Cloud Onload® Couchbase Cookbook

SF-123183-CD

Draft A

Table of Contents

1 Introduction

This chapter introduces you to this document. See:

1.1 About this document

This document is the *Couchbase Cookbook* for Cloud Onload. It gives procedures for technical staff to configure and run tests, to benchmark Couchbase utilizing Solarflare's Cloud Onload and Solarflare NICs.

This document contains the following chapters:

- Introduction on page 1 (this chapter) introduces you to this document.
- Overview on page 4 gives an overviews of the software distributions used for this benchmarking.
- Summary of benchmarking on page 7 summarizes how the performance of Couchbase has been benchmarked, both with and without Cloud Onload, to determine what benefits might be seen.
- Evaluation on page 10 describes how the performance of the test system is evaluated.
- Benchmark results on page 14 presents the benchmark results that are achieved.

and the following appendixes:

- Cloud Onload profiles on page 20 contains the Cloud Onload profiles used for this benchmarking.
- Installation and configuration on page 22 describes how to install and configure the software distributions used for this benchmarking.

1.2 Intended audience

The intended audience for this *Couchbase Cookbook* are:

- software installation and configuration engineers responsible for commissioning and evaluating this system
- system administrators responsible for subsequently deploying this system for production use.

1.3 Registration and support

Support is available from support@solarflare.com.

1.4 Download access

Cloud Onload can be downloaded from: https://support.solarflare.com/.

Solarflare drivers, utilities packages, application software packages and user documentation can be downloaded from: https://support.solarflare.com/.

The scripts and Cloud Onload profiles used for this benchmarking are available on request from support@solarflare.com.

Please contact your Solarflare sales channel to obtain download site access.

1.5 Further reading

For advice on tuning the performance of Solarflare network adapters, see the following:

- *Solarflare Server Adapter User Guide* (SF-103837-CD).

  This is available from https://support.solarflare.com/.

For more information about Cloud Onload, see the following:

- *Onload User Guide* (SF-104474-CD).

  This is available from https://support.solarflare.com/.

For more information about Couchbase, see the following:

- The Couchbase Server documentation.

  This is available from https://docs.couchbase.com/server/current/introduction/intro.html.

For more information about YCSB, see the following:

- The YCSB Wiki.

  This is available from https://github.com/brianfrankcooper/YCSB/wiki.

- *Benchmarking Cloud Serving Systems with YCSB*.

  This is available from https://www2.cs.duke.edu/courses/fall13/compsci590.4/838-CloudPapers/ycsb.pdf.

2 Overview

This chapter gives an overview of the software distributions used for this benchmarking. See:

- Couchbase overview on page 4
- YCSB overview on page 5
- Cloud Onload overview on page 5.

2.1 Couchbase overview

Couchbase Server is an open source distributed NoSQL document database that provides low latency data management for large scale, interactive online applications. It is designed to scale, especially horizontally, with ease and without performance degradation. Built with a strong emphasis on reliability, high availability, and simple management, Couchbase serves data non-stop with minimal human intervention.

Couchbase Server can be used as:

- a managed cache tier
- a key-value store
- a document database.

Couchbase Server along with Couchbase Mobile enables enterprises to increase business agility, achieve faster time to market, and operate at a global scale while reducing costs. Couchbase helps meet the requirements of multiple use cases ranging from enterprise to Cloud infrastructure, to Internet of Things and big data, to mobile devices.

Couchbase is heavily network dependent by design, so its performance can be significantly improved through enhancements to the underlying networking layer.

2.2 YCSB overview

The Yahoo! Cloud Serving Benchmark (YCSB) project develops a framework and common set of workloads for evaluating the performance of different “key-value” and “cloud” serving stores. The project comprises two things:

- the YCSB Client, an extensible workload generator
- the Core workloads, a set of workload scenarios to be executed by the generator.

Although the core workloads provide a well rounded picture of a system’s performance, the Client is extensible so that new and different workloads can be defined to examine system aspects, or application scenarios, not adequately covered by the core workload. Similarly, the Client is extensible to support benchmarking additional databases.

A common use of the tool is to benchmark multiple systems and compare them. For example, multiple systems can be installed on the same hardware configuration, and the same workloads can be run against each system.

2.3 Cloud Onload overview

Cloud Onload is a high performance network stack from Solarflare (https://www.solarflare.com/) that dramatically reduces latency, improves CPU utilization, eliminates jitter, and increases both message rates and bandwidth. Cloud Onload runs on Linux and supports the TCP network protocol with a POSIX compliant sockets API and requires no application modifications to use. Cloud Onload achieves performance improvements in part by performing network processing at user-level, bypassing the OS kernel entirely on the data path.

Cloud Onload is a shared library implementation of TCP, which is dynamically linked into the address space of the application. Using Solarflare network adapters, Cloud Onload is granted direct (but safe) access to the network. The result is that the application can transmit and receive data directly to and from the network, without any involvement of the operating system. This technique is known as “kernel bypass”.

When an application is accelerated using Cloud Onload it sends or receives data without access to the operating system, and it can directly access a partition on the network adapter.

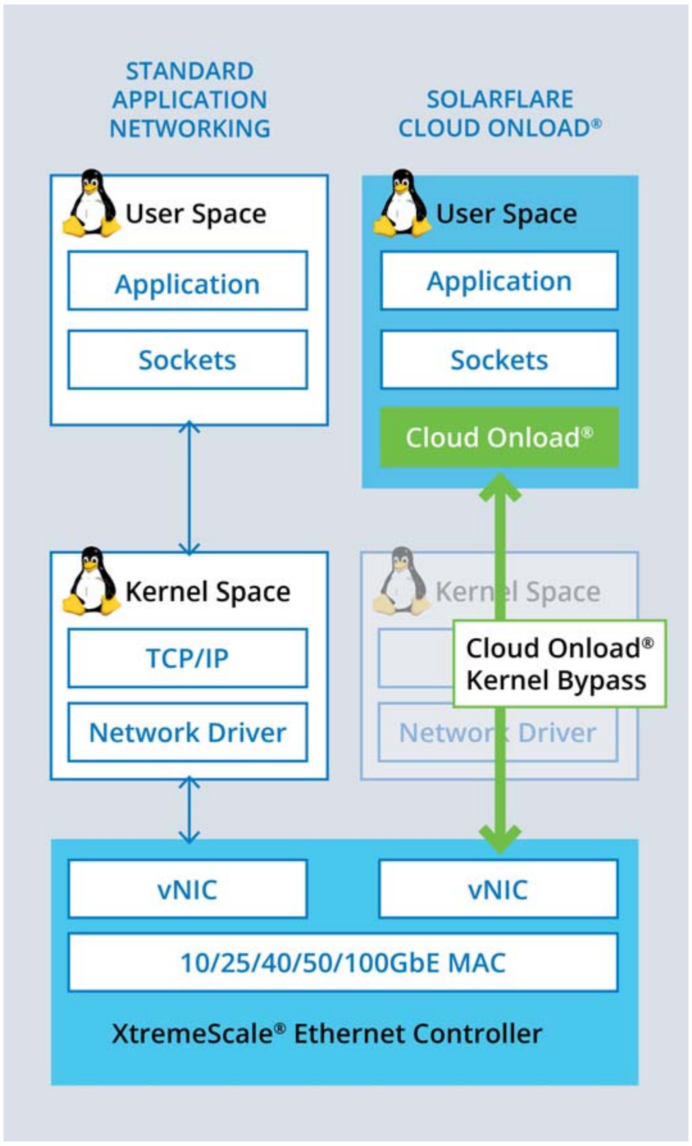


Figure 1: Cloud Onload architecture

3 Summary of benchmarking

This chapter summarizes how the performance of Couchbase has been benchmarked, both with and without Cloud Onload, to determine what benefits might be seen. See:

- Overview of Couchbase benchmarking on page 7
- Architecture for Couchbase benchmarking on page 8
- Couchbase benchmarking process on page 9.

3.1 Overview of Couchbase benchmarking

The Couchbase benchmarking uses two servers:

- The *Couchbase server* runs multiple instances of Couchbase to service requests.
- The *benchmarking client* runs multiple instances of the YCSB software to measure the performance of Couchbase.

Various benchmark tests are run on the benchmarking client, with the Couchbase server using the Linux kernel network stack.

The tests are then repeated, using Cloud Onload to accelerate the Couchbase server.

The results using the kernel network stack are compared with the results using Cloud Onload.

3.2 Architecture for Couchbase benchmarking

Benchmarking was performed with two servers, with the following specification:

	Couchbase server	**Benchmarking client**
Server	Dell R740	Dell R740
Memory	192GB	192GB
NICs	SFN8522 (dual port 10G) X2522-25G (dual port 25G) X2541 (single port 100G)	SFN8522 (dual port 10G) X2522-25G (dual port 25G) X2541 (single port 100G)
CPU	2 × Intel® Xeon® Gold 5120 CPU @ 2.20GHz	2 × Intel® Xeon® Platinum 8153 CPU @ 2.00GHz
OS	Red Hat Enterprise Linux Server release 7.5 (Maipo)	Red Hat Enterprise Linux Server release 7.5 (Maipo)
Software	Couchbase Server 6.0.2-2413	YCSB

Each server is configured to leave as many CPUs as possible available for the application being benchmarked.

Each server has 2 NUMA nodes. 3 Solarflare NICs are fitted, all affinitized to the same NUMA node, and connected via a switch to the corresponding NICs in the other server:

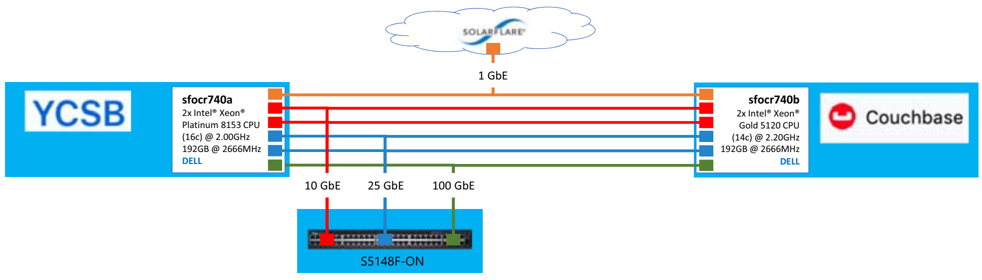


Figure 2: Architecture for Couchbase benchmarking

3.3 Couchbase benchmarking process

These are the high-level steps we followed to complete benchmarking with Couchbase.

- Install Couchbase on the Couchbase server, and verify correct operation.
- Install YCSB on the benchmark client, and verify connectivity to the Couchbase server (benchmark results are displayed).
- Start Couchbase on the Couchbase server, with a newly created bucket.

NOTE: The performance of Couchbase Server is tested using Couchbase buckets, which are persistent. The other types of buckets provided by Couchbase (Memcached and Ephemeral) are not persistent on disk, and only exist in RAM.

- Load the data for the YCSB workload.
- Execute the YCSB workload.

  Record the response rate of the Couchbase server, as the number of operations per second.
- Increase the number of threads for YCSB, and repeat the test.

  Continue doing this until 50 threads are in use.
- Repeat all tests across all interfaces available on the servers.
- Repeat all tests, accelerating Couchbase with Cloud Onload.

These steps are detailed in the remaining chapters of this *Cookbook*.

The scripts and Cloud Onload profiles used for this benchmarking, that perform the above steps, are available on request from support@solarflare.com.

4 Evaluation

This chapter describes how the performance of the test system is evaluated. See:

- Starting Couchbase for kernel benchmarking on page 10
- Running YCSB for kernel benchmarking on page 12
- Cloud Onload benchmarking on page 13
- Graphing the benchmarking results on page 13.

4.1 Starting Couchbase for kernel benchmarking

For each test run, you must delete/create a new bucket:

1 Start the `couchbase-server` executable:

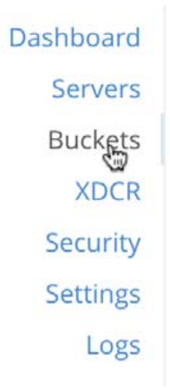
```
/opt/couchbase/bin/couchbase-server -- -noinput
```

2 Connect to the Web Console for the Couchbase Server. This is an embedded web server on port 8091. It can be accessed from a web browser either by hostname or by IP address as follows:

- http://<Couchbase server hostname>:8091/
  For example, http://sfocr740b:8091/
- http://<Couchbase server IP address>:8091/
  For example, http://10.20.128.35:8091/

Login as a user with administrative rights.

3 Click on the **Buckets** tab, in the vertical navigation-bar at the left-hand side of the Web Console:

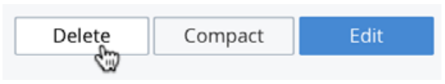


The **Buckets** screen appears, showing any buckets that are currently defined.

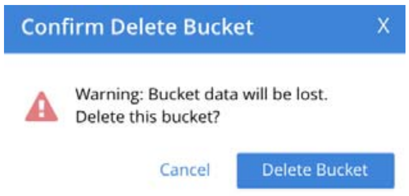
4 Delete all buckets that are shown. For each bucket:

a) Click on the row giving details of the bucket.

b) Click on the **Delete** button that appears:



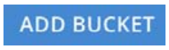
c) In the **Confirm Delete Bucket** dialog that appears, click **Delete Bucket**:


Confirm Delete Bucket

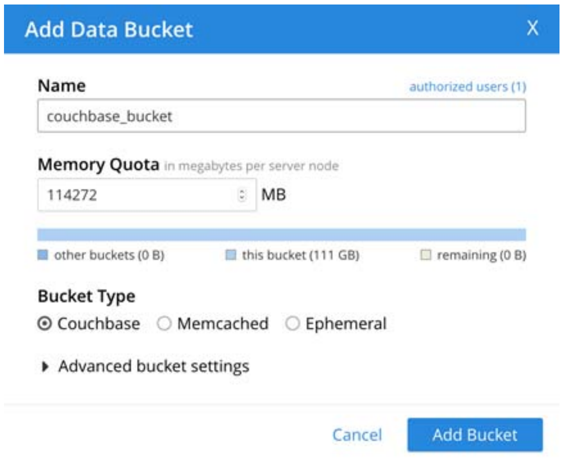
Warning: Bucket data will be lost.
Delete this bucket?

The bucket is deleted, and disappears from the Web Console.

5 Click on the **Add Bucket** button, at the upper-right of the Web Console:



The **Add Data Bucket** dialog appears:


Add Data Bucket

Name: couchbase_bucket

Memory Quota in megabytes per server node: 114272 MB

other buckets (0 B) this bucket (111 GB) remaining (0 B)

Bucket Type: [✓ Couchbase] Memcached Ephemeral

6 Ensure the **Name** of the bucket is set to `couchbase_bucket`, and the **Bucket Type** is set to **Couchbase**.

7 Click **Add Bucket**.

4.2 Running YCSB for kernel benchmarking

The workload for Couchbase is created using the `couchbase2` workload executer, found in the YCSB repository.

To run YCSB:

1 Create a copy of the `workloada` file, changing the record and operation counts to 1000000:

```
# cp workloads/workloada workloads/workloadA
# vi workloads/workloadA
...
recordcount=1000000
operationcount=1000000
...
```

2 Set an environment variable with the IP address of the Couchbase server interface for testing:

```
# SERVER_IP=<Couchbase server IP address>
```

For example:

```
# SERVER_IP=10.20.128.35
```

3 Set an environment variable with the options for YCSB:

```
# OPTIONS=" -s -P workloads/workloadA -p couchbase.host=$SERVER_IP
  -p couchbase.bucket=couchbase_bucket -p couchbase.password=adminis"
```

4 Restart Couchbase with a newly created bucket.

See Starting Couchbase for kernel benchmarking on page 10.

5 Set the number of YCSB threads for this iteration. For the first iteration, this is 5:

```
# THREADS=5
```

6 Prepare for running the workload by loading the data:

```
# ./bin/ycsb load couchbase2 $OPTIONS -threads $THREADS
```

7 Run the workload:

```
# ./bin/ycsb run couchbase2 $OPTIONS -threads $THREADS
```

8 Repeat steps 4 to 7, incrementing the number of YCSB threads in steps of 5, to a maximum of 50.

9 Repeat steps 2 to 8, iterating over the IP addresses of the different Solarflare NICs installed on the Couchbase server.

4.3 Cloud Onload benchmarking

All benchmarking is then repeated using Cloud Onload to accelerate Couchbase and YCSB. This uses the Cloud Onload `redis-balanced.opf` profile

Using the redis-balanced profile

Precede each command with:

```
onload --profile=redis-balanced
```

So, when starting Couchbase:

```
# onload --profile=redis-balanced \
         /opt/couchbase/bin/couchbase-server -- -noinput
```

and when running YCSB:

```
# onload --profile=redis-balanced \
        ./bin/ycsb load couchbase2 $OPTIONS -threads $THREADS
```

```
# onload --profile=redis-balanced \
        ./bin/ycsb run couchbase2 $OPTIONS -threads $THREADS
```

4.4 Graphing the benchmarking results

The results from each iteration of YCSB are now gathered and summed, so that they can be further analyzed. They are then transferred into an Excel spreadsheet, to create graphs from the data.

5 Benchmark results

This chapter presents the benchmark results that are achieved. See:

- Results on page 14
- Analysis on page 19.

5.1 Results

This section shows the results of using different numbers of threads for the YCSB tests.

The results show both kernel and Cloud Onload runs across different Solarflare NICs.

The results utilizing Cloud Onload use its `redis-balanced.opf` profile. See Cloud Onload benchmarking on page 13.

Throughput

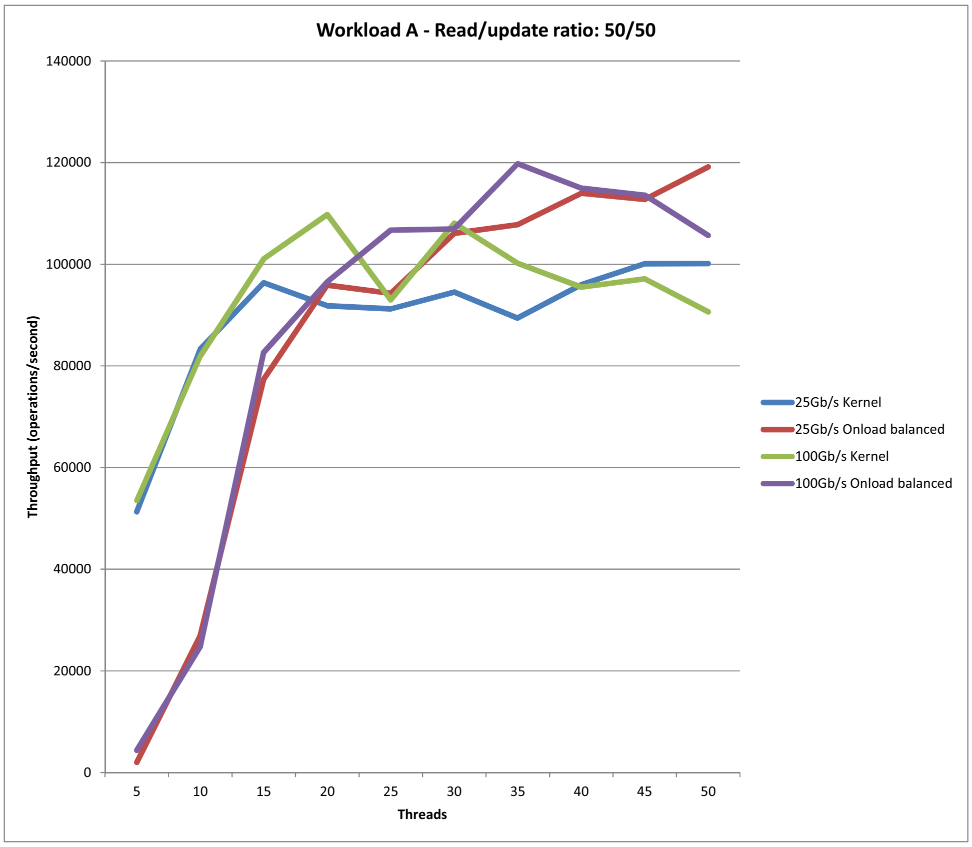


Figure 3: Couchbase throughput in operations per second

Table 1 and Table 2 below shows the results that were used to plot the graph in Figure 3 above:

Table 1: Throughput in operations per second at 25Gb/s

YCSB threads	Kernel	Onload balanced	Onload balanced gain
5	51293	1997	-96.11%
10	83319	26946	-67.66%
15	96339	77417	-19.64%
20	91802	95923	4.49%
25	91241	94277	3.33%
30	94518	106078	12.23%
35	89389	107793	20.59%
40	95979	113973	18.75%
45	100110	112727	12.60%
50	100130	119147	18.99%

Table 2: Throughput in operations per second at 100Gb/s

YCSB threads	Kernel	Onload balanced	Onload balanced gain
5	53527	4356	-91.86%
10	81927	24807	-69.72%
15	101071	82686	-18.19%
20	109782	96534	-12.07%
25	93023	106724	14.73%
30	108085	106929	-1.07%
35	100180	119775	19.56%
40	95493	114956	20.38%
45	97144	113585	16.92%
50	90629	105686	16.61%

Latency

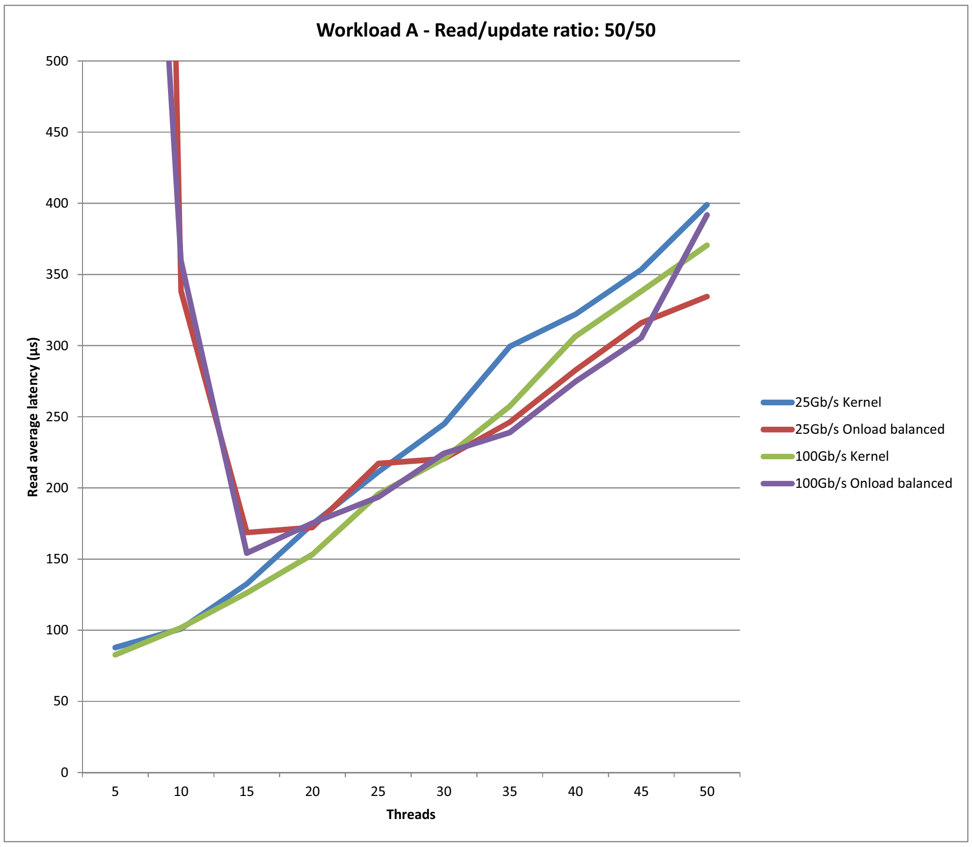


Figure 4: Couchbase read average latency in µs

Table 3 and Table 4 below shows the results that were used to plot the graph in Figure 4 above.

Table 3: Read average latency in µs at 25Gb/s

YCSB threads	Kernel	Onload balanced	Onload balanced gain
5	88	2441	2681.55%
10	101	338	235.03%
15	133	169	27.20%
20	175	172	-1.36%
25	211	217	2.76%
30	245	220	-9.96%
35	299	246	-17.78%
40	322	283	-12.19%
45	353	316	-10.59%
50	399	334	-16.20%

Table 4: Read average latency in µs at 100Gb/s

YCSB threads	Kernel	Onload balanced	Onload balanced gain
5	83	1123	1258.14%
10	102	360	253.41%
15	126	154	22.09%
20	153	175	14.38%
25	196	194	-1.03%
30	221	224	1.64%
35	257	239	-7.15%
40	306	275	-10.35%
45	338	305	-9.65%
50	371	392	5.71%

5.2 Analysis

When compared with the kernel stack, Cloud Onload delivers significant improvements.

- With small numbers of threads, Cloud Onload delivers massively improved latency, by a factor of up to 2681.55%.
- With larger numbers of threads, Cloud Onload delivers a useful improvement in throughput, by a factor of up to 20.59%.

Comparing Cloud Onload vs kernel, requests to the database are processed faster. This is because Couchbase receives data straight from the network directly into its memory, without a detour through the kernel. This direct path reduces memory copies, eliminates kernel context switches, and removes other system overhead. The result is a dramatic reduction in time, and CPU cycles. Similarly, when Couchbase fulfills a database request it can write that data directly to the network. This again saves more time, and reclaims more CPU cycles.

As more CPU cycles are freed up due to decreased latency, those compute resources go directly back into processing additional Couchbase database requests.

A Cloud Onload profiles

This appendix contains the Cloud Onload profile used for this benchmarking. See:

- The Redis Cloud Onload profiles on page 20.

This profile, along with the scripts used for this benchmarking, is available on request from support@solarflare.com.

A.1 The Redis Cloud Onload profiles

The benchmarking for Couchbase uses the balanced profile for the similar Redis application.

There are two Redis Cloud Onload profiles.

- The *performance profile* (not used for this benchmarking) is designed to get maximum performance, including best throughput and transaction rate, as well as best average and 99 percentile transaction response times.

  However this profile relies on the application threads having exclusive use of physical CPU cores. To get best performance, you must explicitly pin application threads to physical cores (avoiding threads sharing hyperthreaded CPU cores), and also ensure there are enough unused CPU cores for other applications to use.

  This profile constantly polls for network events to achieve the lowest latency possible, and so has higher CPU usage. CPU utilization metrics no longer provide a usable indication of system load.

- The *balanced profile* (used for this benchmarking) is designed also to get maximum performance, while avoiding the trade-offs associated with the performance profile.

  This profile does not rely on the application threads having exclusive use of physical CPU cores. CPU cores may be shared with other applications, hyper-threads may be used, and CPU utilization metrics indicate system load.

  Under high load conditions, this profile should deliver throughput and transaction rates that are equivalent to the performance profile. At lower traffic rates, CPU usage is reduced, but transaction response times might increase. However, response times will be better than when not running Onload.

The redis-balanced profile

The `redis-balanced.opf` Cloud Onload profile is as follows:

```
# redis balanced profile
#
# Enable small amount of polling / spinning. When the application makes a blocking call
# such as recv() or poll(), this causes Onload to busy wait for up to 20us
# before blocking.
onload_set EF_INT_DRIVEN 0
onload_set EF_POLL_USEC 20
# Prevent spinning inside socket calls.
onload_set EF_PKT_WAIT_SPIN 0
onload_set EF_TCP_RECV_SPIN 0
onload_set EF_TCP_SEND_SPIN 0
onload_set EF_TCP_CONNECT_SPIN 0
onload_set EF_TCP_ACCEPT_SPIN 0
onload_set EF_UDP_RECV_SPIN 0
onload_set EF_UDP_SEND_SPIN 0
# Use EPOLL mode 2 as will provide the best scalability and speed
# and be compatible with redis process forking
# EPOLL can be multithread safe, as redis poll architecture is single threaded
onload_set EF_UL_EPOLL 2
onload_set EF_EPOLL_MT_SAFE 1
onload_set EF_RXQ_SIZE 4096
onload_set EF_CLUSTER_IGNORE 1
# enable receive packet event batching, this adds a small latency
# cost, but improves transaction rate/efficiency
onload_set EF_HIGH_THROUGHPUT_MODE 1
# disable CTPIO and PIO as these reduce CPU efficiency and don't
# for this class of application, bring major benefits.
onload_set EF_CTPIO 0
onload_set EF_PIO 0
```

B Installation and configuration

This appendix describes how to install and configure the software distributions used for this benchmarking. See:

- Installing Couchbase on page 22
- Installing YCSB on page 26
- Installing Cloud Onload on page 27.

B.1 Installing Couchbase

This section describes how to install and configure Couchbase.

Installation

NOTE: For a reference description of how to install Couchbase, see *Installing using yum* at https://docs.couchbase.com/server/6.0/install/rhel-suse-install-intro.html.

In summary:

1. Download and install the appropriate meta package from the package download location. See the link above for the latest location. This installation uses `couchbase-release-1.0-1`:

   ```
   # curl –o http://packages.couchbase.com/releases/couchbase-
   release/couchbase-release-1.0-1-x86_64.rpm
   ```

   ```
   # rpm -i couchbase-release-1.0-1-x86_64.rpm
   ```

2. Install the actual Couchbase Server package.

   NOTE: This benchmark testing uses `couchbase-server`, not `couchbase-server-community`.

   ```
   # yum install couchbase-server
   ```

3. Increase the maximum process limits.

   Either modify the `/etc/security/limits.conf` file, or create a new `.conf` file in the `/etc/security/limits.d` directory (such as `91-couchbase.conf`), adding the following lines:

   ```
   * soft nproc 70000
   * hard nproc unlimited
   # End of file
   ```

4. View and confirm the new limit:

   ```
   # ulimit -u
   unlimited
   ```

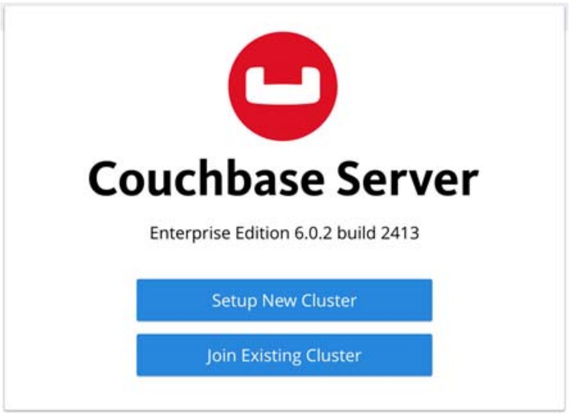
5 Verify the Couchbase Server installation, as described below.

Basic verification

For basic confirmation that the Couchbase Server is available, connect to its Web Console. This is an embedded web server on port 8091. It can be accessed from a web browser either by hostname or by IP address as follows:

- http://<Couchbase server hostname>:8091/
  For example, http://sfocr740b:8091/
- http://<Couchbase server IP address>:8091/
  For example, http://10.20.128.35:8091/

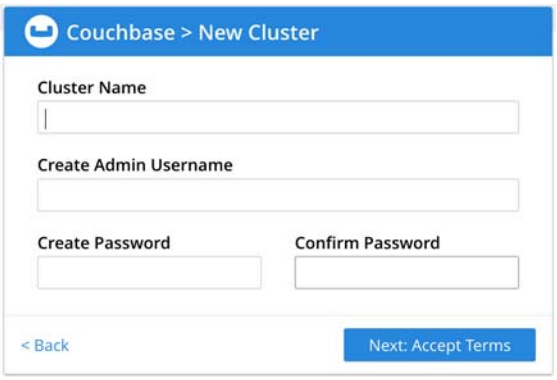
The splash screen appears with options for initial setup:



Initial setup

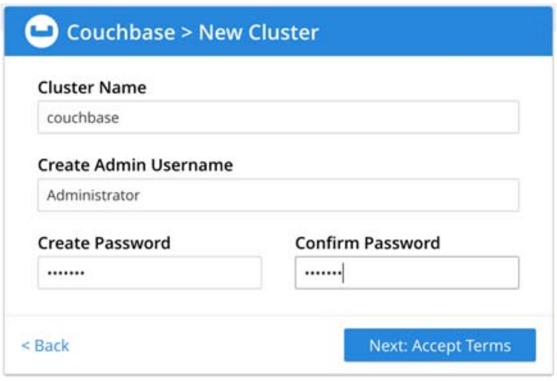
Now perform initial setup of the Couchbase Server software:

1 Click on **Setup New Cluster**. The **New Cluster** dialog appears.



2 Make the following settings:

Setting	Value
Cluster Name	couchbase
Admin Username	Administrator
Password	adminis


Couchbase > New Cluster

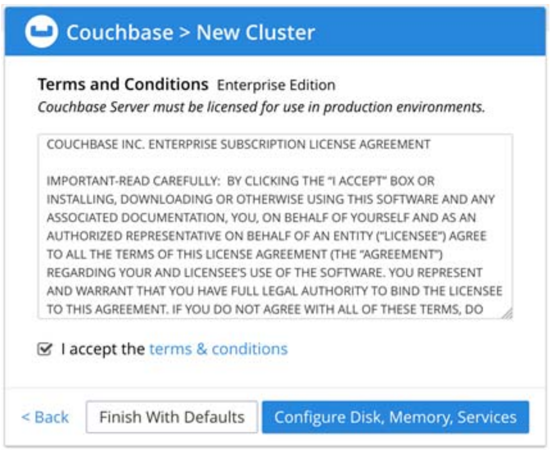
Cluster Name
couchbase

Create Admin Username
Administrator

Create Password
•••••••

Confirm Password
•••••••

3 Click **Next: Accept Terms**. The **Terms and Conditions** dialog appears:


Couchbase > New Cluster

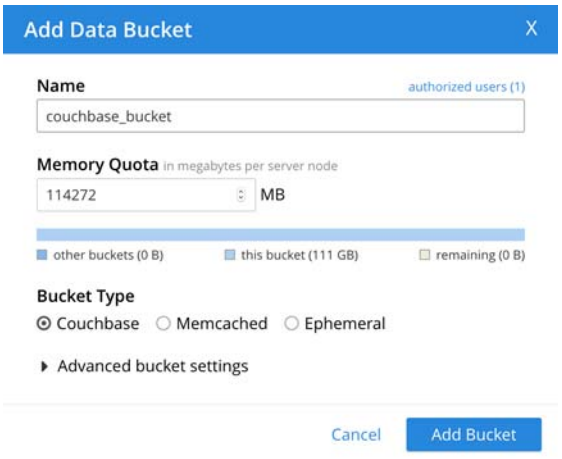
Terms and Conditions Enterprise Edition
Couchbase Server must be licensed for use in production environments.

COUCHBASE INC. ENTERPRISE SUBSCRIPTION LICENSE AGREEMENT

IMPORTANT-READ CAREFULLY: BY CLICKING THE "I ACCEPT" BOX OR INSTALLING, DOWNLOADING OR OTHERWISE USING THIS SOFTWARE AND ANY ASSOCIATED DOCUMENTATION, YOU, ON BEHALF OF YOURSELF AND AS AN AUTHORIZED REPRESENTATIVE ON BEHALF OF AN ENTITY ("LICENSEE") AGREE TO ALL THE TERMS OF THIS LICENSE AGREEMENT (THE "AGREEMENT") REGARDING YOUR AND LICENSEE'S USE OF THE SOFTWARE. YOU REPRESENT AND WARRANT THAT YOU HAVE FULL LEGAL AUTHORITY TO BIND THE LICENSEE TO THIS AGREEMENT. IF YOU DO NOT AGREE WITH ALL OF THESE TERMS, DO

☑ I accept the terms & conditions

4 Click **Finish With Defaults**. The **Add Data Bucket** dialog appears:



Name: couchbase_bucket

Memory Quota in megabytes per server node: 114272 MB

other buckets (0 B) | this bucket (111 GB) | remaining (0 B)

Bucket Type: [✓ Couchbase] Memcached Ephemeral

Advanced bucket settings

5 Ensure the **Name** of the bucket is set to `couchbase_bucket`, and the **Bucket Type** is set to **Couchbase**.

6 Click **Add Bucket**.

Detailed verification

To verify that clients can connect to the node, you can use the `cbworkloadgen` command. This is a Python program that performs basic operations (reads and writes) against the Couchbase Server cluster.

Execute the following command from the Couchbase Server:

```
# /opt/couchbase/bin/cbworkloadgen -u Administrator -p adminis \
      -n <Couchbase hostname or IP address>:8091 -b couchbase_bucket
```

For example:

```
# /opt/couchbase/bin/cbworkloadgen -u Administrator -p adminis \
      -n 10.20.128.35:8091 -b couchbase_bucket
  [####################] 100.0% (10527/estimated 10527 msgs)
bucket: default, msgs transferred...
       :                total |       last |    per sec
 byte  :               105270 |     105270 |   279010.3
done
```

B.2 Installing YCSB

This section describes how to install and configure YCSB.

Installation

NOTE: For a reference description of how to install YCSB, see:
https://github.com/brianfrankcooper/YCSB
https://github.com/brianfrankcooper/YCSB/tree/master/couchbase2.

In summary:

1. Install Maven 3, which is required to build YCSB.

   NOTE: For a reference description of how to install Maven 3, see:
   https://tecadmin.net/install-apache-maven-on-centos/

   To do this:

```
# cd /opt
# wget https://www-eu.apache.org/dist/maven/maven-
3/3.6.0/binaries/apache-maven-3.6.0-bin.tar.gz
# tar -zxvf apache-maven-3.6.0-bin.tar.gz
# ln -s apache-maven-3.6.0 maven
```

2. Setup the Maven 3 environment variables:

```
# cat /etc/profile.d/maven.sh
export M2_HOME=/opt/maven
export PATH=${M2_HOME}/bin:${PATH}

# source /etc/profile.d/maven.sh
```

3. Confirm Maven 3 is installed:

```
# mvn -version
Apache Maven 3.6.0
```

4. Obtain the YCSB.git package:

```
# mkdir YCSB_repo
# cd YCSB_repo
# git clone git://github.com/brianfrankcooper/YCSB.git
```

5. Build YCSB using Maven 3:

```
# cd YCSB
# mvn clean package
...
[INFO] Building tar:
/root/Onload_Testing/Couchbase/YCSB_repo/YCSB/distribution/target/ycsb
-0.17.0-SNAPSHOT.tar.gz
...
[INFO] ---------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ---------------------------------------------------------------
[INFO] Total time:  13:29 min
[INFO] Finished at: 2019-06-20T01:13:29-07:00
[INFO] ---------------------------------------------------------------
```

B.3 Installing Cloud Onload

For instructions on how to install and configure Cloud Onload, refer to the *Onload User Guide* (SF-104474-CD). This is available from https://support.solarflare.com/.