Отчет о проведении тестирования.

Время проведения: *30.05.2020*

1. Цель теста

Бизнес-цель теста – *получение навыков работы с репликацией БД и нагрузочного тестирования.*

Техническая цель теста – оценка производительности с использованием *wrk* и *СУБД MySQL 5.7.30* на кластере *Kubernetes 1.14.10-gke.36*.

В испытании участвовало 4 операций.

Таблица 1. Модель нагрузки.

1	*Нагрузка на систему с одним инстансом БД MySQL.*
2	*Нагрузка на систему с двумя инстансами (master + slave) БД MySQL.*

Было произведено 8 экспериментов с разным количеством запросов в секунду.

Таблица 2. Эксперименты.

1	*1 соединение с одним инстансом БД MySQL.*
2	*10 соединений с одним инстансом БД MySQL.*
3	*100 соединений с одним инстансом БД MySQL.*
4	*1000 соединений с одним инстансом БД MySQL.*
5	*1 соединение с двумя инстансами БД MySQL.*
6	*10 соединений с двумя инстансами БД MySQL.*
7	*100 соединений с двумя инстансами БД MySQL.*
8	*1000 соединений с двумя инстансами БД MySQL.*

2. Выводы

1. Максимальная производительность системы – *140* запроса в секунду при 8-ми потоковом тесте и *100* соединениях.
2. Время *отклика / выполнения* на уровне максимальной производительности по операциям составляло от *470* миллисекунд до *2,6* секунды. Так же были просадки после *90 перцентиля* в диапазоне *12* секунд.
3. Нагрузка при *1-м* соединении и *10-ти* соединениях практически не отличалась, за исключением “прогревания кеша”. При нагрузке в *100* соединениях, распараллеливание запросов позволило в *2* раза сократить latency(отзывчивость) системы. А также увеличить количество запросов в секунду на *100* процентов.

Настройки, создание кластера master/slave

4. Для настройки репликации *master/slave* использовались возможности кластера *Kubernetes*.
5. Развертывание и клонирование БД осуществлялось контейнерами *init-mysql* и *clone-mysql* с использованием программного продукта *xtrabackup*.
6. Репликация так же настраивалась с помощью вспомогательного контейнера *xtrabackup* и конфигурационного скрипта:

```
if [[ -f change_master_to.sql.in ]]; then
 echo "Waiting for mysqld to be ready (accepting connections)"
 until mysql -h 127.0.0.1 -e "SELECT 1"; do sleep 1; done

 echo "Initializing replication from clone position"
 mysql -h 127.0.0.1 \
       -e "$(<change_master_to.sql.in), \
             MASTER_HOST='mysql-0.mysql.default.svc.cluster.local', \
             MASTER_USER='root', \
             MASTER_PASSWORD='', \
             MASTER_CONNECT_RETRY=10; \
             START SLAVE;" || exit 1
 # In case of container restart, attempt this at-most-once.
 mv change_master_to.sql.in change_master_to.sql.orig
fi
```

7. Файл описания *StatefulSet-a*: *mysql-statefulset.yaml*,
   файл описания сервисов: *mysql-services.yaml*,
   файл описания дополнительной конфигурации: *mysql-configmap.yaml*.
8. Балансировка нагрузки на чтение, также осуществлялась при помощи кластера *Kubernetes*. С использованием сервиса *Kubernetes*. Файл описания *mysql-reads.loadBalancer.yaml*.
9. При проведении нагрузочного тестирования наиболее ресурсоемкие запросы:

```
SELECT id, first_name, sur_name, age, sex, city
  FROM user_info
 WHERE first_name LIKE ? AND sur_name LIKE ?
```

и

```
SELECT id, user_info_id, interest FROM user_interest
 WHERE user_info_id IN (?, ?, ?, ...)
```

3. Графики

3.1. График latency и Throughput.

График 1. latency (ms).

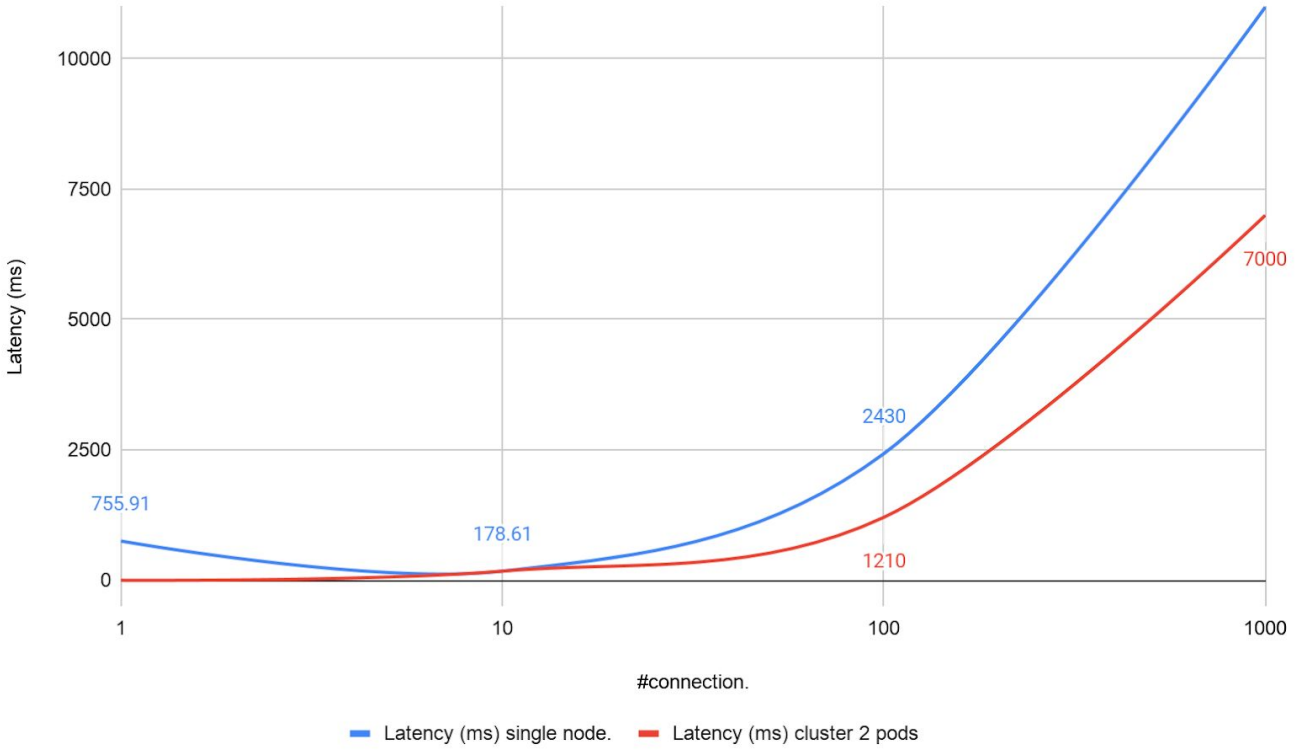
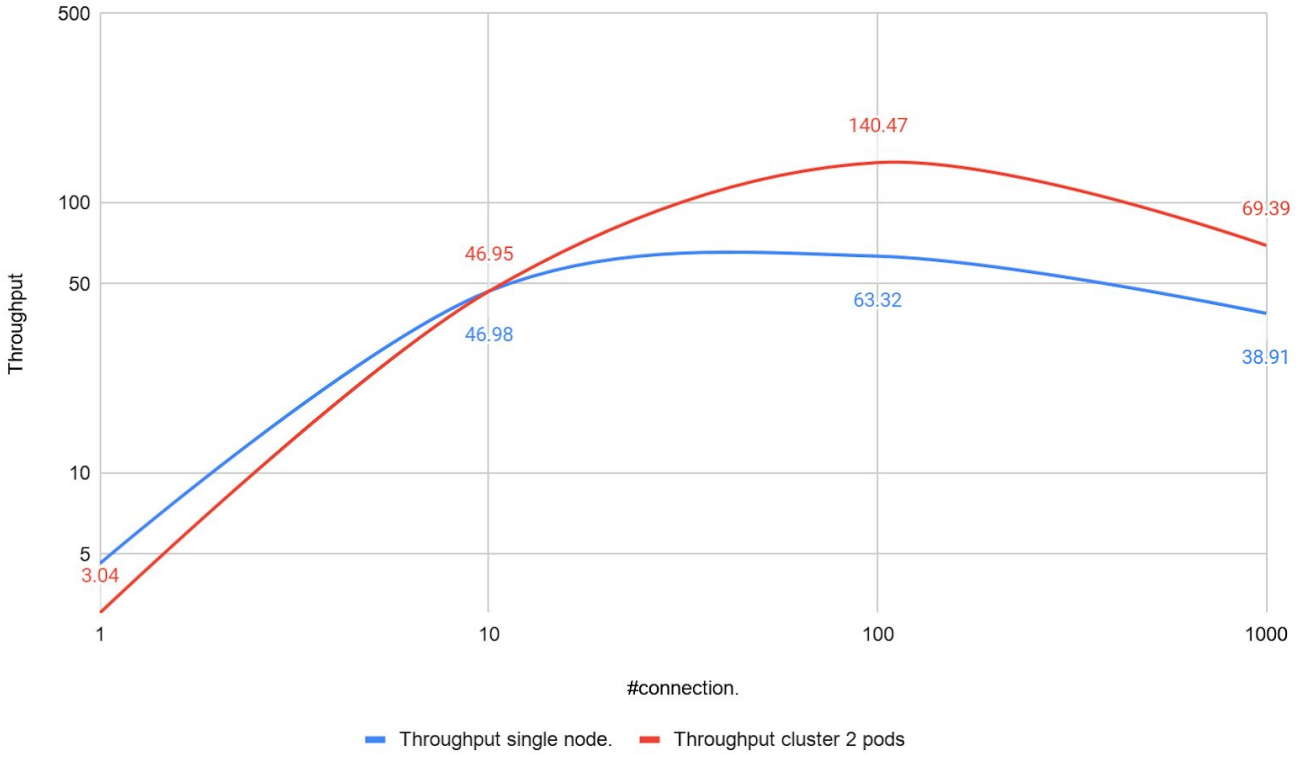


График 2. Throughput.



3.2. Нагрузка CPU.

График 3. Нагрузка CPU.

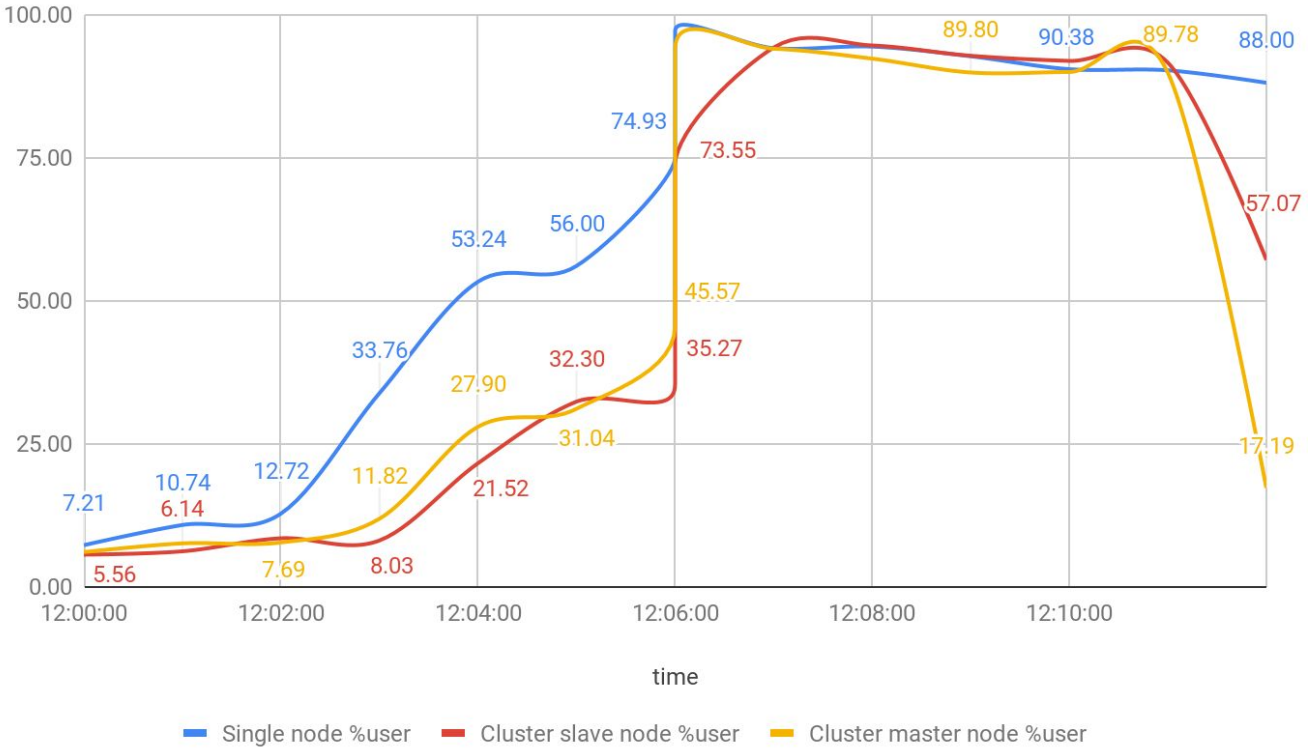
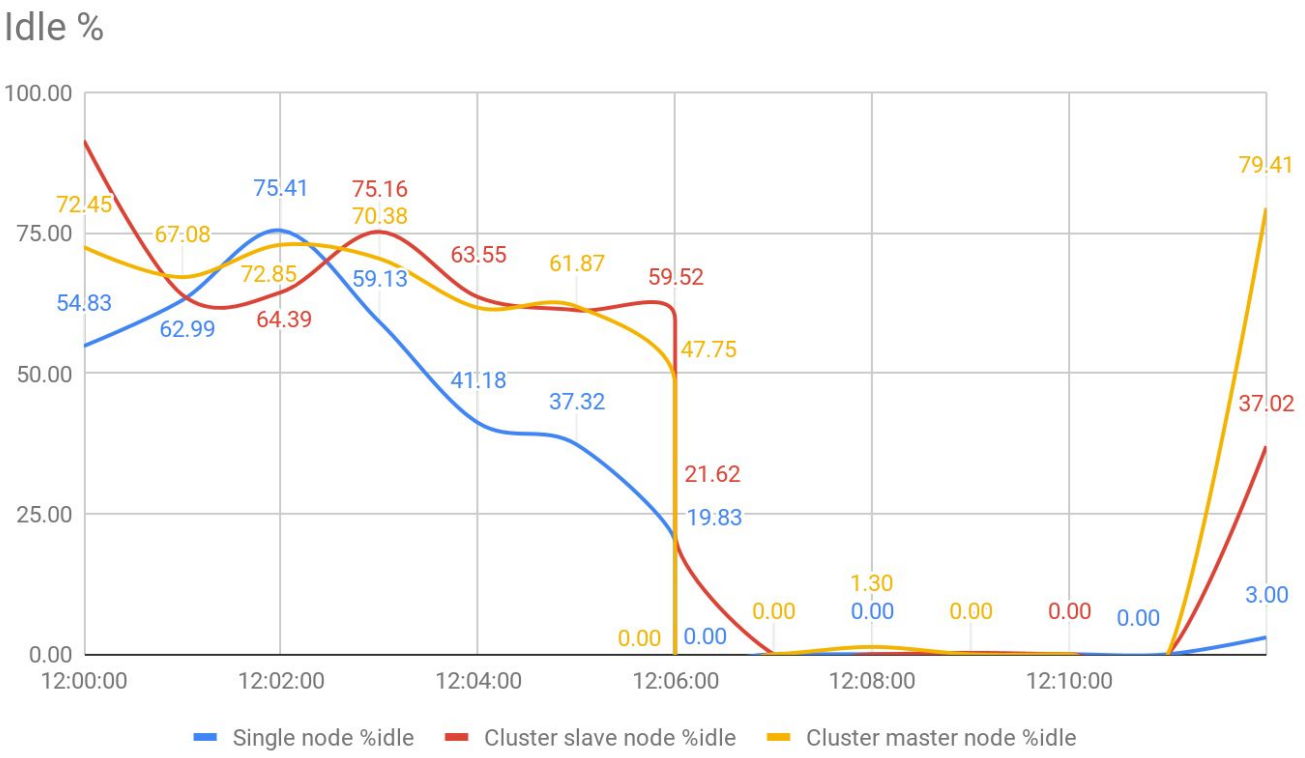


График 4. Нагрузка CPU idle.



3.3. Load averages.

График 5. График LA. Single node.

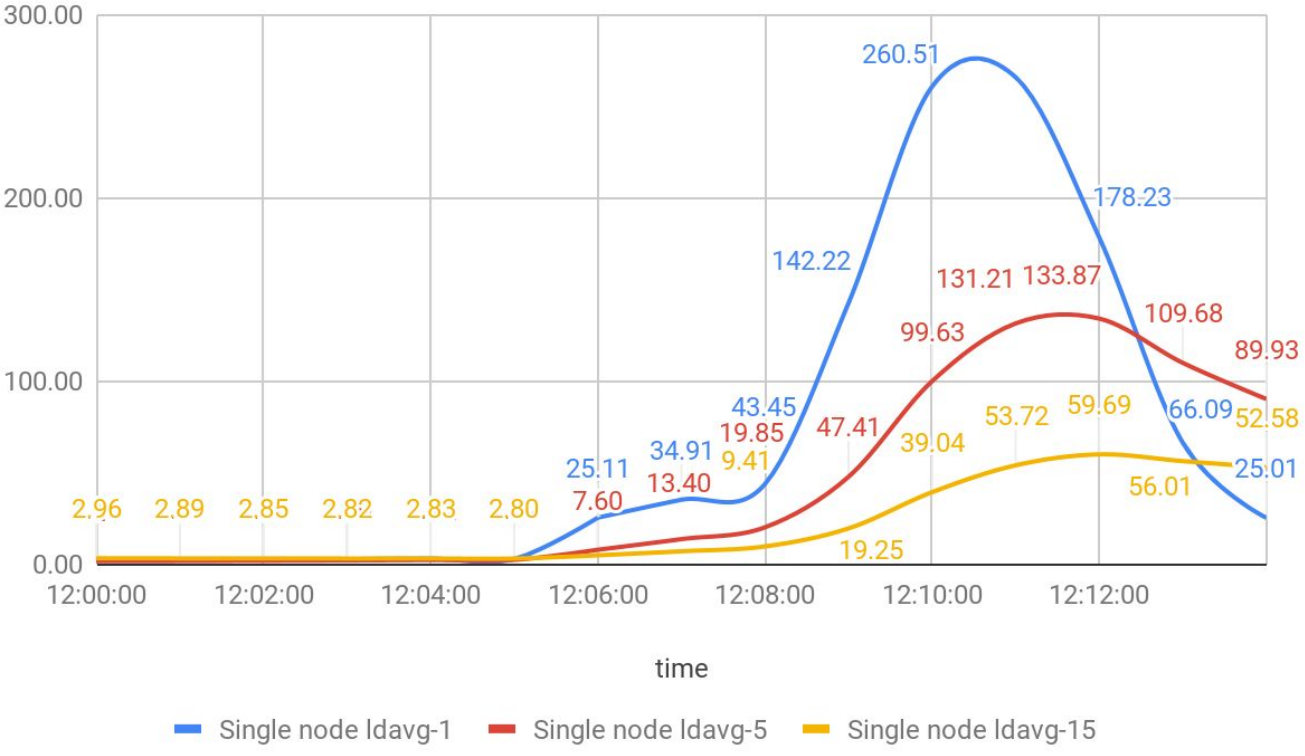


График 6. График LA. Cluster 2 pods master node.

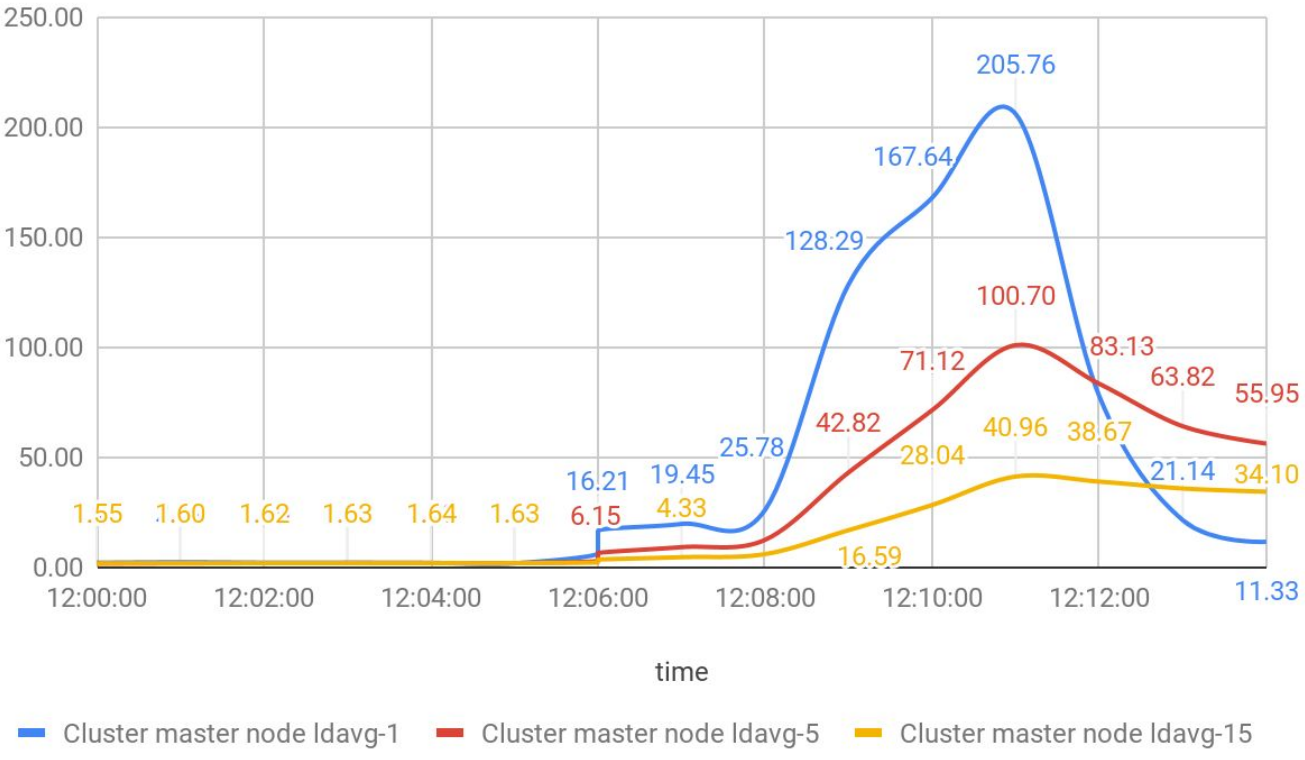
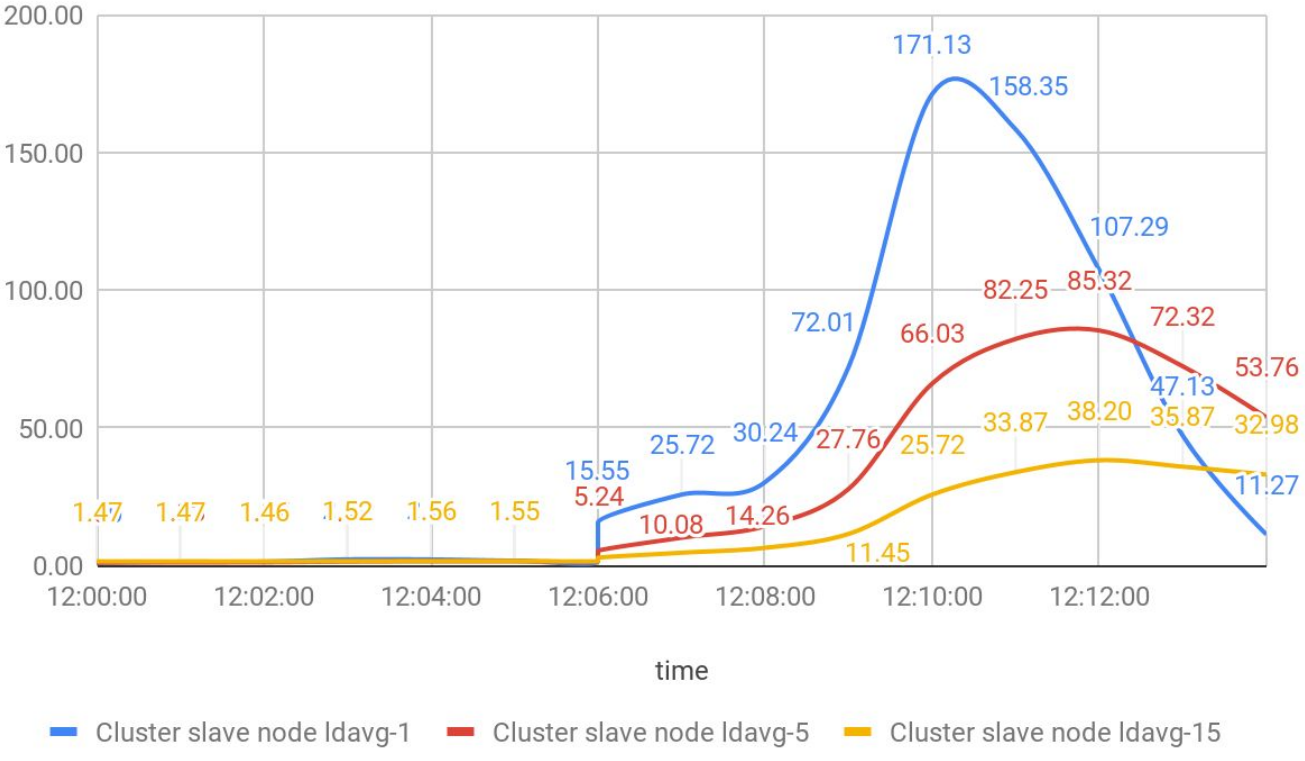


График 7. График LA. Cluster 2 pods slave node.



3.5. I/O - disc usage.

График 8. Количество дисковых операции в секунду и поток чтения. Single node.

График 9. Количество дисковых операции в секунду и поток чтения. Cluster 2 pods master node.

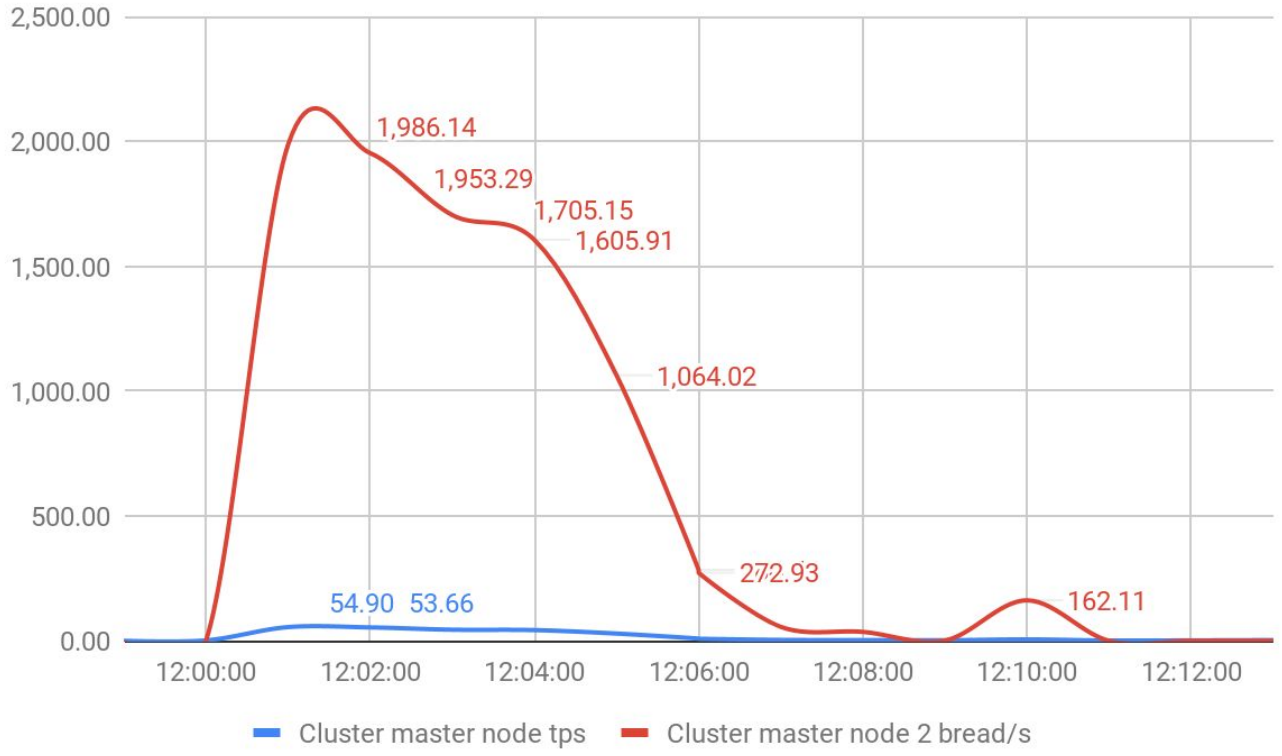


График 10. Количество дисковых операции в секунду и поток чтения. Cluster 2 pods slave node.