Configuration Manual

Benchmarking the Performance of Java Virtual Threads in High-Throughput Workloads
Cloud Computing

Vishesh Pandita
Student ID: x23184531

School of Computing
National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vishesh Pandita
Student ID:	x23184531
Programme:	Cloud Computing
Year:	2024
Module:	Benchmarking the Performance of Java Virtual Threads in High-Throughput Workloads
Supervisor:	Yasantha Samarawickrama
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	1274
Page Count:	6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vishesh Pandita
Date:	12th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vishesh Pandita
x23184531

1 Introduction

This configuration manual describes in details the steps to implement and deploy the benchmarking infrastructure to benchmark Java Virtual Threads in a safe and robust environment that was proposed in the research.
This section provides a detailed description of the steps needed for development of the benchmarking infrastructure. The main components of this infrastructure are AWS EC2 instances to deploy java applications and perform computations, Apache JMeter for load testing and results recording, and VisualVM for profiling the java applications.

1.1 Pre-requisites

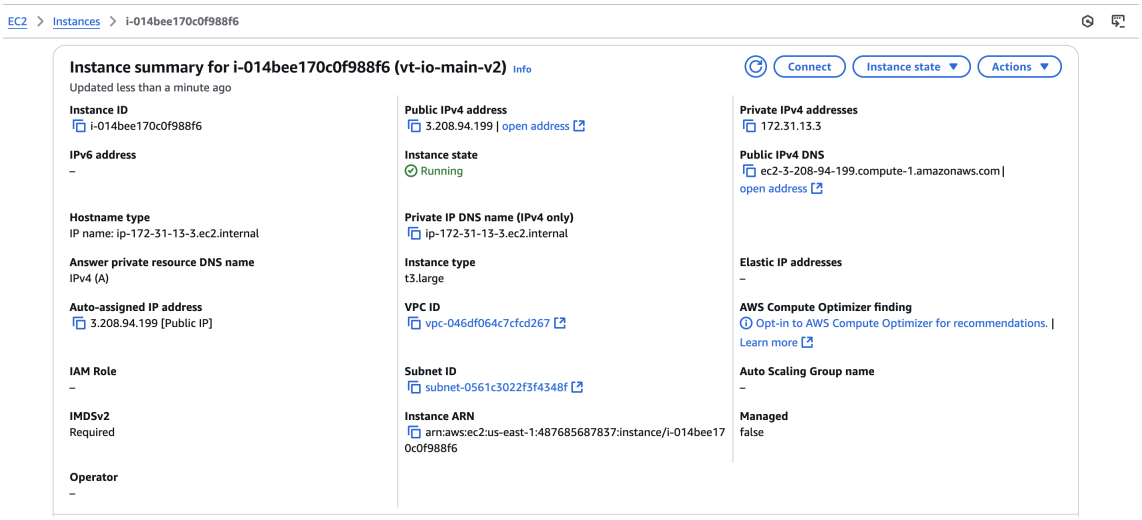
These pre-requisites are essential to perform this benchmarking:

- AWS Account: An AWS Account with valid credentials is required. Valid IAM user with permission to create and use EC2 instances with proper security groups.
- Java: Every instance should have Java 21+ as Virtual Threads were introduced in Java 21.
- Apache Jmeter: Familiarity with creating and applying load testing with the help of Jmeter and recording results is essential.
- VisualVM: Basic familiarity with profiling Java applications is needed. With the help of VisualVM metrics like Memory usage and CPU utilization is calculated.
- Text Editor: A text editor is needed to develop the applications that needs to be tested. I have used Intellij IDEA for development on my java applications.
- Build Tool: A build tool is needed to create and package Java applications into jar files. I have used Maven in this project.

2 Creating AWS EC2 instances

- Step 1: Log in to the AWS account and open AWS Management Console.
- Step 2: Open EC2 Dashboard
- Step 3: Click Launch instance.

- Step 4: Name the instance - Select Ubuntu Server 24..04 LTS -Select instance type as t3.xlarge - Create or Select the Key pair login of type rsa.
- Step 5: Select or create a security group with inbound ports of 22 and 8080 and outbound for all.
- Step 6: now click on "Review and Launch"
- Step 7: After launching the instance you can ssh into the instance using command "ssh -i "secretKey.pem" ubuntu@given-instance-name.amazonaws.com"
- Step 8: In the same way create a total of 5 instances.

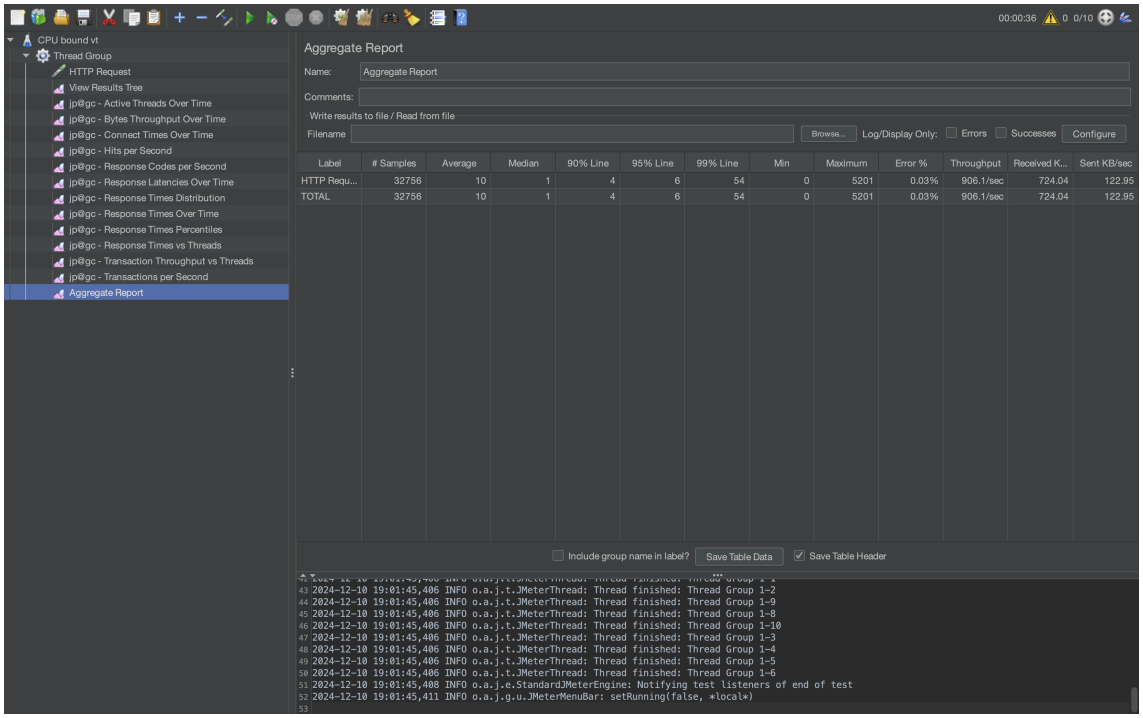


3 Installing Apache Jmeter and VisualVM

These two applications should be installed in the local machine and they will connect with the instance through network calls.

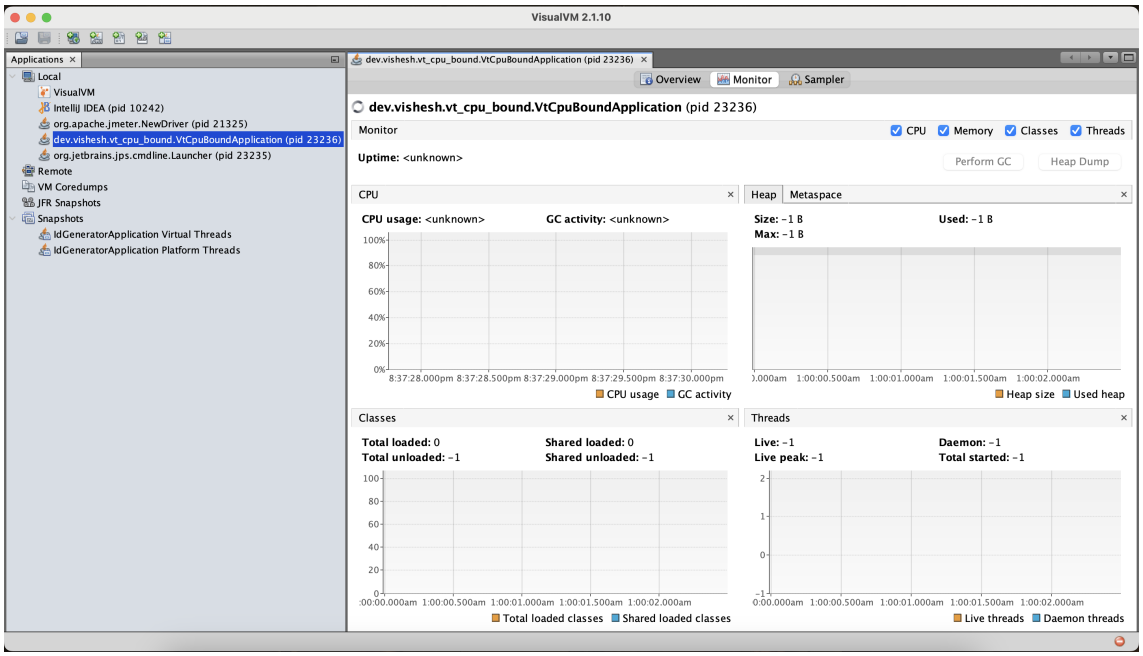
3.1 Apache JMeter

- Step 1: Update the system using "sudo apt update && sudo apt upgrade -y"
- Step 2: Install Java using "sudo apt install default-jre"
- Step 3: Download Apache Jmeter using "wget https://dlcdn.apache.org//jmeter/binaries/apache-jmeter-5.4.1.zip"
- Step 4: Unzip Jmeter using "unzip apache-jmeter-5.4.1.zip"
- Step 5: Go into Jmeter directory using "cd apache-jmeter-5.4.1/bin"
- Step 6: Run Apache Jmeter using "./jmeter" which should open it in GUI mode as shown in the image.



3.2 VisualVM

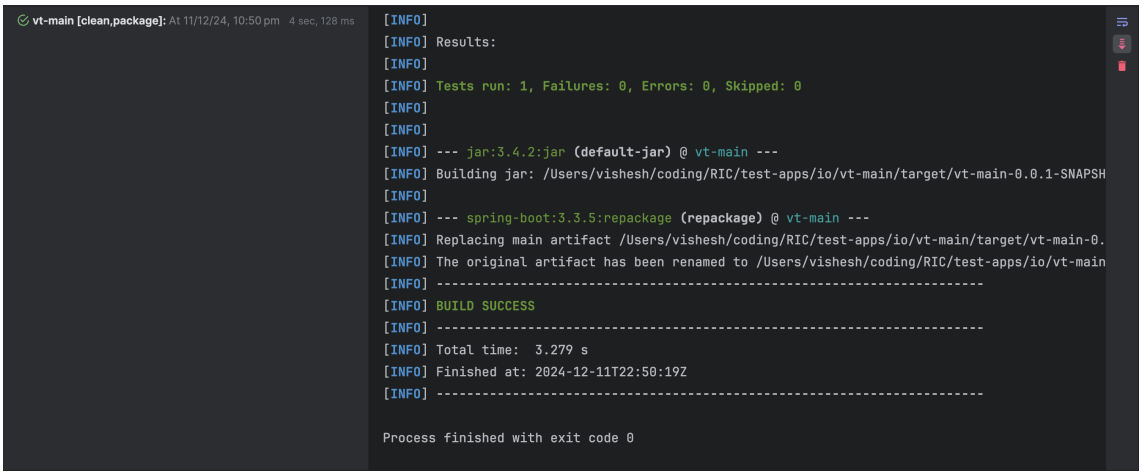
- Step 1: Update the system using "sudo apt update"
- Step 2: Install VisualVM using "sudo apt install visualvm"
- Step 3: Run visualvm and it will open in GUI mode as shown in image.



4 Create Application and Jar Files

- Step 1: Get the source code of all 5 application : vt-cpu-bound, thread-cpu-bound, vt-main, thread-main and vt-blocker.

- Step 2: Update the system using command "sudo apt update".
- Step 3: Install Java in the system using "sudo apt install default-jdk -y"
- Step 4: Download Maven using "wget https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz"
- Step 5: Unzip maven "tar -xvf apache-maven-3.6.3-bin.tar.gz"
- Step 6: Install maven using "mv apache-maven-3.6.3 /opt/"
- Step 7: Go into each application using "cd project-name"
- Step 8: Create Jar for each project using "mvn clean package"
- Step 9: Now transfer the created Jar file to respective AWS EC2 instance using "rsync -avz –progress -e "ssh -i private-key.pem" jarFileName.jar ubuntu@ec2-name.amazonaws.com:/home/ubuntu"



5 Setup on AWS EC2 Instances

- Step 1: Update the instances using "sudo apt update && sudo apt upgrade -y".
- Step 2: Install Java on all the instances using "sudo apt install default-jdk"
- Step 3: REMOTE: Enable JMX on the remote JVM.Add the following parameters to the JVM:
  -Dcom.sun.management.jmxremote.port=9000
  -Dcom.sun.management.jmxremote.ssl=false
  -Dcom.sun.management.jmxremote.authenticate=false
- Step 4: REMOTE: Create a jstatd permissions file. Create a file named permissions.txt with the following contents:
  grant { permission java.security.AllPermission;} ;
- Step 5: REMOTE: Start jstatd. Run jstatd -J-Djava.security.policy=permissions.txt. Leave this running while you monitor the JVM.

- Step 6: LOCAL: Open VisualVM.
- Step 7: LOCAL: Enable VisualVM to use the SSH tunnel.
  - In VisualVM, open menu: Tools – Options – Network
  - Select Manual proxy settings
  - Uncheck Use the same proxy settings for all protocols
  - Set SOCKS Proxy: localhost, Port: 10,000
- REMOTE: Get the IP Address of the server. Run ifconfig and generally you are looking for the ip address after inet addr on eth0 but it may vary.
- LOCAL: Add the Remote Host to VisualVM.
  - In VisualVM, open menu: File – Add Remote Host
  - Add the IP Address from the previous step
  - Under Advanced Settings, remove the default jstatd connection
  - Click OK
  - Right-Click the new host and select "Add JMX Connection..."
  - Enter the IP Address from the previous step with port 9000 as the connection. This should look like: 10.0.0.1:9000.
  - Click OK
  - VisualVM should connect. You should now be able to monitor the remote JVM.

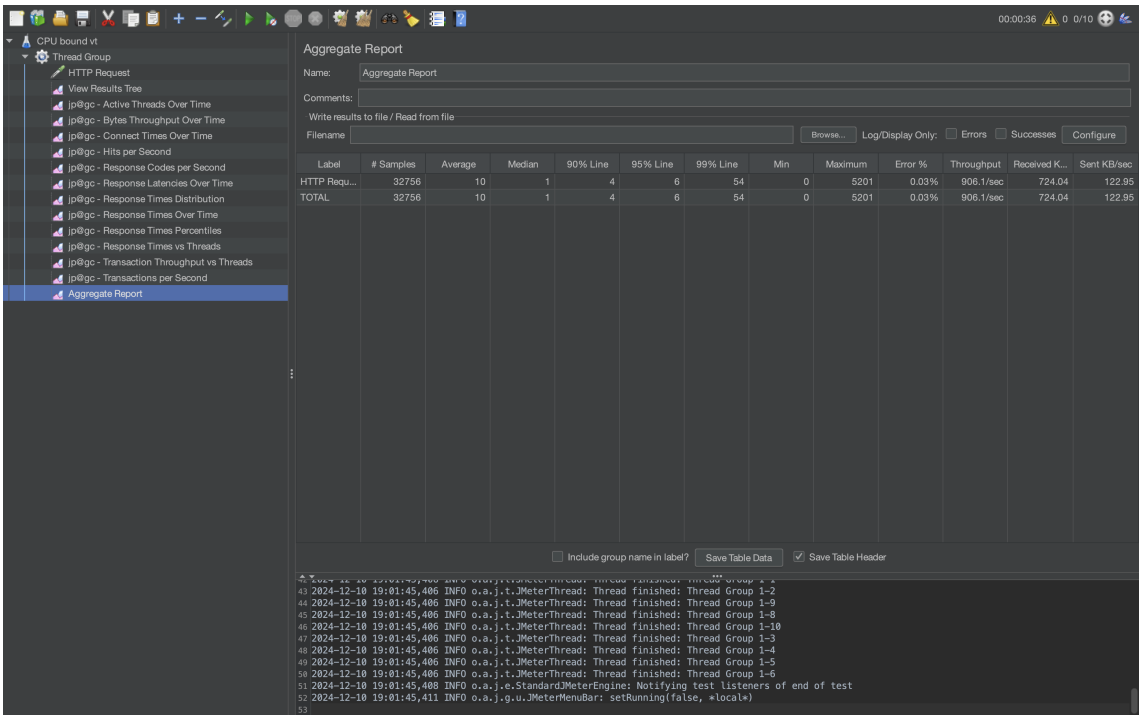
6 Start Java Applications

- Start the Java Applications in every instance using "java -Dcom.sun.management.jmxremote.port=9000 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false jarFileName.jar"

```
ubuntu@ip-172-31-13-3:~$ java -jar vt-io-main-v2.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.5)

2024-12-11T22:52:31.261Z  INFO 1592 --- [vt-main] [           main] dev.vishesh.vt_main.VtMainApplication    : Starting VtMainApplication v0.0.
1-SNAPSHOT using Java 21.0.5 with PID 1592 (/home/ubuntu/vt-io-main-v2.jar started by ubuntu in /home/ubuntu)
2024-12-11T22:52:31.269Z  INFO 1592 --- [vt-main] [           main] dev.vishesh.vt_main.VtMainApplication    : No active profile set, falling b
ack to 1 default profile: "default"
2024-12-11T22:52:34.062Z  INFO 1592 --- [vt-main] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 808
0 (http)
2024-12-11T22:52:34.107Z  INFO 1592 --- [vt-main] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-12-11T22:52:34.108Z  INFO 1592 --- [vt-main] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache
 Tomcat/10.1.31]
2024-12-11T22:52:34.196Z  INFO 1592 --- [vt-main] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded Web
ApplicationContext
2024-12-11T22:52:34.200Z  INFO 1592 --- [vt-main] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: init
ialization completed in 2717 ms
2024-12-11T22:52:35.423Z  INFO 1592 --- [vt-main] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (htt
p) with context path '/'
2024-12-11T22:52:35.478Z  INFO 1592 --- [vt-main] [           main] dev.vishesh.vt_main.VtMainApplication    : Started VtMainApplication in 5.5
38 seconds (process running for 7.42)
```

7 Stress Testing Using Apache Jmeter

- Step 1: Launch the JMeter GUI.
- Step 2: Add test plan elements.
- Step 3: Load and save test plan elements.
- Step 4: Configuring the Thread Group elements.
- Step 5: Enter Number of Threads=100, Ramp-up Period=60, Duration=600.
- Step 6: Create HTTP Request in Thread Group. Enter Server Name=IP address of EC2 instance, Port number=8080, HTTP request=GET, path='/test'.
- Step 7: Run JMeter test plan.



8 Track the metrics

Throughput and Latency will be tracked with Apache Jmeter and Memory usage and CPU utilization will be tracked with VisualVM.