Overview of the CloudXPRT Data Analytics Workload

This machine learning training workload evaluates how well an IaaS stack enables XGBoost to speed and optimize model training time.

January 28, 2021

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 data analytics workload. In the sections below, you will find information about accessing the data analytics installation package, the structure of the workload, its metrics, and test results. For detailed installation and test configuration instructions, consult the CloudXPRT data analytics readme document located in the CloudXPRT GitHub repository.

Workload summary

The CloudXPRT data analytics workload uses the gradient-boosting technique to classify a moderately large dataset with the XGBoost library. XGBoost is a gradient-boosting framework that data scientists often use for ML-based regression and classification problems. In the context of CloudXPRT, the purpose of the workload is to evaluate how well an IaaS stack enables XGBoost to speed and optimize model training. To do this, the data analytics workload uses Kubernetes, Docker, object storage, message pipeline, and monitorization components to mimic an end-to-end IaaS scenario. The workload reports latency (response time in seconds in the 95th percentile) and throughput (jobs per minute) rates. Testers can use this workload's metrics to compare IaaS stack performance and to evaluate whether any given stack is capable of meeting service -level agreement (SLA) thresholds.

Getting started

MINIMUM SYSTEM REQUIREMENTS

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

- Ubuntu 18.04 LTS
- Linux kernel 5.3.0-40-generic

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

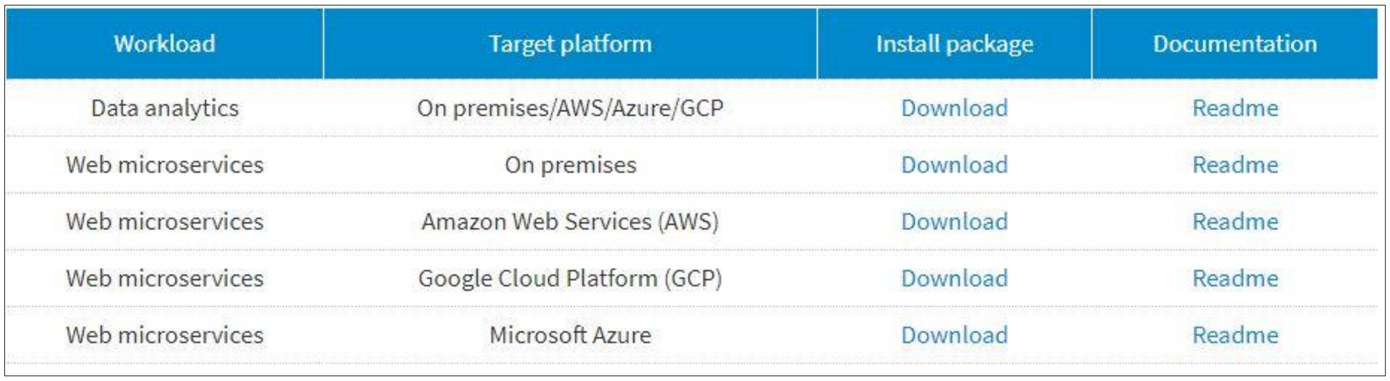
For all target platforms—on-premises, AWS, Azure and GCP—testing requires both Docker and Kubernetes. The installation script takes care of this configuration. Off-premises tests require access to an AWS, Azure, or GCP account, depending on the test configuration.

THE CLOUDXPRT INSTALLATION PACKAGE

Because CloudXPRT is such a complex benchmark, we have created five installation packages: a single data analytics installation package, which is compatible with all four target platforms (on-premises, AWS, Azure, and GCP), and four web microservices installation packages, one for each target platform.

WHERE TO FIND THE DATA ANALYTICS INSTALLATION PACKAGE

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 data analytics workload

We do not include detailed installation or test setup information for the data analytics workload package in this paper. This information is available in the CloudXPRT data analytics readme document located in the CloudXPRT GitHub repository.

CONFIGURING THE WORKLOAD

CloudXPRT provides users with multiple test configuration options for the data analytics workload. Below, we list the workload's user-editable parameters, which testers can adjust by editing the cnb-analytics_config.json file located in \CloudXPRT_vXXX_data-analytics\cnbrun\.

- **cpus_per_pod.** This setting lets the user designate the number of vCPUs per pod. The default setting is 12.
- **Loadgen_lambda.** This setting lets the user configure the interarrival time between transactions following the Poisson distribution. The default setting is 12 seconds.
- **NumKAFKAmessages.** This setting lets the user select the number of messages (transactions, jobs) that the workload will deliver and execute during the simulation. Each transaction is a request to the

main module to use the gradient-boosting technique to create a classification model and perform two inferences with the resulting data. The default setting for testing is 1. In most cases, users should use at least 100 messages to achieve a statistically sound result.

The workload includes a script (cnb-analytics_run-automated.sh) that creates a swept analysis to help you find the best throughput of a particular system.

Workload components

The data analytics workload incorporates five modules that interact with each other: a Kubernetes-based main module that acts as a cloud and container resource orchestrator, a load generator module, a messaging pipeline module, a monitorization module, and an object storage module. The load generator module produces transactions and the Kubernetes cluster module receives, processes, and publishes these transactions.

We constructed the workload using Go, Python, and Bash scripting.

THE MAIN MODULE

We utilize Kubernetes as a cloud and container resource orchestrator for the main module. For three of the four target platforms—on-premises, AWS, and GCP—the workload uses Docker to maintain images of each different test component.

The main module houses the computational pod, which receives requests from the load generator module and then executes transactions. In this context, a transaction is a request to the main module to use the gradient-boosting technique to create a classification model and perform two inferences with the resulting data. During the transaction, the main module must load the dataset (~2 GB) from the object storage module into memory so it can perform the corresponding computations. The main module executes and monitors each transaction seperately.

THE LOAD GENERATOR AND MESSAGE PIPELINE MODULES

The load generator module mimics real datacenter burst scenarios by using a Poisson distribution to send requests and create transactions that the cluster under test (CUT) will process. CloudXPRT users set the lambda parameter— the interarrival time between transactions—prior to the test. After transaction generation, the load generator delivers the transaction to the web server, which routes the transaction to the message pipeline module.

The message pipeline module utilizes Apache Kafka and Apache Zookeeper to handle transactions. Kafka provides a unified, high-throughput, low-latency platform for handling incoming transactions. Zookeeper provides critical services in the Kafka configuration by coordinating internal Kafka components and functions such as producers, brokers, consumers, cluster memberships, and leader election. Within Kafka, the data analytics workload uses up to 64 topics to increase parallelism in the multi-core systems.

During the workload, the message pipeline module uses Kafka to select an available pod to execute each transaction. If no pod is available, Kafka stores the message in a queue. Once a pod becomes available, Kafka delivers the message to the pod so it can execute the corresponding job. After execution, the main compute module uses Kafka to send a response back to the web server that routes the transaction back to the load generator. The load generator then receives the response and logs the transaction accordingly.

The load generator collects several epochs for every transaction, including generation time, queueing delay, execution time, and response time. The load generator stops when it receives the number of messages the CloudXPRT user requested. The workload also tracks when the CUT provisions or removes pods or services, to assess the cluster's availability for reuse.

THE MONITORIZATION MODULE

In the monitorization module, we use Prometheus to monitor workload and record metrics. Apart from workload metrics, we collect metrics related to the Kubernetes services, nodes, and orchestration status. We also use Node Exporter to observe system resource utilization on per-pod, per-instance, and per-node bases.

THE OBJECT STORAGE MODULE

For the object storage module, we use MinIO, a Kubernetes-native object storage suite. With MinIO, we can disaggregate the storage and compute modules, and guarantee that our stateless application can handle unstructured data programmatically. MinIO also provides protection against data corruption and bit rot issues by using an optimized hashing algorithm and an extensive data recovery mechanism based on erasure coding.

THE PATH OF A SINGLE TRANSACTION

A single transaction consists of the following sequential steps:

1. The load generator sends an HTTPS request to the web server.
2. The web server delivers the transaction to the message pipeline module.
3. The message pipeline module delivers the transaction to a working pod.
4. The working pod loads the dataset from the object storage module and uses the gradient-boosting technique to create a classification model and perform two inferences.
5. The working pod delivers results and statistics back to the message pipeline module.
6. The message pipeline module delivers the transaction back to the web server.
7. The web server delivers the transaction back to the load generator module.

Figure 2 shows the workflow and key modules of the CloudXPRT data analytics workload.

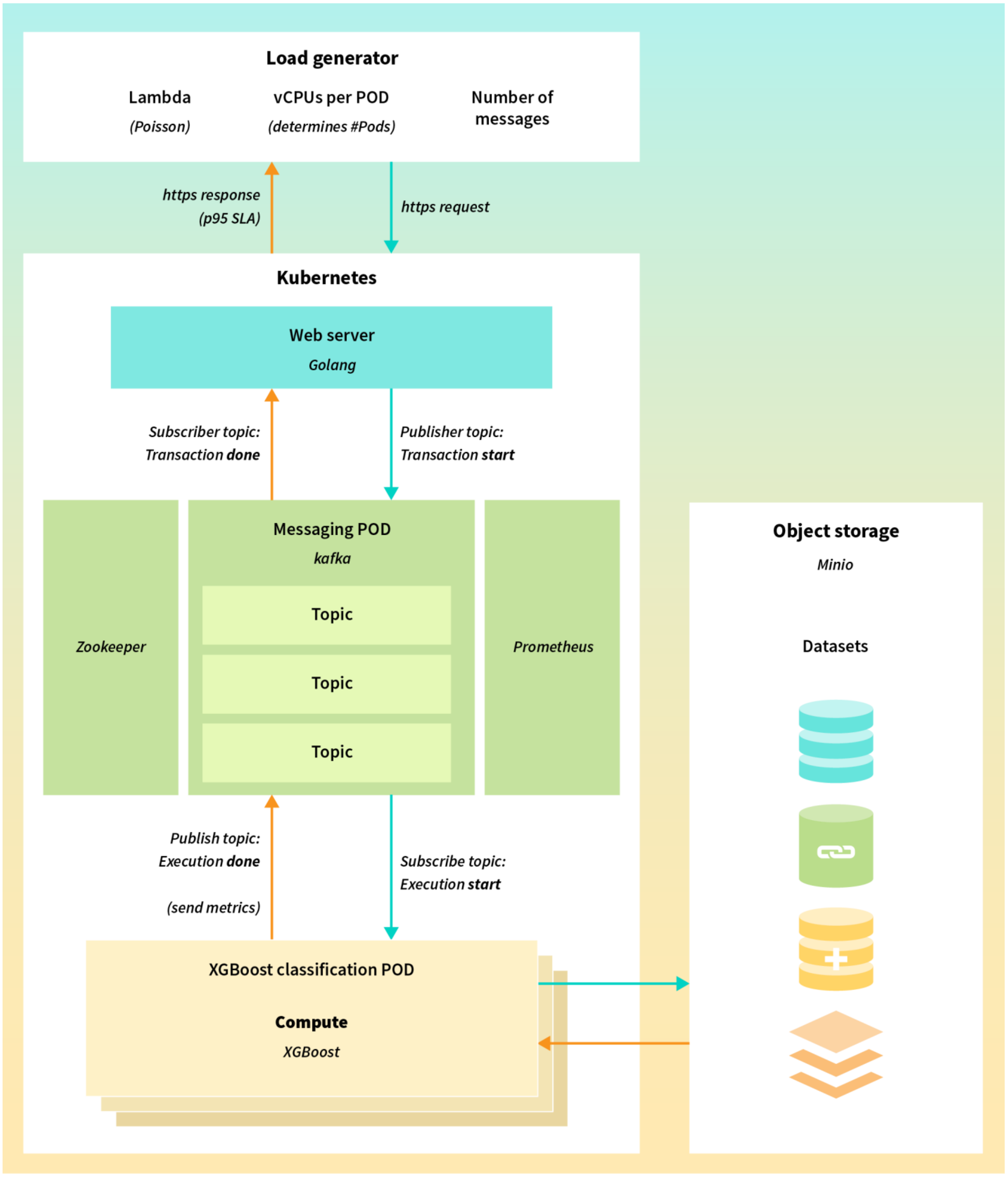


Figure 2: The CloudXPRT data analytics workload workflow.

Workload metrics and results output

Currently, the CloudXPRT data analytics 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.

For most users, we recommend recording the best performance (throughput_jobs/min) the CUT can deliver within a 95th percentile SLA latency of 90 seconds.

CloudXPRT reports output data in 13 categories, and testers must review the results table and identify the metrics that are most relevant to their goals. After each run, the cnb-analytics_parse-all-results.sh script records the 13 metrics in a formatted .csv table, and saves it in the output directory with the file name "output_result.txt."

The 13 key output metrics are as follows:

- **FILE.** This output reports the location of the output_result.txt file that contains the results from a designated simulation.
- **NumberOfPods.** This output reports the number of working pods that executed the XGBoost training task.
- **vCPUsperPod.** This output reports the number of vCPUs used per pod (while executing XGBoost training).
- **#Jobs.** This output reports the total number of jobs executed during the simulation.
- **Lambda.** This output reports the interarrival time between transactions in seconds. A lower lambda represents an increase in traffic sent to the CUT.
- **jobs_duration.** This output reports the elapsed time between the creation of the first transaction and the arrival of the last transaction at the load generator.
- **min_duration.** This output reports the minimum recorded transaction time.
- **max_duration.** This output reports the maximum recorded transaction time.
- **stdev_duration.** This output reports the standard deviation of recorded transaction times.
- **mean_duration.** This output reports the mean of recorded transaction times.
- **90th_Percentile.** This output reports the 90th percentile value for all recorded transaction times.
- **95th_Percentile.** This output reports the 95th percentile value for all recorded transaction times.
- **Throughput_jobs/min.** This output reports recorded throughput as expressed in transactions per minute.

Table 1 displays sample results from a series of test runs with two different configurations on the same CUT with 64 available vCPUs. In this case, the tester sought to find maximum throughput within a latency SLA of 90 seconds or less in the 95th percentile. The tester configured the pods with two different vCPUsperPod settings (20 vCPUs and 30 vCPUs). During the **first run** (20 vCPUs), the benchmark was able to spawn three pods using the 20 vCPUsperPod setting. The tester experimented with a range of lambda values, and the CUT was able to maximize throughput at 2.38 jobs per minute under the 90-second latency boundary. During the **second run**, the CloudXPRT harness was able to configure two pods with 30 vCPUsperPod each. In that configuration, the CUT provided throughput of up to 2.48 jobs per minute within the SLA boundary.

From these results, the tester can conclude that the second configuration would achieve the best throughput performance for the CUT within the designated SLA.

NumberOfPods	vCPUsperPod	Lambda	95th_Percentile	Throughput_jobs/min
3	20	30	104	2.52
3	20	31	103	2.46
3	20	**32**	**84**	**2.38**
3	20	34	73	2.26
3	20	36	68	2.15
3	20	40	63	1.96
3	20	50	58	1.55
2	30	30	135	2.52
2	30	**31**	**83**	**2.48**
2	30	32	66	2.39
2	30	34	59	2.26
2	30	36	62	2.16
2	30	40	60	1.96
2	30	50	52	1.56

Table 1: Sample data analytics results with 20 and 30 vCPUsperPod configurations

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 with a multi-node cluster, CloudXPRT does not save results files to the other nodes within the cluster. Results files accumulate in the 'CloudXPRT_vXXX_data-analytics/cnbrun' directory as you conduct runs, and CloudXPRT does not delete or overwrite them.

For each run, CloudXPRT automatically generates four files:

- A log file with the results in a formatted table
- A .csv file with the raw results
- A log file with all the stdout output from the run
- A copy of the config file 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:
    - The CSV results file located at CloudXPRT_vXXX_data-analytics/output/results.csv.
    - The log files located at CloudXPRT_vXXX_data-analytics/cnbrun/output/*.log.
    - The input run configuration file located at CloudXPRT_vXXX_data-analytics/cnbrun/cnb-analytics_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. because of the complexity of CloudXPRT tests and our wish 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 use 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, and 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 can help visitors navigate the results viewer:

- Click the tabs at the top of the table to switch from web microservices workload results to data analytics workload results.
- Click the header of any column to sort the data on that variable. A single click sorts in ascending order and a double click sorts in descending order.
- 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. You can change this to 16 results, 48 results, 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.

Notes and known issues

Please note the following:

- cnbrun, xgboost.sh, cnb-analytics_config.json must all be in the same directory.
- xgbooost.sh must have executable permissions.
- Some testers have observed the benchmark looping for an extended period of time while setting up "pod/services." To troubleshoot this issue, please open a new console and use the following script to verify that there are no errors or pods with an undefined status.
  ```
  su
  ./cnb-analytics_status.sh
  ```

Conclusion

We hope this paper has answered your questions about the CloudXPRT data analytics workload. You will find additional information about other aspects of the benchmark in the *Introduction to CloudXPRT* white paper, the *Overview of the CloudXPRT Web Microservices Workload* white paper, and on CloudXPRT.com. If you have additional questions, need help with CloudXPRT, or 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 were to call different derivative works by the same name, 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 be able to influence the future of the tools. Register now, or explore the BenchmarkXPRT FAQ to learn more.