview test results. you can draw one transaction metric by selecting it in the table below. JMeter magazines JMeter magazines Live Tail and Download JMeter magazines can be viewed live while test test Entries can be downloaded at the end of the test. For details of the transaction/request information, by selecting one request or transaction, you get access to a sub-report that provides many metrics about the transaction. (Medium, Minimum, Maximum, Standard Deviation, Centiles, Passed vs. Failed and More) Multiple queries and responses are also stored at some random points during load testing. Different metrics can be downloaded as a CSV file for external processing. OctoPerf OctoPerf is a French load testing company founded in September 2016. The OctoPerf reporting system is a modular system developed that can be configured. Any of the paragraphs of the report below can be changed, making the reporting system dynamic. The report is pre-configured with certain elements of the report. Items can be added or removed as needed. Starting the OctoPerf test For more information, please read the documentation on the points of the report. Summary of the test Summary test displays detailed information about the test configuration, such as: Test duration, number of simultaneous users, geographic location used, and more. Statistical Summary Statistics provides a test of broad statistics. You can set up the following settings: the number of statistics displayed, the type of statistics that you should include. There are 30 metrics available. OctoPerf graphics of the reporting system can include an unlimited number of graphs, each configured differently. Each graph has customizable curves, 1 to 4 curves per graph. You can chart both performance and monitoring metrics, even on the same graph. The Results Table Table contains global statistics on a transaction or request. The Threshold Table displays threshold warnings and errors made during the test. The threshold is related to the monitoring function. Monitoring allows you to capture the monitoring metrics of backend servers. Top Chart Top Chart provides the top of the containers or http requests for this metric. This chart is great for drilling to find slow business transactions and/or queries. Pie Chart Pie charts are useful to get a quick overview of the http response code, http methods and http type media repartition. This allows you to quickly determine if the web application is working as expected. The Percentiles Percentiles charts show the point at which a certain percentage of observed values occurs. For example, the 95th percentile is a value that exceeds 95% of the observed values. The Error Table Error Table provides detailed information about each error that occurred during the test. This is understand what happened on the server side during load testing. Details of the error for each reported error can be viewed from the requested request and the response received from the server. JMeter JTL and OctoPerf logs allow you to view JMeter logs after a virtual user check or load test. You can also download the full magazine file clicking on the Download button. .log.gz loads when you click on it. You need a file compression tool like 7 zip to extract it. JMeter JTL files are also automatically centralized at the end of the test. Comparison tables Guess what? We have compiled a comparison table that directly compares the Top 3 JMeter Cloud Solutions market: OctoPerf Blazemeter Flood.io Recorder HAR Imports Single URL/REST Jmeter Import Gatling Import Correlations in Jmeter In Jmeter Variables In Jmeter In Jmeter Validate script In Jmeter In Jmeter Host file override SLA Sandbox (free test unit) 100 tests per month 10 tests only 5 hours only Bandwidth emulation Global only delay emulation Global only think Time Global Only in Jmeter Pacing Ramp Down Hits and RPS Load Policy In Jmeter Real Browser Monitoring LG Launch and Config Automatic Guide LG Monitoring Multiple JMX in One Test Preliminary Test Live Filter Tests Filter Duration Automatic SLAs APDEX Backup IP Po Default Views Good Average Usability Good Average Access Magazines Error Details with Details Edited Graphics One Chart Only Export PDF Exports CSV via JTL Customized Text Comparison Trends Report Unlimited 1 week to Unlimited 1 to 12 Months Jmeter The latest version of Jmeter is supported by several versions of Jmeter supported by one version of Jmeter only, currently not the last (3.1 instead of 3.3) feel free to ask for other features to be checked from the comments. There are many different ways to collect and display JMeter performance indicators. From DIY open source tools to proprietary solutions, there's a solution for everyone. What solution should I use? Here's a quick summary. UI Listeners: Great for debugging targets, you can use them for very small load tests (up to 50 simultaneous users), JTL Files and a simple data author: this solution can be used for distributed tests, although the configuration can be tedious. JTL files can then be analyzed using JMeter XSL sheets or html from Backend and Listener Listener and Grafana: This solution eliminates the tedious work of collecting and merging JTL files into distributed testing. It also provides live metrics during the test. But setting up is difficult, requires advanced knowledge and multiple systems need to be saved, Saas Solutions: The simplest and most powerful solution, but you have to pay for tests more than 50 simultaneous users on most platforms. Potentially, you can save a lot of time on setting up testing and analyzing the results. The decision chosen largely depends on the following factors: Time: Whether the testing phase is tough on time? Budget: Is the budget allocated to cover the cost of load testing? How much does the budget cost? Expertise: How much experience do you have in load testing? Open source and DIY solutions are usually free, but cost a lot Own solutions have value, but are a way of more efficient time. If you have the time, budget or even both, there is a solution for everyone. Book shelf If you want to master JMeter, we would like to recommend you some good books about JMeter. Learn JMeter in one day, Krishna Rungta Book begins with the introduction of Jmeter and performance testing. Gives detailed information steps to install Jmeter on different platforms. It goes to familiarize the reader with the Jmeter graphical interface. The book then teaches you how to create a performance test and improve the test with a timer, approval, controllers, and processor. JMeter CookBook, Bayo Erinle Leverage use existing cloud services for distributed testing and learn to use their own cloud infrastructure if necessary. JMeter's successful integration into a continuous delivery workflow that delivers high-quality products. Test application support services and resources including RESTful, SOAP, JMS, FTP and Database. Database. jmeter report in jenkins. jmeter report in html. jmeter report in pdf. jmeter report in excel. jmeter report in jenkins pipeline. jmeter report in non gui mode. jmeter report in seconds. jmeter report interpretation