Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

«Информатика»

кафедра

Отчёт о научно-исследовательской работе

Институт космических и информационных технологий

место прохождения практики

Сравнение скорости CRUD операций (MongoDB c MySQL)

тема

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель |  |  | Д.В. Грузенкин |
|  |  | подпись, дата | инициалы, фамилия |
| Студент КИ21-17/2Б, 032156940 |  |  | Н.А. Самарин |
| номер группы, зачетной книжки |  | подпись, дата | инициалы, фамилия |

Красноярск 2023

**СОДЕРЖАНИЕ**

1 Индивидуальное задание................................................................................. 3

2 Контрольный график (план) практики........................................................... 3

3 Введение............................................................................................................ 4

4 Теоретическая часть......................................................................................... 4

4.1 Принципы работы хранилищ данных........................................................ 4

4.1.1 Принципы работы MySQL..................................................................... 4

4.1.2 Принципы работы MongoDB................................................................. 5

4.2 CRUD операции в MongoDB и MySQL..................................................... 5

5 Экспериментальная часть................................................................................ 6

5.1 Подготовка к экспериментам..................................................................... 6

5.2 Выполнение и результаты экспериментов................................................ 6

6 Вывод................................................................................................................. 8

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ............................................ 9

ПРИЛОЖЕНИЕ А................................................................................................. 10

**1 Индивидуальное задание**

Сравнить скорости CRUD операций хранилищ данных MongoDB и  
MySQL.

Теоретическая часть:

- Описать что такое CRUD операции;

- Описать как работают хранилища данных, ссылаясь на соответствующую документацию;

- Найти информацию о том, как и почему скорость CRUD операций хранилищ отличается.

Экспериментальная часть:

- Установить docker toolbox (или более свежее решение);

- Скачать контейнеры с соответствующими базами данных;

- Написать два простых скрипта выполняющих CRUD операции для каждой из пары баз данных и измеряющих время выполнения;

- Каждый эксперимент провести несколько раз, при этом:

- Нужно указать параметры (виртуальной) машины, на которой проводились исследования (кол-во RAM, CPU, потоков);

- Указать количество итераций для каждого эксперимента;

- Привести значения математического ожидания и дисперсии для каждого  
результата;

- Сделать графики с пояснениями;

- Сделать выводы о том, почему в данных хранилищах имеются различия  
в выполнении CRUD операций, чем это вызвано и как дизайн системы влияет  
на данный параметр.

**2 Контрольный график (план) практики**

Таблица 1 – График практики

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Разделы (этапы) практики | Виды учебной работы, на практике включая самостоятельную работу обучающихся и трудоемкость (в часах) | Формы контроля |
| 1 | Подготовительный этап | Инструктаж на рабочем месте по технике безопасности, противопожарной безопасности, 2 часа | По фактическому прохождению |
|  |  | Ознакомление практику с заданием на, 2 часа | Общий контроль руководителя |
|  |  | Самостоятельное изучение используемого оборудования и программного обеспечения, 2 часа | Общий контроль руководителя |

Окончание таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Разделы (этапы) практики | Виды учебной работы, на практике включая самостоятельную работу обучающихся и трудоемкость (в часах) | Формы контроля |
| 2 | Экспериментальный этап | Сбор и анализ материала, анализ литературы по предметной области, 16 часов | Глава отчета по практике |
|  |  | Выполнение исследований по теме задания на практику и разработка программного продукта по теме исследований, 76 часов | Общий контроль руководителя |
| 3 | Составление отчета по научно-исследовательской работе и его защита | Подготовка и оформление отчета по практике, 8 часов | Отчет по практике |
|  |  | Подготовка к защите и публичная защита отчета, 2 часа | Зачет с оц. |

**3 Введение**

CRUD — акроним, обозначающий четыре базовые функции,  
используемые при работе с базами данных: создание (create), чтение (read),  
модификация (update), удаление (delete). Введён Джеймсом Мартином в 1983  
году как стандартная классификация функций по манипуляции данными [1].

**4 Теоретическая часть**

**4.1 Принципы работы хранилищ данных**

**4.1.1 Принципы работы MySQL**

Реляционная система управления базами данных MySQL — это  
организованная в виде таблиц открытая система, поддерживающая  
универсальные языки программирования SQL и позволяющая пользователям  
находить, редактировать или удалять хранившуюся в базе данных информацию  
[2].

Работая по модели «клиент/сервер», MySQL осуществляет обмен между  
пользователем и хранилищем данных. Поиск и выдача нужной информации  
происходит по принципу множества и записи, где множества — это  
наименования таблицы, а запись — строка в ней. Это означает, что система  
автоматически подбирает наиболее подходящий под запрос оператора  
результат, исходя из имеющихся в таблицах данных. Все таблицы связаны  
между собой и образуют циркулярную связь [3].

**4.1.2 Принципы работы MongoDB**

MongoDB - система управления базами данных с открытым исходным  
кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL,  
использует JSON-подобные документы и схему базы данных. Если  
реляционные базы данных хранят строки, то MongoDB хранит документы. В  
отличие от строк документы могут хранить сложную по структуре  
информацию. Документ можно представить как хранилище ключей и  
значений. Каждому ключу сопоставляется определенное значение. Но если в  
реляционных базах есть четко очерченная структура, где есть поля, и если  
какое-то поле не имеет значение, ему можно присвоить значение NULL. В  
MongoDB если какому-то ключу не сопоставлено значение, то этот ключ  
просто опускается в документе и не употребляется. В отличие от реляционных  
баз данных вместо таблиц, в MongoDB используются коллекции. И если в  
реляционных БД таблицы хранят однотипные жестко структурированные  
объекты, то в коллекции могут содержать самые разные объекты, имеющие  
различную структуру и различный набор свойств [4].

Вся система MongoDB может представлять не только одну базу данных,  
которая располагается на одном физическом сервере. Функциональность  
MongoDB позволяет расположить несколько баз данных на нескольких  
физических серверах, и эти базы данных смогут легко обмениваться данными и  
сохранять целостность. Система хранения данных в MongoDB представляет  
набор реплик. В этом наборе есть основной узел, а также может быть набор  
вторичных узлов. Все вторичные узлы сохраняют целостность и автоматически  
обновляются вместе с обновлением главного узла. И если основной узел по  
каким-то причинам выходит из строя, то один из вторичных узлов становится  
главным [5].

**4.2 CRUD операции в MongoDB и MySQL**

В обеих этих базах данных мы можем выполнять операции CRUD  
(создание, чтение, обновление, удаление), но их синтаксис отличается.

Таблица 2 – CRUD операции в MongoDB и MySQL

|  |  |  |
| --- | --- | --- |
| CRUD | MongoDB | MySQL |
| Create | db.collection.insertOne() db.collection.insertMany() | INSERT INTO <TABLE> |
| Read | db.collection.find() | SELECT \* FROM <TABLE> |
| Update | db.collection.updateOne() db.collection.updateMany() | UPDATE <TABLE> |
| Delete | db.collection.deleteOne() db.collection.deleteMany() | DELETE FROM <TABLE> |

Основные различия в механике работы CRUD операций основаны на  
различиях в архитектуре хранения данных в MySQL и MongoDB. Большая  
структурированность и строгость таблиц MySQL по сравнению с коллекциями

MongoDB, требует больше проверок на корректность данных при добавлении  
записей в базу, но при этом обеспечивает более быстрое индексирование при  
поиске записей для отображения, обновления и удаления.

**5 Экспериментальная часть**

**5.1 Подготовка к экспериментам**

Для начала были установлены пакеты разработки для создания и  
управления серверами MongoDB и MySQL. Как среда для проведения  
экспериментов с СУБД использовался Docker - программное обеспечение для  
автоматизации развёртывания и управления приложениями в средах с  
поддержкой контейнеризации. После запуска серверов баз данных в Docker  
Desktop были созданы и подготовлены к работе контейнеры, подключённые к  
серверам MongoDB и MySQL соответственно.

На языке программирования Python, с применением требуемых  
библиотек, были написаны программы, осуществляющие подключение к базам  
данных, выполнение CRUD-операций и замер времени их выполнения. Полный  
код программ указан в приложении А.

Для создания существенного объёма данных для обработки СУБД,  
эксперименты было решено проводить на датасете с 100000 записей,  
содержащих по 9 полей текстовой и числовой информации об организациях [8].  
Данные датасета были импортированы в обе базы данных.

Параметры машины, на которой выполнялись эксперименты: процессор  
Ryzen 7 4700U 2.00 GHz, 8 потоков, 16 Gb RAM, SSD.

**5.2 Выполнение и результаты экспериментов**

Были проведены следующе эксперименты: в каждой базе данных были  
поочерёдно были выполнены CRUD операции, с использованием 100, 500 и  
5000 записей. Для каждого эксперимента было выполнено 1000 итераций, в  
каждой из которых было замерено время выполнения каждой CRUD операции.  
После этого было вычислено среднее значение(математическое ожидание) и  
дисперсия времени выполнения CRUD операций.

Результаты экспериментов представлены в таблицах и на диаграммах  
ниже.

Таблица 3 – Среднее время для 100 записей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СУБД | Среднее Create, с | Среднее Read, с | Среднее Update, с | Среднее Delete, с |
| MongoDB | 0.00241 | 0.06393 | 0.074582 | 0.07116 |
| MySQL | 0.00393 | 0.00191 | 0.00233 | 0.00129 |

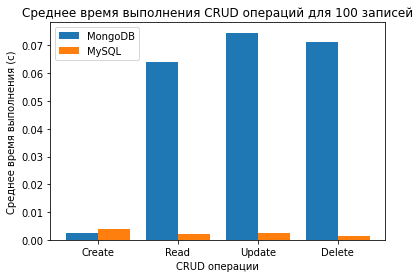


Рисунок 1 – Диаграмма времени для 100 записей

Таблица 4 – Среднее время для 500 записей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СУБД | Среднее Create, с | Среднее Read, с | Среднее Update, с | Среднее Delete, с |
| MongoDB | 0.00697 | 0.0641 | 0.08127 | 0.07639 |
| MySQL | 0.01947 | 0.00685 | 0.01526 | 0.00527 |

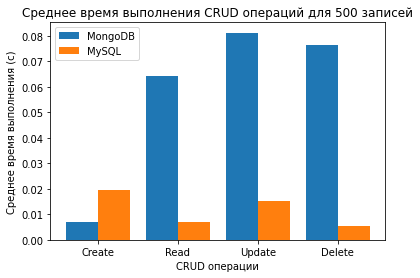


Рисунок 2 – Диаграмма времени для 500 записей

Таблица 5 – Среднее время для 5000 записей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СУБД | Среднее Create, с | Среднее Read, с | Среднее Update, с | Среднее Delete, с |
| MongoDB | 0.05633 | 0.06787 | 0.15848 | 0.12958 |
| MySQL | 0.20564 | 0.06442 | 0.1481 | 0.04925 |

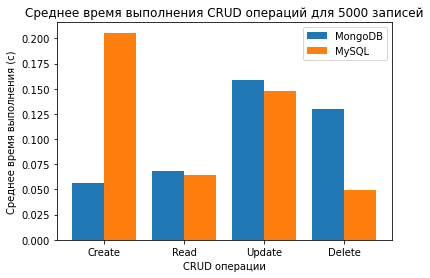


Рисунок 3 – Диаграмма времени для 5000 записей

Таблица 6 – Дисперсия времени выполнения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СУБД | Число записей | Дисперсия Create, с | Дисперсия Read, с | Дисперсия Update, с | Дисперсия Delete, с |
| MongoDB | 100 | 0.000005 | 0.000005 | 0.000005 | 0.000004 |
|  | 500 | 0.000005 | 0.000004 | 0.000025 | 0.000008 |
|  | 5000 | 0.000196 | 0.000006 | 0.000063 | 0.000041 |
| MySQL | 100 | 0.0000006 | 0.0000003 | 0.0000004 | 0.0000002 |
|  | 500 | 0.000007 | 0.0000006 | 0.000002 | 0.000001 |
|  | 5000 | 0.003421 | 0.00006 | 0.00006 | 0.00008 |

**6 Вывод**

Реляционная СУБД MySQL существенно выигрывает у  
документоориентированной СУБД MongoDB во времени выполнения всех  
CRUD операций за исключением Create, однако это опережение начинает  
сильно сокращаться по мере увеличения количества обрабатываемых за раз  
записей. Такие различия полностью объясняются архитектурами этих СУБД,  
документоориентированная MongoDB, записи в коллекциях которой не  
обязаны следовать жёсткой структуре, не выполняет столько же проверок на  
соответствие записи структуре таблицы как это делает MySQL, что приводит к  
высокому быстродействию операции Create относительно MySQL. Однако, по  
этой же причине, CRUD операции с поиском некоторых записей в коллекции  
(Read, Update, Delete) происходят медленнее чем в MySQL, в которой  
быстродействие этих операций обусловлено более эффективным  
индексированием, относительно MongoDB, благодаря строгой типизации  
столбцов таблиц.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 James, Martin Managing the Data-base Environment / James Martin. –  
Englewood Cliffs : Prentice-Hall, 1983. – 766с.

2 General Information | MySQL Documentation [Электронный ресурс] –  
URL: https://dev.mysql.com/doc/refman/8.0/en/introduction.html (Дата обращения:  
02.07.2023).

3 Азбука MySQL: практика, особенности, принципы  
функционирования [Электронный ресурс] – URL:  
https://nangs.org/news/it/azbuka-mysql-praktika-osobennosti-printsipy-funktsionirov  
aniya (Дата обращения: 02.07.2023).

4 Introduction to MongoDB | MongoDB Documentation [Электронный  
ресурс] – URL: https://www.mongodb.com/docs/manual/ (Дата обращения:  
02.07.2023).

5 Replication | MongoDB Documentation [Электронный ресурс] – URL:  
https://www.mongodb.com/docs/manual/replication/ (Дата обращения: 02.07.2023).

6 CRUD Operations Overview | MySQL Documentation [Электронный  
ресурс] – URL:  
https://dev.mysql.com/doc/x-devapi-userguide/en/crud-operations-overview.html  
(Дата обращения: 02.07.2023).

7 MongoDB CRUD Operations | MongoDB Documentation  
[Электронный ресурс] – URL: https://www.mongodb.com/docs/manual/crud/ (Дата  
обращения: 02.07.2023).

8 Datablist [Электронный ресурс] – URL:  
https://www.datablist.com/learn/csv/download-sample-csv-files (Дата обращения:  
07.07.2023).

**ПРИЛОЖЕНИЕ А**

Код python программ, написанных в ходе работы представлен в листингах  
ниже.

Листинг А.1 – Работа с MongoDB

import time  
import numpy  
def insert\_many\_doc(collection, data):  
 return collection.insert\_many(data)  
def find\_document(collection, elements, multiple=False):  
 if multiple:  
 results = collection.find(elements)  
 return [r for r in results]  
 else:  
 return collection.find\_one(elements)  
def update\_document(collection, query\_elements, new\_values):  
 collection.update\_one(query\_elements, {'$set': new\_values})  
def insert\_benchmark(collection, data):  
 t1 = time.time()  
 insert\_many\_doc(collection, data)  
 return time.time() - t1  
def delete\_benchmark(collection):  
 t1 = time.time()  
 collection.delete\_many({"Index": "example"})  
 return time.time() - t1  
def find\_benchmark(collection):  
 return collection.find({"Index": "example"}).explain()["executionStats"]['ex  
 ecutionTimeMillis'] / 1000  
def update\_benchmark(collection):  
 new\_values = {"Description": "updated\_example"}  
 t1 = time.time()  
 collection.update\_many({"Index": "example"}, {'$set': new\_values})  
 return time.time() - t1  
def run\_benchmarks(collection, doc\_num, iter\_num):  
 insert\_times = []  
 find\_times = []  
 update\_times = []  
 delete\_times = []  
 for i in range(0, iter\_num):  
 data = []  
 for j in range(0, doc\_num):  
 doc = {  
 "Index": "example",  
 "Organization Id": "example",  
 "Name": i \* j,  
 "Website": "example",  
 "Country": "example",  
 "Description": "example",  
 "Founded": "example",  
 "Industry": "example",  
 "Number of employees": i \* j  
 }  
 data.append(doc)  
 run\_time = insert\_benchmark(collection, data)  
 insert\_times.append(run\_time)  
 run\_time = find\_benchmark(collection)  
 find\_times.append(run\_time)

Окончание листинга А.1

run\_time = update\_benchmark(collection)  
 update\_times.append(run\_time)  
 run\_time = delete\_benchmark(collection)  
 delete\_times.append(run\_time)  
 return insert\_times, find\_times, update\_times, delete\_times

Листинг А.2 – Работа с MySQL

import pymysql  
import time  
import numpy  
def insert\_benchmark(cursor, data):  
 t1 = time.time()  
 cursor.executemany("INSERT INTO organizations (id, Organization\_Id, Name, We  
 bsite, Country, Description, Founded, Industry, Number\_of\_employees) VALUES   
 (%s, %s, %s, %s, %s, %s, %s, %s, %s)", data)  
 return time.time() - t1  
def delete\_benchmark(cursor):  
 t1 = time.time()  
 cursor.execute("DELETE FROM organizations WHERE Name = 'example'")  
 return time.time() - t1  
def select\_benchmark(cursor):  
 t1 = time.time()  
 cursor.execute("SELECT \* FROM organizations WHERE Name = 'example'")  
 return time.time() - t1  
def update\_benchmark(cursor):  
 t1 = time.time()  
 cursor.execute("UPDATE organizations SET Description='updated\_example' WHERE  
 Name = 'example'")  
 return time.time() - t1  
def run\_benchmarks(cursor, doc\_num, iter\_num):  
 insert\_times = []  
 find\_times = []  
 update\_times = []  
 delete\_times = []  
 for i in range(0, iter\_num):  
 con = pymysql.connect(host='localhost', user='root', password='5x6tpan1'  
 , database='organizations\_schema')  
 cursor = con.cursor()  
 data = []  
 for j in range(0, doc\_num):  
 doc = [i \* doc\_num + j,"example","example","example","example","exam  
 ple",i \* j,"example",i \* j]  
 data.append(doc)  
 run\_time = insert\_benchmark(cursor, data)  
 insert\_times.append(run\_time)  
 run\_time = select\_benchmark(cursor)  
 find\_times.append(run\_time)  
 run\_time = update\_benchmark(cursor)  
 update\_times.append(run\_time)  
 run\_time = delete\_benchmark(cursor)  
 delete\_times.append(run\_time)  
 return insert\_times, find\_times, update\_times, delete\_times

Листинг А.3 – Подключение к базам данных, запуск экспериментов и расчёт  
среднего времени и дисперсии

import pymysql  
from pymongo import \*

Продолжение листинга А.3

import urllib.parse  
import numpy  
import mongo\_work  
import mysql\_work  
host = "localhost"  
port = 27017  
user\_name = "myUserAdmin"  
pass\_word = "abc123"  
db\_name = "admin"  
client = MongoClient(f'mongodb://{user\_name}:{urllib.parse.quote\_plus(pass\_word)  
}@{host}:{port}/{db\_name}')  
db = client['organizations\_db']  
collection = db['organizations\_col']  
print(collection.find())  
con = pymysql.connect(host='localhost', user='root', password='5x6tpan1', databa  
se='organizations\_schema')  
cur = con.cursor()  
#%%  
insert\_mongodb\_100, find\_mongodb\_100, update\_mongodb\_100, delete\_mongodb\_100 = m  
ongo\_work.run\_benchmarks(collection, 100, 1000)  
insert\_mongodb\_100\_mean = numpy.average(insert\_mongodb\_100)  
insert\_mongodb\_100\_var = numpy.var(insert\_mongodb\_100)  
find\_mongodb\_100\_mean = numpy.average(find\_mongodb\_100)  
find\_mongodb\_100\_var = numpy.var(find\_mongodb\_100)  
update\_mongodb\_100\_mean = numpy.average(update\_mongodb\_100)  
update\_mongodb\_100\_var = numpy.var(update\_mongodb\_100)  
delete\_mongodb\_100\_mean = numpy.average(delete\_mongodb\_100)  
delete\_mongodb\_100\_var = numpy.var(delete\_mongodb\_100)  
#%%  
insert\_mongodb\_500, find\_mongodb\_500, update\_mongodb\_500, delete\_mongodb\_500 = m  
ongo\_work.run\_benchmarks(collection, 500, 1000)  
insert\_mongodb\_500\_mean = numpy.average(insert\_mongodb\_500)  
insert\_mongodb\_500\_var = numpy.var(insert\_mongodb\_500)  
find\_mongodb\_500\_mean = numpy.average(find\_mongodb\_500)  
find\_mongodb\_500\_var = numpy.var(find\_mongodb\_500)  
update\_mongodb\_500\_mean = numpy.average(update\_mongodb\_500)  
update\_mongodb\_500\_var = numpy.var(update\_mongodb\_500)  
delete\_mongodb\_500\_mean = numpy.average(delete\_mongodb\_500)  
delete\_mongodb\_500\_var = numpy.var(delete\_mongodb\_500)  
#%%  
insert\_mongodb\_5000, find\_mongodb\_5000, update\_mongodb\_5000, delete\_mongodb\_5000  
 = mongo\_work.run\_benchmarks(collection, 5000, 1000)  
insert\_mongodb\_5000\_mean = numpy.average(insert\_mongodb\_5000)  
insert\_mongodb\_5000\_var = numpy.var(insert\_mongodb\_5000)  
find\_mongodb\_5000\_mean = numpy.average(find\_mongodb\_5000)  
find\_mongodb\_5000\_var = numpy.var(find\_mongodb\_5000)  
update\_mongodb\_5000\_mean = numpy.average(update\_mongodb\_5000)  
update\_mongodb\_5000\_var = numpy.var(update\_mongodb\_5000)  
delete\_mongodb\_5000\_mean = numpy.average(delete\_mongodb\_5000)  
delete\_mongodb\_5000\_var = numpy.var(delete\_mongodb\_5000)  
#%%  
insert\_mysql\_100, find\_mysql\_100, update\_mysql\_100, delete\_mysql\_100 = mysql\_wor  
k.run\_benchmarks(cur, 100, 1000)  
insert\_mysql\_100\_mean = numpy.average(insert\_mysql\_100)  
insert\_mysql\_100\_var = numpy.var(insert\_mysql\_100)  
find\_mysql\_100\_mean = numpy.average(find\_mysql\_100)  
find\_mysql\_100\_var = numpy.var(find\_mysql\_100)

Окончание листинга А.3

update\_mysql\_100\_mean = numpy.average(update\_mysql\_100)  
update\_mysql\_100\_var = numpy.var(update\_mysql\_100)  
delete\_mysql\_100\_mean = numpy.average(delete\_mysql\_100)  
delete\_mysql\_100\_var = numpy.var(delete\_mysql\_100)  
#%%  
insert\_mysql\_500, find\_mysql\_500, update\_mysql\_500, delete\_mysql\_500 = mysql\_wor  
k.run\_benchmarks(cur, 500, 1000)  
insert\_mysql\_500\_mean = numpy.average(insert\_mysql\_500)  
insert\_mysql\_500\_var = numpy.var(insert\_mysql\_500)  
find\_mysql\_500\_mean = numpy.average(find\_mysql\_500)  
find\_mysql\_500\_var = numpy.var(find\_mysql\_500)  
update\_mysql\_500\_mean = numpy.average(update\_mysql\_500)  
update\_mysql\_500\_var = numpy.var(update\_mysql\_500)  
delete\_mysql\_500\_mean = numpy.average(delete\_mysql\_500)  
delete\_mysql\_500\_var = numpy.var(delete\_mysql\_500)  
#%%  
insert\_mysql\_5000, find\_mysql\_5000, update\_mysql\_5000, delete\_mysql\_5000 = mysql  
\_work.run\_benchmarks(cur, 5000, 1000)  
insert\_mysql\_5000\_mean = numpy.average(insert\_mysql\_5000)  
insert\_mysql\_5000\_var = numpy.var(insert\_mysql\_5000)  
find\_mysql\_5000\_mean = numpy.average(find\_mysql\_5000)  
find\_mysql\_5000\_var = numpy.var(find\_mysql\_5000)  
update\_mysql\_5000\_mean = numpy.average(update\_mysql\_5000)  
update\_mysql\_5000\_var = numpy.var(update\_mysql\_5000)  
delete\_mysql\_5000\_mean = numpy.average(delete\_mysql\_5000)  
delete\_mysql\_5000\_var = numpy.var(delete\_mysql\_5000)