Overview of the CloudXPRT Web Microservices Workload

This workload models a traditional three-tier web application with services in the web, application, and data layers. The workload uses Kubernetes, Docker, NGNIX, REDIS, Cassandra, and monitoring modules to mimic an end-to-end IaaS scenario.

September 9, 2020

CloudXPRT

BenchmarkXPRT

BenchmarkXPRT Development Community

Table of contents

Introduction

CloudXPRT is a cloud benchmark that can accurately measure the performance of applications deployed on modern infrastructure as a service (IaaS) platforms, whether those platforms are paired with on-premises (data center), private cloud, or public cloud deployments. Applications increasingly use clouds in latency-critical, highly available, and high-compute scenarios, so we designed CloudXPRT to use cloud-native components on an actual stack to produce end-to-end performance metrics that can help users determine the right IaaS configuration for their businesses.

CloudXPRT

- is compatible with on-premises, private, and public cloud deployments
- runs on top of cloud platform software such as Kubernetes and Docker
- supports multi-tier workloads
- reports relevant metrics such as throughput and critical latency for responsiveness-driven applications, and maximum throughput for applications dependent on batch processing

Testers can run CloudXPRT on local data center, Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure deployments.

CloudXPRT currently includes two workloads that users can install and run independently: web microservices and data analytics. This paper provides an overview of the CloudXPRT web microservices workload. In the sections below, you will find information about accessing the web microservices installation packages, the structure of the workload, its metrics, and test results. For detailed installation and test configuration instructions, consult the CloudXPRT web microservices readme document located in the CloudXPRT GitHub repository.

Workload summary

In the CloudXPRT web microservices workload, a simulated user logs into a web application that does three things: provides a selection of stock options, performs Monte Carlo simulations with those stocks, and presents the user with options that may be of interest. This scenario enables the workload to model a traditional three-tier web application with services in the web, application, and data layers. The workload uses Kubernetes, Docker, NGNIX, REDIS, Cassandra, and monitoring modules to mimic an end-to-end IaaS scenario.

The workload reports performance using a transactions-per-second metric that reflects the number of successful requests per second the stack achieves for each level of concurrency. Testers can use this metric to compare IaaS stack performance and to evaluate whether any given stack is capable of meeting SLA thresholds.

Getting started

MINIMUM SYSTEM REQUIREMENTS

We designed CloudXPRT for high-end servers. During each workload, the benchmark scales to utilize all the available cores. However, for functional testing, the physical node(s) or VM(s) under test must have at least:

- Ubuntu 18.04 LTS

- 16 logical or virtual CPUs
- 8 GB RAM
- 10 GB of available disk space
- An internet connection

For three of the four target platforms—on-premises, AWS, and GCP—testing requires both Docker and Kubernetes. Testing on Microsoft Azure requires only Kubernetes. Off-premises tests require access to an AWS, Azure, or GCP account, depending on the test configuration.

MULTIPLE WEB MICROSERVICES INSTALLATION PACKAGES

To simplify installation, we have separated the CloudXPRT installation packages by workload and target platform. There are five installation packages: four web microservices installation packages—one for each of the four target platforms (on-premises, AWS, Azure, and GCP)—and a single data analytics installation package, which is compatible with all four target platforms.

WHERE TO FIND THE INSTALLATION PACKAGES

All of the CloudXPRT installation packages are available through the CloudXPRT download table (see Figure 1) and through the CloudXPRT GitHub repository.

Workload	Target platform	Install package	Documentation
Data analytics	On premises/AWS/Azure/GCP	Download	Readme
Web microservices	On premises	Download	Readme
Web microservices	Amazon Web Services (AWS)	Download	Readme
Web microservices	Google Cloud Platform (GCP)	Download	Readme
Web microservices	Microsoft Azure	Download	Readme

Figure 1: The CloudXPRT installation package download table.

Installing and running the CloudXPRT web microservices workload

We do not include detailed installation or test setup information for the web microservices workload packages in this paper. These are available in the CloudXPRT web microservices readme document located in the CloudXPRT GitHub repository.

CONFIGURING THE TEST

CloudXPRT provides users with multiple test configuration options for the web microservices workload. Below, we list the workload's user-editable parameters, which testers can adjust by editing the config.json file located in \CloudXPRT_vXXX_web-microservices\cnbrun\.

- **Runoption.** This setting lets the user run only a specific microservice or all of the microservices individually (as opposed to in multitenancy). Currently, the Monte Carlo workload is the only available workload, so the default setting is "mc."

- **Iterations.** This setting lets the user select the number of iterations to run for each microservice. The default setting is 1. If a user designates 3, 5, 7, or 9 iterations, the postprocess binary (if the user has enabled it) produces a set of median values using the results from each run.
- **Hpamode.** With the default setting of false, the workload creates the maximum number of pods from the beginning of the run. If the user sets this option to true, the workload uses Kubernetes Horizontal Pod Autoscaler (HPA) to scale pods as the load increases.
- **postprocess**: With the default setting of false, the workload does not run the postprocess binary at the end of a test run. If the user sets this option to true, the workload runs the postprocess binary at the end of a test run.
- **ppoutputfile**: If the user sets the postprocess binary to run at the end of a test, this setting directs the benchmark to save the postprocess results to a file. The default setting is "".
- **autoloader.initialclients**: This setting lets the user select the initial number of clients the load generator will create. The default is 1.
- **autoloader.clientstep**: This setting lets the user select the number of clients to increase for each load generator iteration. The default is 1.
- **autoloader.lastclient**: This setting lets the user designate the number of clients after which the load generator stops. At the default setting of -1, the load generator continues to run until the cluster is saturated (i.e., CPU ~100%).
- **autoloader.SLA**: This setting lets the user designate the SLA for a 95th percentile latency constraint. The default setting is 3,000 milliseconds. If set to -1, there is no latency constraint.
- **autoloader.timeinterval**: This setting lets the user designate how long the load generator spends for each iteration with the specified number of clients. The default setting is 60 seconds.
- **workload.version**: This setting lets the user designate which Docker image version the test uses. The current default setting is v1.0.
- **workload.cpurequests**: This setting lets the user designate the number of CPU cores (integers only) the workload assigns to each pod. Currently, the benchmark supports values of 1, 2, and 4. The default setting is 4. Values of 1 and 2 are more appropriate for relatively low-end systems or configurations with few vCPUs.

Workload components

The web microservices workload comprises two main components: the load generator and the Kubernetes cluster (a set of node machines). The load generator produces transactions and the Kubernetes cluster receives, processes, and publishes these transactions.

LOAD GENERATOR

The load generator simulates a group of users connecting to a series of web-tier microservices, and the transactions the users generate. Each simulated user can issue only one request at a time, but the service can process requests from many users simultaneously.

By default, the load generator creates a single routine—a “light-weight” thread—that makes a single HTTP request to the cluster under test (CUT). The load generator then waits until this request receives a response from the CUT. Once it does, the load generator creates another request. This process continues until the end of the time interval specified in the configuration file (60 seconds by default). After the first request has completed, the load generator controls the number of routines the CUT can serve concurrently as the workload scales, with each routine making a single HTTP request and waiting for the response from the CUT.

THREE LAYERS OF MICROSERVICES

The web microservices workload models a traditional three-tier web application. It includes services in the web, application, and data layers. Table 1 describes the role of each layer's key components.

Web layer	
Web server	A NGINX web server configured as a reverse proxy for a Golang HTTPS web server that contains all the routes for all services and frontend logic
Application layer	
User profile management	Handles user login and profile creation
User data processing	Handles the encryption/decryption of user data
Stock options calculation	Executes Monte Carlo stock options calculations
Data layer	
Database management	Contains APIs that handle database transactions for Redis and Cassandra
Redis	Redis in-memory caches for each service in the application layer
Cassandra cluster	A Cassandra ring that consists of three pods used to store user profiles and their associated stock options

Table 1: The key components of CloudXPRT web-tier microservices.

MONTE CARLO SIMULATIONS

The stock options calculation service is a self-contained container with all the data it needs to run the Monte Carlo simulations. The model mimics European option pricing to compute call option prices. It uses Monte Carlo algorithms to calculate the value of an option with multiple sources of uncertainties and stochastic inputs, such as changing interest rates, stock prices, exchange rates, etc.

THE PATH OF A SINGLE ROUTINE

A single routine— a single HTTP request to the CUT—consists of the following sequential steps:

1) The load generator sends an HTTP request to the web server.
2) The web server uses profile management APIs to validate that the user exists in the database (because this is a benchmark, we pre-populated the database with user IDs).
3) A user data processing service decrypts the user profile data.
4) The web server asks the stock options service to conduct Monte Carlo simulations.
5) The benchmark parses the resulting stock options, and stores the results in the database.
6) The benchmark sends a JSON response back to the load generator.

Figure 2 shows the workflow and components (Kubernetes services, pods, and vCPU resources) of a CloudXPRT web microservices deployment in a 24x vCPU environment. Each hexagon represents a pod the benchmark has assigned to a specific task. The number of vCPUs the benchmark allocates for each pod is listed in each pod. 1000m = 1vCPU, 500m = 0.5vCPU. The rounded green shapes represent Kubernetes services that handle routing to individual pods. This Kubernetes management is especially important for any services that interact with more than one deployed pod.

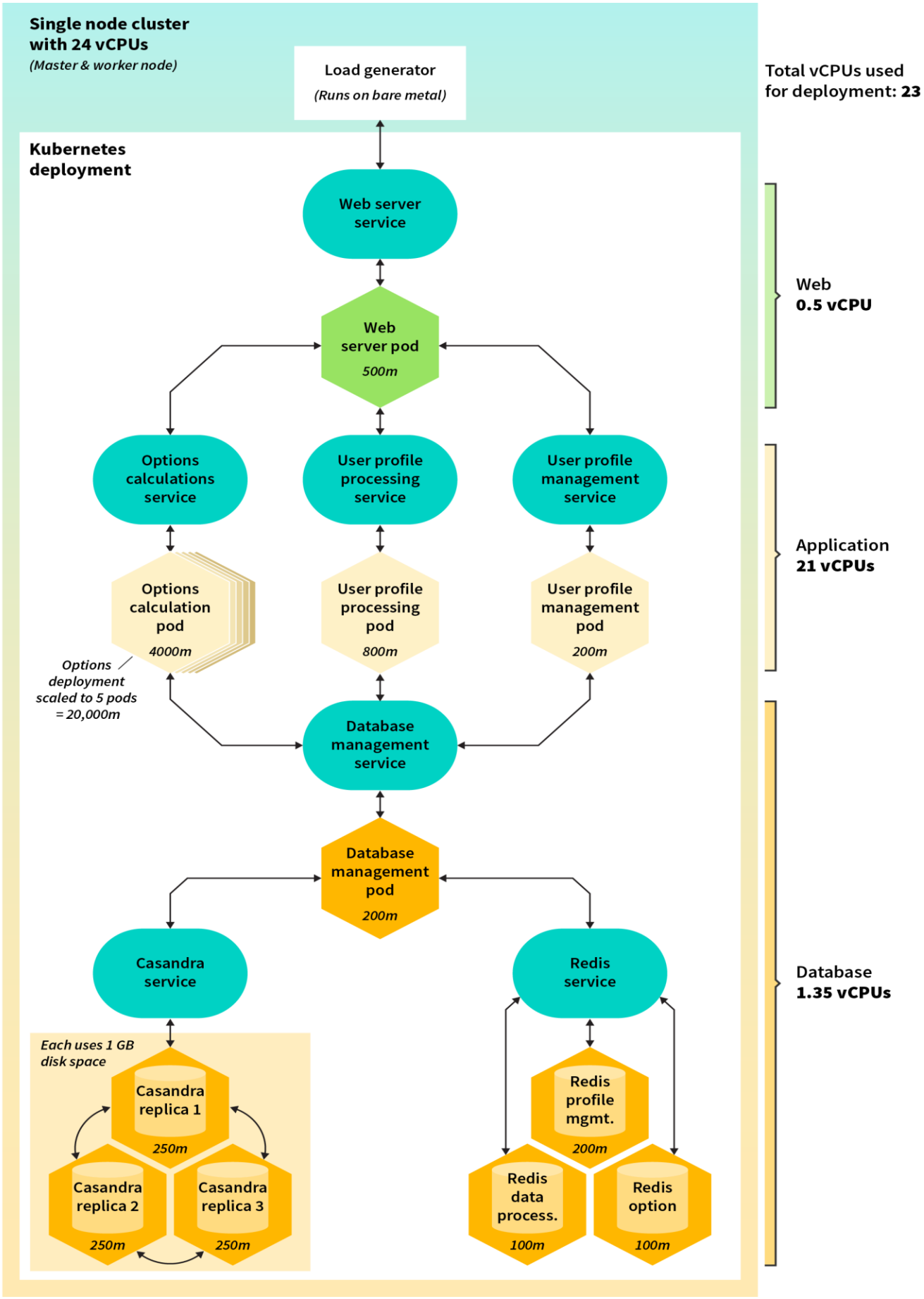

Figure 2: An example of a 24x vCPU deployment.

Workload metrics and results output

Currently, the CloudXPRT web microservices workload does not produce a single score. As is true with many workloads, the specific requirements of the user determine whether lower latency (response time), greater throughput, or a specific balance between the two constitutes the best score.

By default, the max SLA for 95th percentile latency is 3,000 milliseconds (specified by the 'autoloader.SLA' parameter in the config.json file). The load generator stops when (1) the CUT can no longer meet the specified SLA, or (2) the CUT cannot beat the current maximum number of successful requests within the next five attempts.

For each 60-second interval during a run, CloudXPRT reports data in eleven categories, which we present in Table 2. After each run, the benchmark records this data in a formatted table and saves it in the output directory with the file name "autoloader_mc_TIMESTAMP.log". Testers must review the results table and identify the metrics that are most relevant given their goals.

Data category	Description
Concurrency	The number of routines that ran concurrently
Requests	The number of requests made by routines
Successful requests	The number of requests that received responses with a successful HTTP status code
Failed requests	The number of requests that did not receive responses with a successful HTTP status code
Response mismatch	The number of responses that do not have an expected data field within the response payload
Successful request rate (requests/ second)	The number of successful requests divided by the time interval
Read throughput (bytes/second)	The read throughput rate for requests
Write throughput (bytes/second)	The write throughput rate for requests
Average CPU usage (%)	Average percentage CPU utilization for all nodes within the cluster
Time (seconds)	The time interval for the collected data
Response time (95th percentile) (milliseconds)	The 95th percentile latency in milliseconds for all requests made within this time interval

Table 2: CloudXPRT web microservices workload data collection categories.

For most testers, one of the following metrics will be an appropriate choice for comparing CUTs :

- The maximum number of successful requests per minute under a specified SLA or regardless of SLA.
- The highest rate of successful requests per second (SUCC_REQS_RATE (REQ/S) under a specified SLA or regardless of SLA.

Table 3 shows a condensed results table from a sample run. To determine throughput within SLAs of 1,000, 2,000, and 3,000 milliseconds, a tester could compare the number of successful requests that the CUT was able to execute within those times.

From the results, we can see that the CUT was able to handle:

- 615 requests per minute under 1,000 ms,
- 1,263 requests per minute under 2,000 ms, and
- 1,305 requests per minute under 3,000 ms

The max throughput within this run is 1,305 successful requests per minute.

Concurrency	SUCC_REQS	...	SUCC_REQS_RATE (REQ/S)	...	MC_RESP_TIME(95%ile)(MS)
1	70	...	1.17	...	888
2	142	...	2.37	...	876
...	...	...	...	...	...
7	470	...	7.83	...	968
8	547	...	9.12	...	932
9	**615**	...	10.25	...	**934**
10	559	...	9.32	...	1,499
...	...	...	...	...	...
28	1,214	...	20.23	...	1,625
29	1,246	...	20.77	...	1,656
30	**1,263**	...	21.05	...	**1,706**
31	1,250	...	20.83	...	2,050
...	...	...	...	...	...
33	1,260	...	21.00	...	2,398
34	1,291	...	21.52	...	2,213
35	**1,305**	...	**21.75**	...	**2,228**
36	1,174	...	19.57	...	4,039

Table 3: Condensed sample CloudXPRT web microservices results.

To determine the maximum rate of successful requests within a given SLA, a tester identifies the highest value in the SUCC_REQS_RATE (REQ/S) that corresponds with an acceptable MC_RESP_TIME(95%ile)(MS) value. For example, given the default SLA of 3000ms, the maximum rate of successful requests per second from the sample run is 21.75.

Additionally, testers can use all the values within the successful requests column to create a graph that illustrates workload performance from start to finish. We recommend plotting 95th percentile SLA latency (MC_RESP_TIME(95%ile)(MS)) versus throughput (SUCC_REQS).

Figure 3 displays data from a sample CloudXPRT web microservices test run on a random Kubernetes cluster (a different run than the one the created the data in Table 3). In this case, we can designate the best throughput (SUCC_REQS) score of 1,160 as the overall score for the cluster.

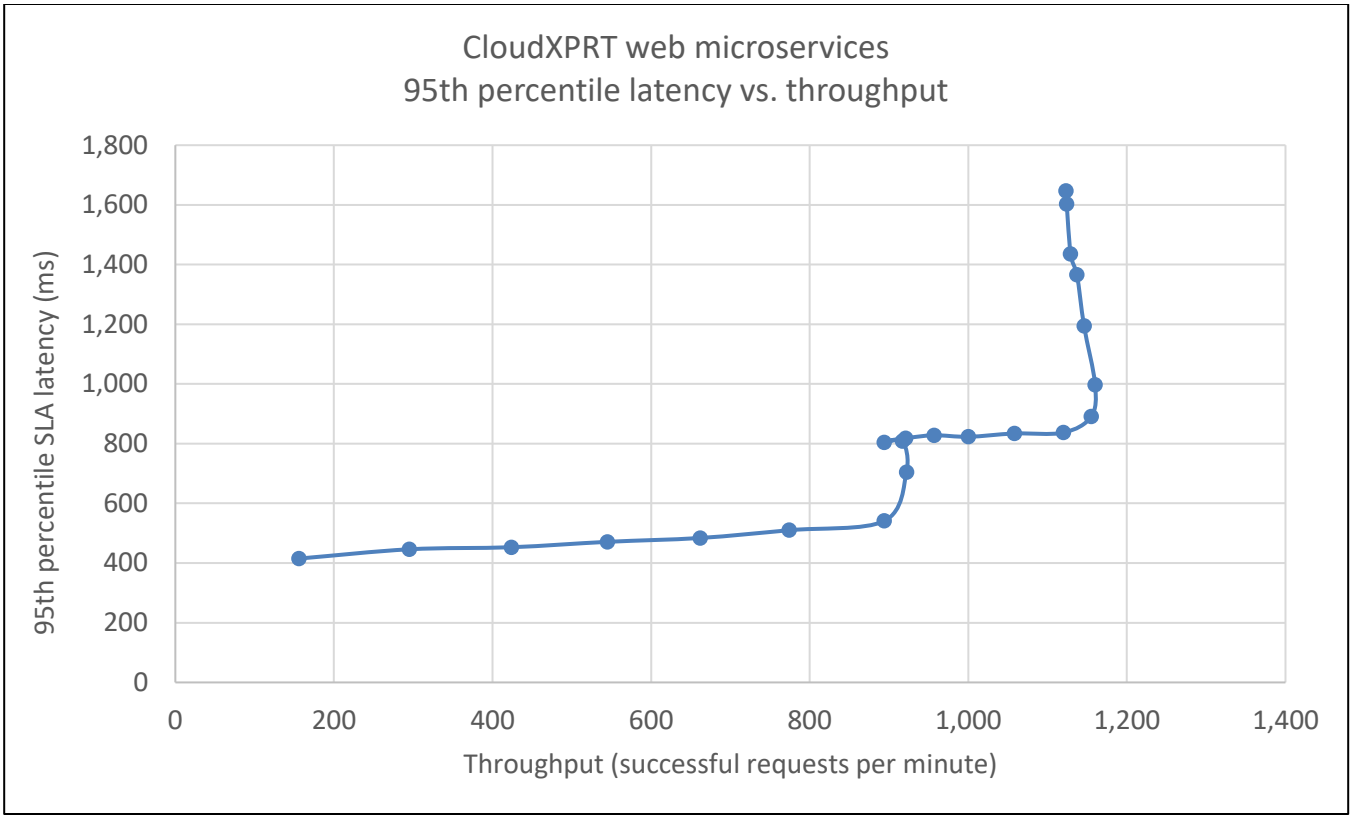

Figure 3: The latency vs. throughput matrix for a sample run.

Results submission and review

TEST RESULTS FILES

After each run, CloudXPRT saves results files on the node that ran the load generator. If you are testing a multi-node cluster, CloudXPRT will not save results files to the other nodes within the cluster. Results files will accumulate in the 'cnbrun/output' directory as you conduct runs, and CloudXPRT will not delete or overwrite them.

For each run, CloudXPRT automatically generates four files:

1. A log file with the results in a formatted table
2. A csv file with the results
3. A log file with all the stdout output from the run
4. A copy of the config file used for that run

SUBMITTING TEST RESULTS

We invite and encourage everyone to submit benchmark results from their testing for inclusion in the public CloudXPRT results table. To do so, please follow the steps below.

1. Create a folder to collect the files you will submit.
2. After a benchmark run completes, copy the following items to the folder you created in step 1:
   For the web microservices workload
   - The CSV results file located at CloudXPRT_vXXX_web-microservices/cnbrun/output/autoloader_mc_YYYYMMDD_*.csv.

- The log files located at CloudXPRT_vXXX_web-microservices/cnbrun/output/autoloader_mc_YYYYMMDD_*.log.
- The input run configuration file located at CloudXPRT_vXXX_web-microservices/cnbrun/config.json.

3. Locate the SystemInfo.csv file in the root directory of the installation package you used, and open it.
4. Complete the required fields, and describe any changes you made to customize scripts, models, or files for result generation. (Note: We collect this information to make it possible for others to reproduce the test and confirm that they get similar results.)
5. Save the SystemInfo.csv file to the folder you created in step 1, and create a single zip file containing all the files in this folder.
6. Create an email message to the BenchmarkXPRT Community Administrator:
   - Send to benchmarkxprtsupport@principledtechnologies.com.
   - Confirm that the reply-to address you specify is a valid address inside your organization.
   - Use the subject "CloudXPRT Results Submission."
   - In the body of the message, specify your company name and name of the person responsible for the test.
   - Attach the zip file you created in step 5.

When we receive your submission, we will verify your identity and validate the results. Due to the complexity of CloudXPRT tests, and to be as transparent and accurate as possible with our published results, we may ask follow-up questions about the tests and/or system configuration. If we decide to include your results in the public database, we will notify you.

THE RESULTS REVIEW GROUP AND SUBMISSION SCHEDULE

We are using a periodic results submission process for CloudXPRT. Each month, a results review group examines results submissions, and we publish all approved results on a scheduled day.

While testers are free to publish results outside of the monthly review process, we publish results on CloudXPRT.com only after first vetting them through the formal process. Our goal is to strike a balance between allowing the tech press, vendors, or other testers to publish CloudXPRT results on their own schedule, and simultaneously building a curated results database that OEMs and other parties can use to compete for the best results.

The results review group

The CloudXPRT results review group serves as a sanity check and a forum for comments on each month's submissions. All registered BenchmarkXPRT Development Community members who wish to participate in the review process can join the group by contacting us via email. We'll confirm receipt of your request and add you to the review group mailing list. Any non-members who would like to join the review group can contact us and we'll help you become community members.

The submission, review, and publication cycle

We update the CloudXPRT results database once a month on a published schedule. While testers can submit results through the CloudXPRT results submission page at any time, two weeks prior to each publication date, we close submissions for that review cycle. One week prior to each publication date, we email details of that month's submissions to the results review group, along with the deadline for sending post-publication feedback.

The publication schedule

We publish results to the database on the last business day of each month and close the submission window at 11:59 PM on the business day that falls two weeks earlier (with occasional adjustments for holidays). The schedule is available at least six months in advance on CloudXPRT.com. Due to holiday schedules, we do not publish results in December.

VIEWING RESULTS FROM OTHER TESTERS

At CloudXPRT.com, interested parties can view test results published by the BenchmarkXPRT Development Community. The following tips will help visitors navigate the results viewer:

- Click the tabs at the top of the table to switch from Data analytics workload results to Web microservices workload results.
- Click the header of any column to sort the data on that variable. Single click to sort A to Z and double-click to sort Z to A.
- Click the link in the Source/details column to visit a detailed page for that result, where you'll find additional test configuration and system hardware information and the option to download results files.
- By default, the viewer displays eight results per page, which you can change to 16, 48, or Show all.
- The free-form search field above the table lets you filter for variables such as cloud service or processor.

We will add more features, including expanded filtering and sorting mechanisms, to the results viewer in the future. We are also investigating ways to present multiple data points in a graph format, which will allow visitors to examine performance behavior curves in conjunction with factors such as concurrency and resource utilization.

Known issues

There are no known issues with the web microservices workload at the time of publication.

Conclusion

We hope this paper has answered any questions you have about the CloudXPRT web microservices workload. You will find additional information about other aspects of the benchmark in the *Introduction to CloudXPRT* white paper and on CloudXPRT.com. If you have additional questions, you need help with CloudXPRT, or you have suggestions on ways to improve CloudXPRT, send an email to our team at BenchmarkXPRTsupport@principledtechnologies.com.

About the BenchmarkXPRT family

The BenchmarkXPRT tools are a set of apps that help you test how well devices do the kinds of things you do every day. In addition to CloudXPRT, the BenchmarkXPRT suite currently comprises the following tools:

- AIXPRT, an AI benchmark tool that makes it easier to evaluate a system's machine learning inference performance by running common image-classification, object-detection, and recommender system workloads.
- WebXPRT, a browser benchmark that compares the performance of almost any web-enabled device

- CrXPRT, an app to test the responsiveness and battery life of Chromebooks
- HDXPRT, a benchmark to test how well Windows PCs handle real-world apps
- TouchXPRT, a Universal Windows Platform app to test the responsiveness of Windows 10 devices
- MobileXPRT, an app to test the responsiveness of Android devices

We designed the apps to test a wide range of devices on a level playing field. When you look at results from XPRTs, you get unbiased, fair product comparison information.

THE COMMUNITY MODEL

We built BenchmarkXPRT around a unique community model. Community membership is open to anyone, and there are many ways to participate. Members of the BenchmarkXPRT Development Community are involved in every step of the process. They give input on the design of upcoming versions, contribute source code, and help test the resulting implementation.

The community helps us avoid the ivory tower syndrome. Diversity of input during the design process makes the tests more representative of real-world activity. Giving community members access to the source code both improves the implementation of the design and increases confidence in the code.

The community model differs from the open source model primarily by controlling derivative works. It is important that the BenchmarkXPRT benchmarks return consistent results. If the testing community calls different derivative works by the same name, the result would be that test results would not be comparable. That would limit, if not destroy, the tools' effectiveness.

If you are not currently a community member, we encourage you to join! Our community is open to everyone, from software developers to interested consumers. Not only will you get early releases of future XPRTs, but you will also be able to influence the future of the tools. Register now, or for more information, see the BenchmarkXPRT FAQ.